Update on ORNL TRANSFORM Tool: Preliminary Architecture / Modules for High-Temperature Gas-Cooled Reactor Concepts and Update on ALMR Control

Richard E. Hale
David L. Fugate
M. Sacit Cetiner
Syd J. Ball
A. Lou Qualls
John J. Batteh
Modelon, Inc.

August 2015
DOCUMENT AVAILABILITY


Reports produced before January 1, 1996, may be purchased by members of the public from the following source:

National Technical Information Service  
5285 Port Royal Road  
Springfield, VA 22161  
**Telephone** 703-605-6000 (1-800-553-6847)  
**TDD** 703-487-4639  
**Fax** 703-605-6900  
**E-mail** info@ntis.gov  
**Website** [http://www.ntis.gov/help/ordermethods.aspx](http://www.ntis.gov/help/ordermethods.aspx)

Reports are available to DOE employees, DOE contractors, Energy Technology Data Exchange representatives, and International Nuclear Information System representatives from the following source:

Office of Scientific and Technical Information  
PO Box 62  
Oak Ridge, TN 37831  
**Telephone** 865-576-8401  
**Fax** 865-576-5728  
**E-mail** reports@osti.gov  
**Website** [http://www.osti.gov/contact.html](http://www.osti.gov/contact.html)

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.
UPDATE ON ORNL TRANSFORM TOOL: PRELIMINARY ARCHITECTURE / MODULES FOR HIGH-TEMPERATURE GAS-COOLED REACTOR CONCEPTS AND UPDATE ON ALMR CONTROL

Richard E. Hale
David L. Fugate
M. Sacit Cetiner
Syd J. Ball
A. Lou Qualls
John J. Batteh, Modelon, Inc.

Date Published: August 2015

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-BATTELLE, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725
CONTENTS

LIST OF FIGURES ................................................................................................................................. v
LIST OF TABLES ................................................................................................................................. vi
ACRONYMS ........................................................................................................................................ vii
ACKNOWLEDGMENTS ...................................................................................................................... ix
EXECUTIVE SUMMARY .................................................................................................................... xi
1. INTRODUCTION ........................................................................................................................... 1
  1.1 BACKGROUND ................................................................................................................... 1
2. PRELIMINARY HTGR ARCHITECTURE AND MODELS .............................................................. 1
  2.1 HTGR INTRODUCTION .................................................................................................... 1
  2.2 HTGR MODELICA ARCHITECTURE .............................................................................. 4
  2.3 MODELICA VERSUS FORTRAN IMPLEMENTATIONS ................................................ 6
  2.4 HTGR MODELS ............................................................................................................... 7
  2.5 CORE MODEL .................................................................................................................. 7
  2.6 PRIMARY HEAT TRANSPORT SYSTEM WITH A SINGLE-CHANNEL CORE MODEL .......... 9
  2.7 CORE MODEL MODELICA WRAPPER .............................................................................. 13
  2.8 RCCS MODEL .................................................................................................................. 17
  2.9 CROSS VESSEL MODEL .............................................................................................. 17
  2.10 STEAM GENERATOR MODEL .................................................................................... 18
  2.11 BALANCE OF PLANT MODELS .................................................................................. 20
  2.12 EXAMPLE SIMULATIONS ............................................................................................. 20
3. ALMR END-TO-END CONTROL UPDATE .................................................................................... 24
  3.1 MAIN STEAM TURBINE CONTROL VALVE .................................................................... 24
  3.2 FEEDWATER HEATER CONTROL ................................................................................ 29
4. PRELIMINARY VALIDATION AND VERIFICATION STRATEGY ............................................... 37
  4.1 DEVELOPMENT ARCHITECTURE/ENVIRONMENT ...................................................... 37
  4.2 MODEL DEVELOPMENT AND PRODUCTION OVERVIEW ......................................... 38
    4.2.1 Sandbox Area ........................................................................................................... 39
    4.2.2 Development Branch ............................................................................................... 39
    4.2.3 Merging, Testing, and QA ...................................................................................... 41
    4.2.4 Production Deployment ......................................................................................... 42
  4.3 MODEL RELEASE AND ACCESS .................................................................................... 44
5. CONCLUSIONS ............................................................................................................................ 44
6. REFERENCES ............................................................................................................................... 45
APPENDIX A .......................................................................................................................................... A-1
**LIST OF FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 1. High-temperature gas-cooled reactor: isometric [8]</td>
<td>2</td>
</tr>
<tr>
<td>Fig. 2. High-temperature gas-cooled reactor: prismatic core/vessel details [10]</td>
<td>3</td>
</tr>
<tr>
<td>Fig. 3. High-temperature gas-cooled reactor system [9]</td>
<td>4</td>
</tr>
<tr>
<td>Fig. 4. ALMR Modelica/Dymola architecture [5]</td>
<td>5</td>
</tr>
<tr>
<td>Fig. 5. FHR Modelica/Dymola architecture [4]</td>
<td>5</td>
</tr>
<tr>
<td>Fig. 6. HTGR Modelica/Dymola architecture (power production)</td>
<td>5</td>
</tr>
<tr>
<td>Fig. 7. ALMR Modelica/Dymola architecture (process heat production)</td>
<td>6</td>
</tr>
<tr>
<td>Fig. 8. HTGR core fuel choices [8]</td>
<td>7</td>
</tr>
<tr>
<td>Fig. 9. NGNP core model choices: prismatic (L) and pebble bed (R) [1]</td>
<td>8</td>
</tr>
<tr>
<td>Fig. 10. NGNP prismatic core detail [9]</td>
<td>8</td>
</tr>
<tr>
<td>Fig. 11. Power density profile as a function of axial position</td>
<td>11</td>
</tr>
<tr>
<td>Fig. 12. A simple configuration of the HTGR PHTS with a single-channel reactor core model</td>
<td>12</td>
</tr>
<tr>
<td>Fig. 13. Nodal mean coolant temperature profile as a function of axial channel position</td>
<td>13</td>
</tr>
<tr>
<td>Fig. 14. Simplified Modelica Fortran external call wrapper example</td>
<td>14</td>
</tr>
<tr>
<td>Fig. 15a. Modelica Fortran external call wrapper “core” example</td>
<td>16</td>
</tr>
<tr>
<td>Fig. 15b. Simplified Fortran example “core” code</td>
<td>16</td>
</tr>
<tr>
<td>Fig. 15c. Simplified Modelica “CoreTest” model for Fortran implementation</td>
<td>17</td>
</tr>
<tr>
<td>Fig. 16. Modelica representation of the cross-vessel subsystem</td>
<td>18</td>
</tr>
<tr>
<td>Fig. 17. Modelica representation of the steam generator subsystem</td>
<td>19</td>
</tr>
<tr>
<td>Fig. 18. Variation of helium mass flow rate in a single average channel due to partial loss of flow</td>
<td>20</td>
</tr>
<tr>
<td>Fig. 19. Variation of normalized reactor power in response to partial loss of flow</td>
<td>21</td>
</tr>
<tr>
<td>Fig. 20. Variation of helium temperatures as a function of time</td>
<td>22</td>
</tr>
<tr>
<td>Fig. 21. (top) Helium pressure dynamics in the channel in response to partial loss of flow rate and (bottom) variation of pressure drop across the core channel</td>
<td>22</td>
</tr>
<tr>
<td>Fig. 22. Pressure vessel temperatures in response to the partial loss of primary helium flow rate</td>
<td>23</td>
</tr>
<tr>
<td>Fig. 23. Helium axial temperature profiles for different flow rates</td>
<td>24</td>
</tr>
<tr>
<td>Fig. 24. Turbine speed control concept diagram</td>
<td>25</td>
</tr>
<tr>
<td>Fig. 25. Baseline ModelPower conversion system with throttle valve added to high pressure turbine supply</td>
<td>26</td>
</tr>
<tr>
<td>Fig. 26. Throttle valve parameters</td>
<td>26</td>
</tr>
<tr>
<td>Fig. 27. Control loop algorithm</td>
<td>27</td>
</tr>
<tr>
<td>Fig. 28a. Power plant dynamic test from 111 to 151MW with high pressure (HP) steam turbine throttle valve control</td>
<td>27</td>
</tr>
<tr>
<td>Fig. 28b. Power plant dynamic test from 111 to 151 MW with HP steam turbine throttle valve control</td>
<td>28</td>
</tr>
<tr>
<td>Fig. 29. Power plant dynamic test from 111 to 151 MW with HP steam turbine throttle valve control</td>
<td>28</td>
</tr>
<tr>
<td>Fig. 30. Feedwater heater example</td>
<td>29</td>
</tr>
<tr>
<td>Fig. 31. PRISM diagram for one turbine-generator system [Fig. 10.1-1 in Ref. 12]</td>
<td>32</td>
</tr>
<tr>
<td>Fig. 32. PRISM diagram for extraction steam system flow [Fig. 10.3-2 in Ref. 12]</td>
<td>33</td>
</tr>
<tr>
<td>Fig. 33. PRISM diagram for condensate system flow [Fig. 10.2-2 in Ref. 12]</td>
<td>34</td>
</tr>
<tr>
<td>Fig. 34. PRISM diagram for feedwater system flow [Fig. 10.2-3 in Ref. 12]</td>
<td>35</td>
</tr>
<tr>
<td>Fig. 35. PRISM diagram for heater drains system flow [Fig. 10.2-4 in Ref. 12]</td>
<td>36</td>
</tr>
<tr>
<td>Fig. 36. Sample issues from a GitHub repository</td>
<td>39</td>
</tr>
<tr>
<td>Fig. 37. Sample page for an individual issue with comments</td>
<td>40</td>
</tr>
<tr>
<td>Fig. 38. Branch creation serves as the start of the GitHub flow</td>
<td>40</td>
</tr>
</tbody>
</table>
Fig. 39. Commits on the branch by the development team ................................................................. 41
Fig. 40. Opening a pull request ......................................................................................................... 42
Fig. 41. Code review and discussion ............................................................................................... 42
Fig. 42. Deployment initiation following completion of branch testing ............................................. 43
Fig. 43. Merge from development branch into master ..................................................................... 43

LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1. Design data for HTGR steam generator</td>
<td>19</td>
</tr>
<tr>
<td>Table 2. Advanced reactor modeling environments</td>
<td>38</td>
</tr>
<tr>
<td><strong>ACRONYMS</strong></td>
<td><strong>DESCRIPTION</strong></td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td>ALMR</td>
<td>advanced liquid-metal reactor</td>
</tr>
<tr>
<td>ART</td>
<td>Advanced Reactor Technology</td>
</tr>
<tr>
<td>DOE</td>
<td>US Department of Energy</td>
</tr>
<tr>
<td>DRACS</td>
<td>direct reactor auxiliary cooling system</td>
</tr>
<tr>
<td>FH</td>
<td>feedwater heating</td>
</tr>
<tr>
<td>FHR</td>
<td>fluoride high-temperature reactor</td>
</tr>
<tr>
<td>FMIE</td>
<td>functional mockup interface for Excel</td>
</tr>
<tr>
<td>FMU</td>
<td>functional mockup units</td>
</tr>
<tr>
<td>GRSAC</td>
<td>Graphite Reactor Severe Accident Code</td>
</tr>
<tr>
<td>HP</td>
<td>high pressure (turbine)</td>
</tr>
<tr>
<td>HTGR</td>
<td>high-temperature gas-cooled reactor</td>
</tr>
<tr>
<td>HTR</td>
<td>heater</td>
</tr>
<tr>
<td>I&amp;C</td>
<td>instrumentation and controls</td>
</tr>
<tr>
<td>IHTS</td>
<td>intermediate heat transport system</td>
</tr>
<tr>
<td>IHX</td>
<td>intermediate heat exchanger</td>
</tr>
<tr>
<td>LOFC</td>
<td>loss of forced circulation</td>
</tr>
<tr>
<td>LOOP</td>
<td>loss of electrical power</td>
</tr>
<tr>
<td>LP</td>
<td>low pressure (turbine)</td>
</tr>
<tr>
<td>MHTGR</td>
<td>modular high-temperature gas reactor</td>
</tr>
<tr>
<td>NGNP</td>
<td>next generation nuclear plant</td>
</tr>
<tr>
<td>NHSS</td>
<td>nuclear heat supply system</td>
</tr>
<tr>
<td>NQA</td>
<td>nuclear quality assurance</td>
</tr>
<tr>
<td>ORNL</td>
<td>Oak Ridge National Laboratory</td>
</tr>
<tr>
<td>PCS</td>
<td>power conversion system</td>
</tr>
<tr>
<td>PHTS</td>
<td>primary heat transport system</td>
</tr>
<tr>
<td>PRISM</td>
<td>power reactor innovative small module</td>
</tr>
<tr>
<td>QA</td>
<td>quality assurance</td>
</tr>
<tr>
<td>RCCS</td>
<td>reactor cavity cooling system</td>
</tr>
<tr>
<td>RPV</td>
<td>reactor pressure vessel</td>
</tr>
<tr>
<td>SMR</td>
<td>small modular reactor</td>
</tr>
<tr>
<td>TRANSFORM</td>
<td>Transient Simulation Framework of Reconfigurable Models</td>
</tr>
<tr>
<td>V&amp;V</td>
<td>verification and validation</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

The research described in this report was sponsored by the Advanced Reactor Technology (formerly Advanced Reactor Concept) Program of the US Department of Energy Office of Nuclear Energy.
EXECUTIVE SUMMARY

The Small Modular Reactor (SMR) Dynamic System Modeling Tool project is in its fourth year of development. The project supports collaborative modeling and study of various advanced SMR (non-light water cooled reactor) concepts, including the use of multiple coupled reactors at a single site.

Previous deliverables focused on development of component and system models, as well as end-to-end system models, using Modelica and Dymola for two advanced reactor architectures: (1) the advanced liquid metal reactor and (2) the fluoride high-temperature reactor. This report focuses on the initial development of architecture and preliminary modules for the high-temperature gas-cooled reactor. The example chosen is based on the next generation nuclear plant development described in Ref. 1. The initial core model is both developed within Modelica and presented as a Modelica wrapper around an existing Fortran code as discussed and documented in Ref. 2.

Furthermore, improvements to the end-to-end control system for the advanced liquid-metal reactor model have been added. These include mainsteam turbine valve and feedwater heater control system updates. The mainsteam turbine valve model has been developed and documented in this report, along with initial testing to determine the sensitivity to operations for valve control. The power reactor innovative small module (PRISM) design for feedwater heaters and their control has been reviewed and used to develop the requirements for modeling them [12]. The requirements for the feedwater heater and its control indicate the inherent complexity of balancing the heat transfer and flow with the steam drum operation. Further work towards development of a full feedwater heater system model remains. The strategy for model validation and verification is discussed in this report. This strategy uses the developed GitHub repository to provide the appropriate staging areas for workflow development of code, beginning with the initial preliminary models and progressing through to the final production code. The production code is available for web application simulation or model cosimulation with other platform models. Local directories are used as a sandbox area, along with three branches of the repository that have been identified to support this work flow.

As noted in the previous update, in 2015, the project has transitioned from the Advanced SMRs Research and Development Program to the Advanced Reactors Technology (ART) Program to promote safety, technical, economic, and environmental advancements of innovative Generation IV nuclear energy technologies. The combined simulation environment and suite of models have been identified as the Transient Simulation Framework of Reconfigurable Models (TRANSFORM) tool. Critical elements of this effort include (1) defining a standardized, common simulation environment to be applied throughout the ART Program, (2) developing a library of baseline component modules to be assembled into full plant models using available geometry, design, and thermal-hydraulic data, (3) defining modeling conventions for interconnecting component models, and (4) establishing user interfaces and support tools to facilitate simulation development (i.e., configuration and parameterization), execution, and results display and capture.

These efforts have resulted in a set of streamlined tools and models to be used throughout the ART Program. This report is the final deliverable to accomplish this goal, providing a collaborative foundation as a path forward for future continued development and use.
1. INTRODUCTION

1.1 BACKGROUND

As documented in previous reports [3–5], the goal of this project is to develop simulation resources and tools to allow a collaborative modeling and control study for various advanced (non-light water reactor), small modular reactor (SMR) configurations. The project has been funded under the Advanced SMR Research and Development Program and is in its fourth and final year of development. However, it is being transitioned to the Advanced Reactor Technology (ART) Program, with less emphasis on SMRs and more emphasis on advanced reactor concepts. Any further development will be transitioned to the ART Program’s priorities.

The high-level objectives of this effort include (1) development of initial Modelica end-to-end system models for the ALMR and fluoride high-temperature reactor (FHR) design concepts, (2) development of the initial instrumentation and controls (I&C) overlays for the reactor and primary system as well as for critical balance of plant systems, and (3) create a library of models and user interfaces that support the further collaborative development of advanced reactor concepts. These objectives have been met and are documented in previous reports [3–5]. This report documents the progress to meet the final objective, which focuses on extending the range of advanced reactor concepts to include high-temperature gas-cooled reactors (HTGRs). Furthermore, work on end-to-end control of the balance of plant systems continues. This report considers the complex dynamics of the main steam turbine valve and the feedwater heater control. Development of I&C for these systems follows industry practices and demonstrates the challenges of control stability. Finally, the widespread use of these models for execution and development will require workflows and procedures that allow for simultaneous development of simulation models by multiple collaborators. This document also reviews validation and verification (V&V) of the resulting models and workflows as required before final production implementation.

2. PRELIMINARY HTGR ARCHITECTURE AND MODELS

2.1 HTGR INTRODUCTION

Documented previous work [3–5] defines a flexible architecture and structure using Dymola and Modelica to support modeling of various advanced reactor power plant concepts, including the advanced liquid-metal reactor (ALMR) and the FHR. This modeling architecture and structure is the Transient Simulation Framework Of Reconfigurable Models (TRANSFORM) tool, which supports the selection and simulation of different power plant configurations and components, including different I&C configurations. A third potential advanced reactor concept is the HTGR concept. The concept chosen for development here is based on the next generation nuclear plant (NGNP) as described in Ref. 1. The HTGR is a modular helium-cooled reactor concept with features that improve its inherent safety. The reactor and the nuclear heat supply system (NHSS) are comprised of three major components: the reactor, a heat transport system, and a cross vessel that routes the helium between the reactor and the heat transport system. The NHSS supplies energy in the form of steam and/or high temperature fluid that can be used to (1) generate electricity highly efficiently and (2) to support a wide range of industrial processes requiring high-temperature process heat. This concept has a number of attractive features. These features include the range of power ratings, temperatures, and heat transport system configurations that provides flexibility in adapting the modules to the specific application.
The unique design features of the HTGR concept (Figs. 1–3) present several modeling differences from prior efforts. An isometric of the basic HTGR system concept designed for power production is seen in Fig. 1. Detail of the vessel and core can be seen in Fig. 2, along with an overall system schematic for either power or process heat production in Fig. 3. The use of single-phase helium as a coolant and the form of the fuel are the most obvious differences. Together, the design features allow for greater potential reactor safety and improvements associated with high-temperature energy generation efficiencies, as well as process heat applications. Challenges associated with HTGRs need to be modeled and understood before the system can be developed, optimized, and finalized.

Fig. 1. High-temperature gas-cooled reactor: isometric [8].
Fig. 2. High-temperature gas-cooled reactor: prismatic core/vessel details [10].
2.2 HTGR MODELICA ARCHITECTURE

The implementation of this concept into a Modelica-based model is similar to the adaptation of the other concepts previously considered (ALMR and FHR). The architecture makes use of the same structure developed earlier, with the major components replaced with HTGR specifics. In particular, a reactor cavity cooling system (RCCS) replaces the direct reactor auxiliary cooling system (DRACS) system with HTGR-specific components developed using helium cooling with a steam generator for power production or a helium-to-molten-salt heat exchanger used for process heat applications. The existing architectures for the ALMR and FHR and the proposed architecture for the HTGR are represented in Figs. 4–7. There are two potential architectures for the HTGR. For process heat applications, the heat exchanger is a helium-to-molten-salt heat exchanger that delivers source heat to the application of interest. For power generation, this heat exchanger is replaced with one that connects to a steam generator for power production. Examples of these Modelica/Dymola architectures are shown in Figs. 6–7.
Fig. 4. ALMR Modelica/Dymola architecture [5].

Fig. 5. FHR Modelica/Dymola architecture [4].

Fig. 6. HTGR Modelica/Dymola architecture (power production).
The initial HTGR starting point models were developed in Fortran. Consistent with this, a multistep approach has been used by other industries to adopt and develop models from different languages into Modelica. The first step is to create a program that wraps around the Fortran code. This program calls the code and returns values to the Modelica module. This approach uses existing code that has often been validated and verified, so the initial Modelica code can be easily developed and benchmarked against the existing code. However, with this approach, the control of the simulation is no longer available in the Modelica environment. Rather, variables are passed back and forth between Modelica and Fortran and the Modelica solver cannot optimize the simulation. The ability to down-select different architectures and components through Modelica is retained, however. Reconstituting Fortran models within Modelica provides modeling equations within the Modelica environment to allow for simulation control and to provide for easy leveraging to other potential implementations by copying and pasting applicable systems and component code in other Modelica objects. This process may be time consuming, so its relative value should be evaluated for each case. If it is judged to be worthwhile, then the implemented Modelica models can be benchmarked against the Modelica wrapper models that were benchmarked against the original Fortran models. This chain of V&V allows for a transition between the Fortran models and full Modelica-based system models. In this deliverable, the initial activity is to develop both an initial Modelica HTGR core model as well as demonstrate the ability to create the Modelica wrappers around the existing Fortran code. These initial models are described below.

### 2.3 MODELICA VERSUS FORTRAN IMPLEMENTATIONS

Code implementation of a core model both with and without a “wrapper” is included to highlight Fortran implementation. The equations represented in the Fortran discussions may be used later to provide comparison with the Modelica model. The optimum strategy for incorporating Fortran code into a Modelica model includes several considerations. Rewriting the Fortran code into a Modelica representation of the underlying physics in the system or component model would allow the Modelica solver (like Dymola) to optimize the solution of the system of equations. Without this Modelica implementation, the separate component/system models linked together in Modelica all act as black box subroutines that return values based on inputs. While Modelica readily handles external calls to Fortran code, complex interactions between multiple systems and components are usually handled best within Modelica to allow the solver to monitor the time steps and to converge these systems toward an optimal
solution. Therefore, an advantage of Modelica is the potential for improved convergence and rapid simulations. However, this is not always the case where existing Fortran code runs rapidly. If the equations are brought into Modelica, the ability to reproduce these models for other components and systems is also considerably improved. Conversely, an advantage to retaining models as Modelica external calls to Fortran is the speed with which existing models can be incorporated into a Modelica framework. These models in many cases have already completed V&V, and minimizing the modification to the code improves the potential and speed for generating V&V Modelica code. This is of considerable interest for the range of systems codes that exist within the accident simulation world for nuclear designs. In either case, the highly desirable ability to choose different implementations of the architecture as part of a graphical drop-down interface can be retained. The choice for whether to redevelop existing Fortran models in Modelica code versus using external calls to Fortran is best decided on a case-by-case basis, considering all of the factors identified above.

### 2.4 HTGR MODELS

The HTGR models are based on the physics representations in Ref. 1; the associated equations and details of the mathematical representation for these models can be found in Appendix A.

### 2.5 CORE MODEL

There are two implementations of the core model for HTGRs based on the fuel type that results in either a prismatic core or a pebble bed core, as shown in Figs. 8–9.

---

**Fig. 8. HTGR core fuel choices [8].**
A brief description of the challenging modeling physics associated with the HTGR core [1] is seen below.

A single-node representation of the temperature and the energy storage in a large hexagonal graphite block fuel element [Fig. 10] or an array of fuel pebbles could not accurately portray the fuel-to-moderator temperature differences that exist at full-power conditions. It would also preclude approximating the at-power reactivity feedback for the neutron kinetics equations because the individual effects of fuel and moderator temperature changes are not modeled. However, for studies of shutdown power and flow scenarios, for which [Graphite Reactor Severe Accident Code] GRSAC is primarily intended, the radial temperature gradients within the blocks and pebble arrays are reduced to small values within a few minutes after shutdown, and the reactivity effects are no longer significant after the reactor is scrammed.

The question remains, however, as to how accurately the single-node-per-element model can be used to predict the temperature transients. In general, the accuracy of any finite differencing scheme for modeling diffusion decreases as the frequency content of the perturbation increases;
and for heat conduction models, the grosser the node mesh size, the more the transient heat flux between nodes is underestimated. In most cases, an underestimation of heat flux between adjacent elements would yield conservative (i.e., higher-than-actual) hot fuel-element temperatures.

Details of the physics equations associated with the core model are found in Appendix A.

2.6 PRIMARY HEAT TRANSPORT SYSTEM WITH A SINGLE-CHANNEL CORE MODEL

A simple primary heat transport system (PHTS) configuration was implemented with a single-channel reactor core model. The reactor is represented by six-group normalized point kinetic equations.

\[
\frac{dn}{dt} = \frac{(\rho - \beta)}{\Lambda} n(t) + \sum_{i=1}^{6} \frac{\beta_i}{\Lambda} c_i(t)
\]

\[
\frac{dc_i}{dt} = \lambda_i \left[ n(t) - c_i(t) \right]
\]

where \(\rho(t)\) is the total reactivity in the multiplying medium, \(\beta_i\) and \(\lambda_i\) are the delayed neutron fractions and the decay constants for the \(i\)th precursor group, \(\Lambda\) is the mean neutron generation time, \(n(t)\) is the normalized prompt neutron flux, \(c_i(t)\) is the normalized \(i\)th-group delayed neutron flux, and \(\beta\) is the total precursor fraction defined as

\[
\beta = \sum_{i=1}^{6} \beta_i
\]

The rate equations are subject to steady state initial condition, i.e.,

\[
\frac{dn}{dt} = 0, \ t = 0
\]

\[
\frac{dc}{dt} = 0, \ t = 0
\]

The delayed portion of normalized heat generation is implemented using Eq. (1)

\[
Q_{n-decam} = 0.1 \left[ (t + 10)^{-\frac{1}{5}} - (t + T_s)^{-\frac{1}{5}} + 0.87(t + T_s + 2 \times 10^7)^{-\frac{1}{5}} - 0.87(t + 2 \times 10^7)^{-\frac{1}{5}} \right]
\]

where \(t\) is time after shutdown and \(T_s\) is the operation time prior to shutdown—both in seconds.

The reactivity feedbacks are modeled as follows:

\[
\rho_f = \alpha_f \left( T_{fe} - T_f \right), \quad \text{(2.a)}
\]

\[
\rho_M = \alpha_M \left( T_{Me} - T_M \right), \quad \text{and} \quad \text{(2.b)}
\]
\[ \rho_t = \rho_{ex} + \rho_{CR} + \rho_f + \rho_M, \]  

(2.c)

where \( \rho_f \) is the fuel Doppler reactivity feedback, \( \rho_M \) is the moderator reactivity feedback, \( \rho_{ex} \) is the external reactivity, \( \rho_{CR} \) is the control-rod reactivity, and \( \rho_t \) is the total reactivity.

The axial neutron flux is considered to have a cosine shape defined as

\[ \varphi(z) = \varphi_{max} \cos \left( \pi \frac{z}{H} \right), \]

(3)

where \( H \) is the active core length.

In order to account for axial leakage, Eq. (3) should be modified to account for extrapolated length, which leads to a chopped-cosine distribution, that is,

\[ \varphi(z) = \varphi_{max} \cos \left( \pi \frac{z - H}{2H_e} \right), \]

(4)

where \( H_e = H + 2\epsilon \) is the extrapolated height of the core, and \( \epsilon \) is the extrapolation distance at the top and the bottom of the active core region.

Similarly, the power density profile, \( q'''(z) \), is proportional to neutron flux profile, that is,

\[ q'''(z) = q'''_{max} \cos \left( \pi \frac{z - H}{2H_e} \right). \]

(5)

Obviously, Eqs. (4) and (5) are acceptable forms for analytical calculations. For nodal computations where local values of variables are averaged over a finite domain, it should be discretized:

\[ \langle q'''_i \rangle = \frac{1}{\Delta z_i} \int_{z_i-1}^{z_i} q'''_{max} \cos \left( \pi \frac{z - H}{2H_e} \right) dz, \]

(6)

where \( \Delta z_i = z_i - z_{i-1} \) is the axial node \( i \). For a uniform mesh size, as adopted in this derivation, the value becomes \( \Delta z_i = H/N \) for a total number of \( N \) nodes.

Taking the integral in Eq. (6) leads to the following expression for node-averaged power density:

\[ \langle q'''_i \rangle = q'''_{max} \frac{N H_e}{\pi H} \left\{ \sin \left[ \pi \frac{H}{H_e} \left( \frac{i - 1}{2N} \right) \right] - \sin \left[ \pi \frac{H}{H_e} \left( \frac{i-1}{2N} - \frac{1}{2} \right) \right] \right\}, \]

(7)

where \( q'''_{max} \) is the ratio of maximum axial power to average power defined as.
\[ q_{\text{max}}''' = f_{\text{pp}} \langle q''' \rangle, \]  

(8)

where \( f_{\text{pp}} \) is the power peaking factor, and \( \langle q''' \rangle \) is the core average power density. The value of \( f_{\text{pp}} = 1.3 \) is used as the default value, but the value can be changed through the user interface.

A continuous and discretized power density profile is plotted in Fig. 11 as a function of axial position.

![Power density profile as a function of axial position.](image)

The reactor kinetics module is coupled to the coolant channel with helium as the primary coolant. As the preliminary implementation, ideal gas equation-of-state model was used to compute the helium properties [15]. This model uses temperature \( T \) and pressure \( p \) as the independent variables. Only density is a function of \( T \) and \( p \) to incorporate compressibility of the substance. All other quantities are calculated only as a function of \( T \). The properties are valid in the range \( 200 \, K \leq T \leq 6000 \, K \).

The core design is based on General Atomics modular high-temperature gas reactor (MHTGR) concept. The geometric parameters were taken from the NGNP point reactor design study [16]. The reactor delivers a thermal power of 600 MW(t). The nominal coolant inlet temperature is \( T_i = 490^\circ C \) and the outlet temperature is \( T_o = 850^\circ C \); the average core temperature differential is approximately \( \Delta T = 360^\circ C \). The total nominal helium flow rate throughout the core is \( \omega = 250 \, \text{kg/s} \). The nominal pressure drop across the core is 25 kPa. The core has an annular configuration that contains 102 fuel columns; with each column having ten stacked fuel elements. Each fuel element contains approximately 100 coolant channels—depending on its location and function.

The Modelica implementation of the simple core configuration is shown in Fig. 12. The model includes a reactor kinetics module, which calculates normalized point power as a function of time. This model block...
delivers a pre-calculated linear heat generation rate to the fuel element block, which is used to compute the thermal conduction of heat into the coolant. The coolant channel is represented by a dynamic pipe model that computes the convective heat transfer from the fuel block.

The dynamic pipe element computes the basic fluid flow behavior as a function of time. This object solves the mass, momentum and energy equations using control-volume formulation for a user-specified number of axial nodes along the flow direction. Friction losses and fluid heat transfer characteristics are captured using proper correlations.

The nominal coolant temperature profile at steady state operation for an average channel is plotted in Fig. 13. The system model reaches steady state equilibrium at an inlet temperature of 491.2ºC and an outlet temperature of 853.4ºC with an average single channel flow rate of $\omega = 0.024$ kg/s and a pressure drop of $\Delta p = 23$ kPa.
In addition to the core flow channel, the primary heat transport system also contains the lower and upper plena; return coolant channel, a circulator and a pressure vessel.

The reactor pressure vessel is represented by a stack of circular rings of metal, and functionally acts as a thermal storage element. The element is connected to the return coolant channel, which carries helium returning from the steam generator up through the annulus around the core shroud through the upper plenum and back into the reactor core. Incorporation of the reactor vessel model has noticeable effects on the transient behavior of the system.

**2.7 CORE MODEL MODELICA WRAPPER**

A second potential workflow for developing advanced reactor system Modelica models includes the use of existing models developed in other languages (principally Fortran) accessed via an external call in Modelica. Modelica is designed to support this. For a Fortran subroutine call an example of the corresponding Modelica code that is included in the Dymola examples is seen in Fig. 14.
Fig. 14. Simplified Modelica Fortran external call wrapper example.

For the purpose of developing HTGR Modelica models from the GRSAC model described in this report and detailed in Ref. 1, a Modelica call and “wrapper” (Fig. 15a) and a simple representative Fortran “core” code (Fig. 15b) was developed to illustrate how Fortran models can be brought into Modelica. Using the simplified function call in Fig. 14 as an example, a more extensive Modelica “wrapper” was developed (Modelica package “CoreTest.mo”) around a simplified Fortran based core model example. The Modelica code is seen in Fig. 15a. The code is roughly divided into three sections indicated by the green, blue and red boxes below. The green box represents the establishment of the Modelica package and initialization of Modelica elements, parameters and variables consistent with the necessary inputs and outputs between the Modelica and Fortran codes. The blue box is the external call to the Fortran routine returning the derivatives of the variables of interest. The red box represents the output returned back to Modelica along with the integration scheme for solving the core element along with the other elements in the end-to-end system model.

```modelica
model Core "sample model for a core implemented in FORTRAN"
replaceable package Medium =
  Modelica.Media.Interfaces.PartialMedium "Medium in the component"
  annotation (choicesAllMatching = true);
parameter Integer n "discretization";
parameter Modelica.SIunits.Length L "core length";
parameter Modelica.SIunits.Temperature T_start "start temperature";
parameter Modelica.SIunits.Pressure p_out "outlet pressure";
parameter Modelica.SIunits.Temperature T_out "outlet temperature";
parameter Modelica.SIunits.MassFlowRate mdot_out "outlet mass flow rate";
parameter Modelica.SIunits.Temperature Tcore[n](start=fill(T_start,n)) "core temperature";
parameter Modelica.SIunits.Temperature Tfluid[n](start=fill(T_start,n)) "fluid temperature in core";
parameter Modelica.SIunits.Density rho_fluid "inlet fluid density evaluated at T and p_out";
parameter Modelica.Fluid.Interfaces.FluidPort_a port_a(redeclare package Medium =
  Medium)
  annotation (Placement(transformation(extent={{90,-10},{110,10}})));
parameter Modelica.Blocks.Interfaces.RealInput T "inlet temperature";
parameter Modelica.Blocks.Interfaces.RealInput mdot "inlet mass flow rate";
boundary (Transfer function)
```

```modelica
```

```modelica
```

```modelica
end Core;
```

```modelica
```
use_m_flow_in=true,
use_T_in=true,
nPorts=1,
redeclare package Medium = Medium
annotation (Placement(transformation(extent={{60,-10},{80,10}})));
annotation (Placement(transformation(extent={{20,-2},{40,18}})));
Modelica.Blocks.Sources.RealExpression temp_source(y=T_out)
annotation (Placement(transformation(extent={{20,-30},{40,-10}})));

// Function with FORTRAN call to subroutine "core" to return der_Tcore and der_Tfluid
// with interface to link parameters, variables, and outputs between Modelica and FORTRAN
function core_derivs
  "Modelica function call to external FORTRAN code core, returns temperature derivatives"
input Integer n;
input Real L;
input Real T_start;
input Real p_out;
input Real rho_fluid;
input Real Tcore[n];
input Real Tfluid[n];
input Real mdot;
input Real T;
output Real der_Tcore[n];
output Real der_Tfluid[n];
external "FORTRAN 77" core(n,L,T_start,p_out,rho_fluid,Tcore,Tfluid,mdot,T,der_Tcore,der_Tfluid)  annotation(Library="core");
end core_derivs;

equation
  // Pressure is equal to downstream volume pressure. Distributed pressures could be calculated if80
  // handled properly
  p_out=port_a.p;
  // Let's assume the mass flow out is equal to the mass flow in but need not be if properly calculated
  mdot_out = mdot;
  // Outlet temperature is last element of Tfluid
  T_out=Tfluid[n];
  // Properties must be calculated consistently using Modelica medium model representation since surrounding
  // components could be implemented in Modelica so sample density calculation made and passed to FORTRAN code for illustration
  rho_fluid = Medium.density_pT(p_out,T);
  // Time derivatives returned from FORTRAN code via call to core Modelica function defined above
  (der_Tcore,der_Tfluid) = core_derivs(n,L,T_start,p_out,rho_fluid,Tcore,Tfluid,mdot,T);
  // Integration of FORTRAN derivatives in Modelica
  der(Tcore)=der_Tcore;
  der(Tfluid)=der_Tfluid;

connect(boundary.ports[1], port_a)  
  annotation (Line(points={{80,0},{100,0}}, color={0,127,255}));
connect(flow_source.y, boundary.m_flow_in) 
  annotation (Line(points={{41,8},{46,8},{60,8}}, color={0,0,127}));
connect(temp_source.y, boundary.T_in) annotation (Line(points={{41,-20},{48,-20},{48,4},{58,4}}, color={0,0,127}));
annotation (Diagram(coordinateSystem(preserveAspectRatio=false, extent={{-100,-100},{100,100}}), Icon(coordinateSystem(preserveAspectRatio=false, extent={{-100,-100},{100,100}}), graphics={
  Rectangle(extent={{-70,60},{80,-60}}, lineColor={28,108,200},
    fillPattern=FillPattern.HorizontalCylinder,
    fillColor={175,175,175}),
  Polygon(points={{-30,-2},{-20,-28},{-12,-22},{24,-32},{26,-24},{48,-34},{52,-22},{64,2},{58,18},{40,24},{34,2},{24,24},{14,12},{8,-2},{-4,12},{-14,}})
}));
In the example, there is a component called Core which is meant to represent a “dummy” core model (Fig. 15b) and includes the function to the external Fortran code called “core.” For the implementation of this, the actual Fortran code is included in the Modelica Resources\Include folder and the compiled code is in Resources\Library folder. It is required that you compile the Fortran code as a .lib file and include the lib file in the directory before simulation.

```fortran
SUBROUTINE CORE(N, L, T_START, P_OUT, RHO_FLUID, TCORE, TFLUID, MDOT, T, DER + _TCORE, DER_TFLUID)
  C This subroutine converts DTIME (time in decimal hours) to individual
  C hours, minutes and seconds
  INTEGER N
  DOUBLE PRECISION L, T_START, P_OUT, RHO_FLUID, TCORE(1:N), TFLUID(1:N),
  + MDOT, T, DER_TCORE(1:N), DER_TFLUID(1:N)
  C dummy expression to make sure that is time variant.
  INTEGER I
  DO 10 I = 1,N
    DER_TCORE(I) = MDOT*0.1
    DER_TFLUID(I) = T*0.01
  10 CONTINUE
  RETURN
END
```

The example core Modelica model was setup so that it looked representative. It has parameters, variables, property calculations, etc. and it has been annotated so that one can see the structure and what would be required to create this sort of interface. The Fortran code returns the derivatives and then Dymola integrates. The FORTRAN code has dummy calculations for the derivatives shown below, that represent the necessary structure.

\[
\begin{align*}
\text{DER\_TCORE}(I) & = \text{MDOT} \times 0.1 \\
\text{DER\_TFLUID}(I) & = T \times 0.01
\end{align*}
\]

The Modelica test model “CoreTest” as seen in Fig. 15c includes some other components to represent how the Core model would exist as a component in a system integrated with other components in Modelica.
The larger point to be made here is that there may be instances in which it is simpler and easier to make an external call to a program and return a value than it is to develop the code in Modelica. However, there is a tradeoff. Models developed in Modelica can be optimized for potential solutions easier than can be done with externally called code. These considerations are important when deciding how best to implement dynamic system models in Modelica.

### 2.8 RCCS Model

Like many advanced reactor concepts, the HTGR includes decay heat and passive vessel and core cooling systems to allow for normal shutdown and decay heat removal, even with a loss of electrical power (LOOP) transient. Reactor pressure vessel (RPV) heat removal occurs in loss of forced circulation (LOFC) events by the RCCS, where most (typically ~70–90%) of the heat transferred from the RPV to the RCCS is by thermal radiation, and the balance is by natural convection in the reactor cavity air. RCCS degradation and failures involving reductions in coolant flows can currently be modeled in GRSAC (v2.6); however, complete flow stoppages, where heat losses are primarily through the RCCS structure to the surrounding cavity, are not modeled. The RCCS model makes use of some concepts previously considered in DRACS, and it occupies the architectural DRACS element previously detailed in the ALMR architecture. In particular, the use of natural convection cooling for both systems allows for leveraging of existing models to be relatively easily tailored to the HTGR system.

### 2.9 Cross Vessel Model

The cross vessel model is essentially a transport model between the reactor core and vessel and the steam generator and/or heat exchanger that delivers either steam or a heated fluid (typically a molten salt) to power or process heat applications. This transport model performs the same function and will be based on the intermediate heat transport system (IHTS) models developed for the ALMR and FHR systems. Appropriate changes to sizes, dimensions, and fluid transport factors are expected.

The cross-vessel dynamics is modeled by two parallel pipes interacting through two concentric metal walls, which are separated by a thermal insulator. The geometric parameters were obtained from Ref. 16. The diagram layer of the design is shown in Fig. 16.
2.10 STEAM GENERATOR MODEL

The steam generator model includes heat transfer from the helium coolant to water for the production and pressure control of steam output. Although helium heat exchange has not been considered in the previous steam generator models, the details of the heat exchange and the fluid properties will be adjusted from previous Modelica implementations of the ALMR and FHR concepts to develop this component model. Reference of the details to be modified for this model can be found in previous reports [3–5] and in the HTGR GRSAC model [1].

The Modelica diagram layer of the steam generator subsystem is shown in Fig. 17. The steam generator dynamics are represented by two dynamic pipe elements interacting via a metal wall through counter-flow convective heat transfer. The primary coolant, helium, flows on the shell side, and the secondary coolant, water-steam-superheated steam, flows in the tubes.
The steam generator design parameters were obtained from Ref. 17. A partial list of design data is shown in Table 1.

### Table 1. Design data for HTGR steam generator

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heat duty (MWt)</td>
<td>547</td>
</tr>
<tr>
<td>Bundle height (mm)</td>
<td>3793</td>
</tr>
<tr>
<td>Helium inlet temperature (°C)</td>
<td>850</td>
</tr>
<tr>
<td>Helium outlet temperature (°C)</td>
<td>490</td>
</tr>
<tr>
<td>Helium flow rate (kg/s)</td>
<td>250</td>
</tr>
<tr>
<td>Helium inlet pressure (MPa)</td>
<td>7</td>
</tr>
<tr>
<td>Helium pressure drop (kPa)</td>
<td>24</td>
</tr>
<tr>
<td>Water inlet temperature (°C)</td>
<td>200</td>
</tr>
<tr>
<td>Steam outlet temperature (°C)</td>
<td>538</td>
</tr>
<tr>
<td>Water flow rate (kg/s)</td>
<td>216</td>
</tr>
<tr>
<td>Feedwater inlet pressure (MPa)</td>
<td>18.2</td>
</tr>
<tr>
<td>Steam outlet pressure (MPa)</td>
<td>17.2</td>
</tr>
<tr>
<td>Number of tubes</td>
<td>441</td>
</tr>
<tr>
<td>Tube mid-wall temperatures (°C)</td>
<td></td>
</tr>
<tr>
<td>Feedwater inlet</td>
<td>332</td>
</tr>
<tr>
<td>Evaporator inlet</td>
<td>483</td>
</tr>
<tr>
<td>Evaporator exit</td>
<td>551</td>
</tr>
<tr>
<td>Initial super heater inlet</td>
<td>551</td>
</tr>
<tr>
<td>Finishing super heater inlet</td>
<td>597</td>
</tr>
<tr>
<td>Finishing super heater outlet</td>
<td>720</td>
</tr>
</tbody>
</table>
2.11 BALANCE OF PLANT MODELS

The balance-of-plant models for power production make use of many of the modules developed previously for the ALMR and FHR models. For the power production architecture displayed in Fig. 6, these include the power conversion system (PCS) and the grid models. These are described fully in Ref. 3 and are displayed in Fig. 4, so they are not reproduced here. For the process heat application, no production of power is assumed. Rather, there is a direct delivery of heat to applications such as seawater desalination, hydrogen production, district heating, tertiary oil recovery, and other industrial applications. A specific application with heat loads and system design and parameters must be defined further to complete the end-to-end system for process heat applications. This will be considered for future work but is out of the scope of this deliverable.

2.12 EXAMPLE SIMULATIONS

This section is intended to demonstrate the developed simulation capability for HTGRs.

The transient is partial loss of flow in the primary heat transport loop, where 50% of the circulator capacity is considered lost. The nominal flow rate is assumed recovered after a certain period. The variation of mass flow rate in the primary loop as a result of circulator loss of performance is shown in Figs. 18–19.

![Graph showing variation of helium mass flow rate in a single average channel due to partial loss of flow.](image)

Fig. 18. Variation of helium mass flow rate in a single average channel due to partial loss of flow.

The partial loss of mass flow rate through the core results in degraded heat rejection performance, which results in increased fuel and moderator temperatures. Elevated temperatures, in turn, lead to a very fast response due to fuel and moderator temperature feedbacks, resulting in lower reactor power.
Fig. 19. Variation of normalized reactor power in response to partial loss of flow.

It should be noted that this implementation only considers the open-loop point (0-D) response of the reactor core. Xenon dynamics are not yet included in this implementation.

The dynamic response of helium temperatures in an average coolant channel is plotted in Fig. 20. The figure contains sixteen temperature traces that correspond to the temperature response of individual axial nodes along the channel. The first node, i.e., the lowest temperature trace, corresponds to the coolant entering the reactor core, and the last node, i.e., the highest temperature trace, corresponds to the exiting fluid. The helium outlet temperature increases by about 40°C and the core temperature difference increases from approximately 360°C to 400°C.
The reduction in flow rate and increase in fluid temperature impact the pressure dynamics of the coolant. Because the momentum dynamics are included in the simulation, the pressure transient exhibits asymmetry due to compressibility of the fluid (Fig. 21.). Furthermore, the helium viscosity varies as a function of temperature. Unlike liquids, gases exhibit a proportional relationship between temperature and viscosity; that is, the coolant viscosity increases as the coolant temperature increases.

Fig. 20. Variation of helium temperatures as a function of time.

Fig. 21. (top) Helium pressure dynamics in the channel in response to partial loss of flow rate and (bottom) variation of pressure drop across the core channel.
Changes in primary helium temperatures slightly impact the pressure vessel temperatures as shown in Fig. 22. As seen in the figure, the wall temperatures are not much impacted by the slight increase in the helium temperature due to its large thermal inertia.

![Graph of pressure vessel temperatures in response to the partial loss of primary helium flow rate.](image)

**Fig. 22.** Pressure vessel temperatures in response to the partial loss of primary helium flow rate.

It was observed that incorporation of pressure vessel thermal dynamics significantly slows down the simulation as it results in a highly stiff formulation.

It should be noted that the pressure vessel thermal model did not include axial conduction mode, which may affect the thermal response under extreme temperature conditions. This capability will be considered in the future releases of TRANSFORM package.

The variation of helium axial temperature profile in response to changes in coolant mass flow rate is shown in Fig. 23. The figure includes three temperature traces; with red trace representing the nominal profile, and blue and magenta traces representing the equilibrium profile with 25 and 50% loss of flow rate.
Modules for I&C overlays and event drivers have been developed for the models discussed in [3–5]. These I&C models were developed following the chosen flexible modeling architecture, and they demonstrated the ability to adjust the power plant model to changing power outputs while maintaining proper temperatures, flows, and pressures in the power plant. Previous work with I&C development included control of the PHTS reactor power output using the control rod position, the PHTS reactor core outlet temperature using the PHTS fluid temperature, intermediate heat transport system (IHTS) fluid temperature using the IHTS cooling pump, and PCS electric power generation by adjusting the steam flow to the turbine. Two control strategy approaches were developed, with control based on a desired temperature or the temperature difference of key power plant temperatures. In addition to reactor control, models were developed for steam generator control [3]. For a complete end-to-end I&C system for the principal basic functions of a nuclear reactor system, two remaining control functions are essential: the main steam turbine control valve, and the feedwater heater control. These control systems constitute the balance-of-plant control necessary to regulate power production to the grid, and they process system support for optimized steam generation. Their further development is discussed in the subsections below.

3.1 MAIN STEAM TURBINE CONTROL VALVE

In modern power plants, a throttling or governing steam valve is used to adjust the steam pressure and flow supplied to the turbines to regulate a constant speed during varying generation loads. The turbine speed must always be compatible with the generation frequency. As load changes, the turbine will inherently react to the generation load change with a speed change which requires small amplitude and fast acting control adjustments to the steam input. This control is performed by a steam regulation valve that regulates the direct supply flow into the turbine or a bypass flow path. Figure 24 illustrates the
relationship of the valve area to steam pressure and flow, which ultimately affects the generator speed and the electrical frequency. This approach is not used to vary the actual power output; that is performed by controlling the energy provided to the turbine by the reactor and the heat transport systems.

The baseline liquid metal reactor model presented in Ref. 11 was examined for adding a throttle valve to the high pressure turbine steam supply in the power conversion system. Figure 25 illustrates the updated model with the throttle valve and the associated control concept. The throttle valve parameters are shown in Fig. 26. These parameters were estimated based on power reactor innovative small module (PRISM) reference data [12] but require further validation due to limited information on the steam control subsystem.

The valve control method is a proportional, derivative-based approach that compares the actual generation electrical frequency with the desired reference and augments the valve area more open or closed based on the frequency error (Fig. 27). The use of a derivative function is due to the desire to react quickly in a predictive manner. The derivative acts due to the rate of change of the frequency error which is sensitive to quick changes to the generator frequency due to a power plant dynamic change. Figures 28a and 28b illustrate that the throttle valve dynamically closing ~20% due to the control can reduce the generation power output overshoot (Fig. 28a) and also the frequency error (Fig. 28b) during a power plant dynamic test of increasing the power ~36% from 111MW to 151MW. The peak frequency error was reduced ~28% with the addition of the throttle valve control. The actual values of frequency error and the dynamic behavior should be considered as representative only because the turbine and generator inertia values are conceptual. Fig. 29 illustrates the throttle valve effect on pressure and flow for the steam turbine. The pressure is increased ~14%, which decreases the enthalpy flow ~4% (Fig. 29).

Future work would include integrating the steam turbine valve control with the steam generator level control to provide a multivariable control for load following and responding to system failures and degradation. The steam turbine valve parameters should be enhanced to provide the desired flow response characteristics for different operating conditions.
Fig. 25. Baseline ModelPower conversion system with throttle valve added to high pressure turbine supply.

Fig. 26. Throttle valve parameters.
Fig. 27. Control loop algorithm.

Fig. 28a. Power plant dynamic test from 111 to 151MW with high pressure (HP) steam turbine throttle valve control.
Fig. 28b. Power plant dynamic test from 111 to 151 MW with HP steam turbine throttle valve control.

Fig. 29. Power plant dynamic test from 111 to 151 MW with HP steam turbine throttle valve control.
3.2 FEEDWATER HEATER CONTROL

In modern power plants, the process referred to as *regeneration*, or preheating the feedwater before entry to the steam generator for boiling, is called *feedwater heating* (FH). This is typically accomplished by extracting some bypass steam from the turbine for the purpose of heating up the feedwater prior to the steam generator with a heat exchanger system [13]. An open FH system design consists of directly mixing high pressure turbine exhaust extraction steam with the feedwater flow to raise its temperature. A closed FH system design consists of using a heat exchanger that does not mix the steam and feedwater directly.

In closed FH systems, the feedwater typically flows through banks of heat exchanger tubes, with steam flowing on the outside of the heat exchanger tubes.

Regeneration by preheating the feedwater reduces negative effects such as thermal shock and irreversibility that are associated with unrestrained expansion. Regeneration also improves the thermodynamic efficiency of the system by increasing the steam generator feedwater inlet temperature. Large steam power plants commonly employ large quantities of FH using multistage extraction to supply steam to the various feedwater heaters [13].

The extraction steam is the thermal energy source input to the FH vessel (Fig. 30). The feedwater typically consists of collected condensed steam from the turbine exhaust and some makeup water. The feedwater is pumped through the heat exchanger, typically a shell and tube type, in the vessel, and it exits at a raised temperature due to the heat transfer. The steam condenses in the heat exchanger as it cools, which creates a condensed steam water level in the vessel. A drain outlet is used to regulate the fluid level in the vessel. During variations in the flows of feedwater or variation in the turbine steam extraction, the heat transfer energy balance can become unstable. The key characteristics of an FH system are the water level, the feedwater inlet and outlet temperatures, the steam inlet temperature, and the drain outlet temperature [14].

For example, if the condensate water level is lower than desired, then hot steam can approach the condensate drain. This will heat the condensate back up and potentially flash the condensate back to steam. This has negative impacts on the condensate drain subsystem. If the condensate water level is higher than desired, water injection into the turbine can occur, steam extraction flow can be restricted, and the condensing zone is restricted to interact with less of the heat exchanger area. In addition to these negative concerns, improper condensate water levels can reduce the overall heat exchange effectiveness.

![Fig. 30. Feedwater heater example.](image_url)
Typical feedwater efficiency and performance monitoring includes measurements of the water level and temperature [14]. The following defined measurements would be included in the instrumentation and control system.

1. The feedwater temperature rise is the difference between the feedwater outlet and inlet temperatures. The temperature rise will be stable if the feedwater level is also stable. This is a common design point indicator of the proper water level.

2. The terminal temperature difference is the saturation temperature of the extraction steam minus the feedwater outlet temperature. This can indicate the heat transfer performance. For example, an increase indicates a reduction of heat transfer, and a decrease indicates an increase of heat transfer. The design goal is typically 3~5°F.

3. The drain cooler approach temperature is the temperature difference between the drain cooler outlet and the feedwater inlet. This suggests that there are condensate levels present in the feedwater heater. For example, an increase in the drain cooler approach temperature will indicate that the level is decreasing, while a decrease in the temperature will indicate that the level is increasing. The design goal is typically 10°F.

In the PRISM reference design [12], each turbine has multiple stages of steam extraction for feedwater heating (Fig. 31). The high-pressure turbine has a single extraction nozzle for high-pressure steam FH (Fig. 32). The low-pressure turbine has four extraction nozzles for low-pressure steam FH (Fig. 32). The low pressure (LP) turbines provide extraction steam to the two trains of LP regenerative feedwater heaters (heater [HTR] 1 – HTR 4). The high pressure (HP) turbine shaft seal leakoff is directed to heater HTR 4. The feedwater drain system directs condensed turbine extraction steam that was used for regenerative FH to the condenser hotwell (Fig. 33).

The feedwater flow into the steam drum, feedwater heating, and other properties are part of the steam generator drum control system. The PRISM design has a total of three feedwater systems and three FH drain systems.

The PRISM feedwater heating instrumentation and control [12] includes:

1. Adjusting the different turbine steam extraction control valves to maintain the proper heating efficiency as the feedwater flow is varied.
   a. The extraction lines to feedwater heaters HTR 3 – HTR 5 have motor-operated valves for automatic shutoff on an extreme high level in the feedwater heater to prevent backflow to the turbine. Immediately downstream of each motor-operated valve, a fast closing bleeder trip valve (non-return valve) is used to limit turbine overspeed due to entrained energy in the extraction system. This valve affords protection from a water induction standpoint. The bleeder trip valves are normally closed by heater high water level or turbine trip signals.
   b. In the condensate system, a 5°F terminal temperature difference and a 10°F drain cooler approach temperature in the low-pressure feedwater heaters are the proper operating conditions.
   c. In the feedwater system, a 5°F terminal temperature difference and a 10°F drain cooler approach temperature in the high-pressure feedwater heaters are the proper operating conditions.
   d. A steam generator inlet temperature of 420°F should be provided to avoid thermal shock.

2. The deaerator steam supply controls activate on turbine trip or on a signal from the turbine load control system to prevent feedwater pump cavitation as a result of rapid load reductions or a turbine trip (Fig. 34).

3. The steam dump system is controlled by the turbine bypass system controls to permit steam dump to the condenser when the condenser is available.
4. The extraction steam isolation valve controls permit manual operation of each extraction steam isolation valve from the main control room, and they provide automatic valve closing in the event of an extremely high water level in the feedwater heater.

5. The extraction steam bleeder trip valve controls permit local testing of each extraction line bleeder trip valve, and they provide power assist closing in the event of a turbine trip or an extremely high water level in an associated heater.

6. The molten salt reactor team flow valve controls modulate steam flow to each reheater for gradual heat-up to protect the reheater and LP turbines from rapid temperature transients. It also permits manual operation of the steam flow valve from the control center.

7. The FH drain is used to regulate the FH vessel shell side water level. The heater drains are controlled with a series of control valves (Fig. 35).

A future project will develop the proper subsystems for the FH and the associated instrumentation and control features. Developing a Modelica model of the different low-pressure and high-pressure feedwater heaters will require development of the following subsystem models:

1. high and low-pressure turbine steam extraction connections from the turbine model, to include
   a. nozzles and flow passages with proper geometries and flow properties, and
   b. control valves and fast acting bleeder valves to regulate and bypass the turbine steam extraction;

2. feedwater heater heat exchangers, to include
   a. proper geometry heat exchangers for the various feedwater heaters,
   b. proper plumbing configurations for the phased heat exchanges,
   c. proper geometry, flow properties, and control valves for the feedwater drain systems, and
   d. exchanger condensate-level monitoring and temperature monitoring; and

3. feedwater pumping, to include appropriate feedwater pump configurations and control.
Fig. 31. PRISM diagram for one turbine-generator system [Fig. 10.1-1 in Ref. 12].
Fig. 32. PRISM diagram for extraction steam system flow [Fig. 10.3-2 in Ref. 12].
Fig. 33. PRISM diagram for condensate system flow [Fig. 10.A-2 in Ref. 12].
Fig. 34. PRISM diagram for feedwater system flow [Fig. 10.A-3 in Ref. 12].
Fig. 35. PRISM diagram for heater drains system flow [Fig. 10.A-4 in Ref. 12].
4. PRELIMINARY VALIDATION AND VERIFICATION STRATEGY

Simulation models are never identical to the operation of real-world systems but instead represent approximations. V&V of computer simulation models is conducted during the development of a simulation model with the ultimate goal of producing an accurate and credible model. For the purposes of confirming models, V&V have specific definitions. Verification is the process of confirming that models are correctly implemented with respect to the conceptual model. Validation is the process of checking the simulation against data or other benchmarks that represent the real system [6]. Development of models that have undergone V&V is critical in nuclear reactor design analysis. Because of the critical importance of reactor safety systems, considerable effort is made to develop codes that meet rigorous standards for quality assurance. The level of effort necessary to produce a model that has passed V&V depends on the model’s intended use. Some models are intended to provide scoping studies of potential design spaces. For these, the development of full V&V as stipulated in requirements such as NQA-1 is not necessary. The Modelica models that have been produced previously and described in Refs. 3 and 5 are all considered operational models with no expectation of being used for accident simulation. A model should be V&V to the degree needed for its intended purpose or application. Each system model should be based on referenced preliminary concept models that are used to benchmark the Modelica models. For these purposes, the benchmark and/or calibration of the Modelica models against existing models or test data without extensive documentation is considered acceptable.

This report represents the first attempt to implement models designed specifically for accident analysis (GRSAC code for HTGRs). This use case introduces a greater range of expected plant response and a consequently larger range of expected uncertainty. However, these models are still considered preliminary, as they are not part of any licensing calculations. As such, the appropriate level of V&V is still a benchmark or calibration of the Modelica models against the existing models. In this case, the basis models are developed in Fortran. The transition of Fortran-based models into Modelica is an important part of any development of system-based models for nuclear applications; since over 50 years of modeling work in nuclear systems analysis has been performed principally in Fortran. The ability to rapidly assimilate these models and validate and verify them is essential for creating a new paradigm of reactor systems modeling. The basic development of models within the Modelica framework is being accomplished using a standard development architecture and environment. Below is a brief tailored description of this standard code development workflow architecture as described in Ref. 7.

4.1 DEVELOPMENT ARCHITECTURE/ENVIRONMENT

The discussion within this section is based extensively on that provided in Ref. 7.

In software development, an environment is the computer system in which a computer program or software component is deployed and executed. This environment may be consistent with the user’s environment. Typically, if software is developed for use by nonprogrammers, the user and development environments are distinctly different. Changes to software are developed in the development environment. The developer’s environment typically includes tools such as a compiler, an integrated development environment, different or additional versions of libraries and support software, etc. These tools are not present in a user’s environment, but they are useful for initial modification development, testing, and revision control.

To ensure revision control, particularly with multiple developers, a developer has a working copy of source code on his or her machine, and changes are submitted to the repository, being committed either to the trunk or a branch, depending on development methodology. The environment on an individual workstation, where changes are worked on and tried out, may be referred to as the local environment or
a sandbox. Building the repository’s copy of the source code in a clean environment is a separate step and is part of integration (integrating disparate changes). This environment may be called the integration environment or the development environment. In continuous integration this is done frequently, often for every revision. The source code level concept of committing a change to the repository, followed by building the trunk or branch, corresponds to pushing to release from the local, individual developer’s environment to integration.

Environments may vary significantly in size: the development environment is typically an individual developer’s workstation, while the production environment may be a network of many geographically distributed machines in data centers, or virtual machines in cloud computing. Code, data, and configuration may be deployed in parallel.

Exact definitions and boundaries between environments vary. The testing environment may be considered part of or separate from the development environment, whereas the quality assurance (QA) environment may be considered part of the testing environment, or it may be separate. The main tiers (or branches) are progressed through in order, with new releases being deployed (rolled out or pushed) to each in turn. For the purposes of this effort, these separate environments are specified in Table 2. A description of their implementation in the GitHub repository library of models follows in the next section.

<table>
<thead>
<tr>
<th>Environment/tier name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local (sandbox)</td>
<td>Developer’s desktop/workstation. This includes developers distributed around the country under many separate computer systems and architectures.</td>
</tr>
<tr>
<td>Development</td>
<td>This is the lowest level shared environment for developing and collaborating on system/component model development.</td>
</tr>
<tr>
<td>Test/QA</td>
<td>Testing includes functional, performance, and quality assurance testing, etc.</td>
</tr>
<tr>
<td>Production/live</td>
<td>Upon completion of QA and functional testing, the models are moved to the production environment for distribution and use by the user community.</td>
</tr>
</tbody>
</table>

4.2 MODEL DEVELOPMENT AND PRODUCTION OVERVIEW

An effective collaboration environment requires structure and rules for development and modification of shared models. The overall V&V strategy described in Sect. 5.0 requires implementation in the GitHub repository that serves as the development and testing environment. Strategies developed in other software disciplines can be used to ensure this effective collaboration. The concept of workflow through the system is described and detailed in the subsections below based on the special needs of each staging area. The intended end results are production models that can be shared among other applications, including web-based simulations that are available without special simulation software.

GitHub supports a distributed version control environment that is flexible for collaboration while also providing structure for development, testing, verification, and deployment. GitHub support for the various modeling environments and workflows is documented below using GitHub Flow and graphics from the GitHub website (https://guides.github.com/introduction/flow/index.html). As described on this website, GitHub Flow is a “lightweight, branch-based workflow that supports teams and projects where deployments are made regularly.”
4.2.1 Sandbox Area

GitHub employs a distributed version control approach, so every user has a version of the repository on his or her local machine that serves as a sandbox area. The sandbox area(s) are various cloned areas from the GitHub repository copied onto each developer’s local machine. These areas are not intended to be shared, but they constitute preliminary development of models. There are no requirements for configuration control or procedures for development in the sandbox area. It is an unrestricted area for early model development. The sandbox area can still be under version control in the user’s cloned repository, which is isolated from the main repository and not accessible in general to other users.

4.2.2 Development Branch

A common approach to collaborative development is based on issue tracking and tickets (Figs. 36–37). GitHub has excellent support for issue tracking (https://guides.github.com/features/issues/). Issues can represent new model development or modification to existing models. Issues can be reported, discussed/commented, categorized, prioritized, assigned to milestones, assigned to developers, and act as the starting point for development.

Fig. 36. Sample issues from a GitHub repository.
When working on a given issue, the developer should create a branch (see Fig. 38) in his or her repository and name it according to the issue number (e.g., “issue25”). Until the developer’s changes have been approved by the project manager, they remain in their repository. Nevertheless, they can still collaborate with other developers.

Active collaboration and communication between developers occurs in this branch (see Fig. 39). Code modification is tracked through the use of commits and Git push/pull protocols for developmental changes on the branch and isolated from the main line of development. Individual users provide improvements to modeling code or libraries and subroutines accessed and used in the simulations.
Although not yet established, primary responsibility for various models is expected to be established via assigning particular developers and team members to an issue and the subsequent branch. This approach does not prevent others from collaborating or commenting on the models, but it identifies the individual who coordinates the collaboration and provides the necessary documentation or decision-making authority.

### 4.2.3 Merging, Testing, and QA

Once the developer considers an issue to be resolved, he or she can submit a pull request (Fig. 40). This signals to the project manager that the developer feels the changes are complete and ready to be incorporated into the main line of development.

At this point, the project manager reviews the changes. This review may involve inspecting the code changes to ensure that style guidelines were followed and that the code looks correct. At that point, any established testing procedures should be implemented to ensure that the models behave as expected and that there are no regressions in existing tests.

Once a pull request has been opened, the person or team reviewing changes may have questions or comments (see Fig. 41). Models that fail testing can undergo continued development on the branch until adequate performance is reached.

Testing includes both functional and performance tests, as well as quality assurance tests. The primary use of the QA/testing phase is to test all system and component models before they are applied to production environment. This ensures that all selectable configurations in the production environment will be completed reliably without errors and in minimum time. Performance testing, particularly load testing, is also important to ensure the system and component models can be run in a reasonable period of time. QA and functional tests are expected to be designated to either an overall integrator for the models (Oak Ridge National Laboratory [ORNL]), or the tests are assigned to the most knowledgeable developers under direction for final approval by ORNL.
4.2.4 Production Deployment

Deploying to production is the most sensitive step. Once the pull request has been reviewed and the branch passes testing, changes can be deployed to verify them in production, including other changes made to the main line of development. If the branch causes issues, changes can be rolled back, and the existing master can be deployed to production. Once changes have been verified in production, the code can be merged into the master branch (see Figs. 42–43). Once merged, pull requests preserve a record of the historical changes to the code, and they allow any developer to understand how and why changes were made in a completely searchable way.
With the exception of the web and the functional mockup interface for Excel (FMIE) applications, the models themselves are not remote executables but rather constitute compiled simulations that can be run via access to a Modelica solver. Currently the solver of choice is Dymola, as it retains many user-friendly features and simulation libraries that enhance the simulation and execution of the models. Developers and users will need access to a Dymola license to execute the simulations in the native form. However, the production environment will also include the functional mockup units (FMUs) that allow for co-simulation in other platforms. Therefore, except for the web application, the movement of the QA-tested, approved models into production constitutes deployment. For the web application, minor modifications associated with the database used for user choice selections will be needed to deploy the tested and approved models through the web application. This may require restart of the web application, but it will not result in any significant unavailability of the system. Users will be notified of any interruption associated with this movement or any other interruption associated with system maintenance.
4.3 MODEL RELEASE AND ACCESS

Upon the completion of testing and movement of tested models into the production environment, release notification to the development user base (GitHub registered and approved users) will be made by the development integrator (ORNL). The Modelica models and any FMUs used for co-simulation modeling and/or web application simulations will be deployed, and the web application will be updated to reflect these new model choices. If released models are proprietary, information for permissions and/or request for access will be included in the notification.

5. CONCLUSIONS

An initial HTGR architecture and preliminary example for a core Modelica model have been developed as a starting point for the full development of an end-to-end system model. Additionally, an example for wrapping existing Fortran based HTGR models developed as part of the GRSAC code has been presented. HTGR architectures will include implementation of power production and process heat application. Core and RCCS models are new, with the existing heat exchanger and steam generator models expected to be modified to account for change in flows, geometries, and fluids. The balance-of-plant models will also be tailored from the existing library of ALMR and FHR models, with the exception of the process heat application. A new model(s) for process heat applications will be developed for use with the HTGR, as well as other non-power production nuclear applications. These are expected to include things such as seawater desalination, hydrogen production, district heating, tertiary oil recovery, and other industrial applications.

Development has continued for the ALMR end-to-end control system. The remaining balance-of-plant control systems include the main steam turbine valve and the feedwater heater. Strategies, approaches, and example control systems have been developed for further refinement and implementation in the repository.

A strategy for model V&V has been developed that makes use of the GitHub repository by the creation of separate branches for the tiered workflow structure associated with model development. Initial scoping development for models will be performed locally on user/developer computers in the sandbox area. These areas are on the users'/developers’ local machine and are not shared or distributed. The developer, QA/test, and production branches have been described and reside on the GitHub server. These branches constitute shared areas for model development and collaboration. Procedures to promote models through these work areas have not yet been fully developed, but they are expected to be consistent with those developed for other software applications that make use of tiered development.

The work documented in this report represents the expected conclusion of the project under the ART program. Further funding and development of the concept and models will be sought under other DOE programs as well as the integration of these libraries and techniques in other projects.
6. REFERENCES


2. S. J. Ball to R. E. Hale, email correspondence, “Re: External Wrapper Example,” Date: 8/18/2015.


APPENDIX A.
Appendix A

Details for the HTGR models are reproduced below from Ref. 1. The initial models include the Fortran implementation of the physics described below. As described in Section 2, these initial models within Modelica include Fortran calls within Modelica. The wrappers for each of these models need to be further developed consistent with the methodology and approached described for the Modelica core model detailed in Ref. 1.
A.1 FUEL AND REFLECTOR ELEMENT CONDUCTION MODELING

A single-node representation of the temperature and the energy storage in a large hexagonal graphite block fuel element (Fig. A.1), or an array of fuel pebbles could not accurately portray the fuel-to-moderator temperature differences that exist at full-power conditions. It would also preclude approximating the at-power reactivity feedback for the neutron kinetics equations because the individual effects of fuel and moderator temperature changes are not modeled. However, for studies of shutdown power and flow scenarios, for which GRSAC is primarily intended, the radial temperature gradients within the blocks and pebble arrays are reduced to small values within a few minutes after shutdown, and the reactivity effects are no longer significant after the reactor is scrammed.

![Fig. A.1. Typical prismatic fuel block.](image-url)

The question remains, however, as to how accurately the single-node-per-element model can be used to predict the temperature transients. In general, the accuracy of any finite differencing scheme for modeling diffusion decreases as the frequency content of the perturbation increases; and for heat conduction models, the grosser the node mesh size, the more the transient heat flux between nodes is underestimated.\(^1\) In most cases, an underestimation of heat flux between adjacent elements would yield conservative (i.e., higher-than-actual) hot fuel-element temperatures.

A rough approximation of the accuracy of the one-node-per-element mesh can be derived by use of a method developed by the author to determine the ratio of approximate-to-actual slab geometry heat fluxes as a function of perturbation frequency.\(^1\) The dimensionless perturbation frequency \(\Omega\) is defined by:

\[
\Omega = \frac{(\Delta x)^2 \omega}{2D_h},
\]

where

- \(\Delta x\) = node thickness, ft;
- \(\omega\) = perturbation frequency, rad/h;
- \(D_h = k/\rho C_p\) = composite core heat diffusivity, \(\text{ft}^2/\text{h}\);
- \(k\) = conductivity, Btu/(h ft °F);
\[ \rho = \text{density, lb/ft}^3; \]
\[ C_p = \text{specific heat, Btu/(lb\cdot{}^\circ\text{F})}. \]

**Note:** at the time the original ORECA code was developed, English units were used for Fort St. Vrain (FSV) and other gas-cooled reactor calculations. Hence, most GRSAC calculations use English units.

To “translate” an accident transient into a perturbation frequency, we note that because typical modular HTGR loss-of-forced cooling and depressurization accident analyses show that peak core temperatures occur no less than 10 h after the initial failures, a complete (sinusoid) cycle would occur in <40 h so an equivalent maximum perturbation frequency would be \(~0.025\) cycle/h or \(0.16\) rad/h. Applying Fig. 2 of Ref. A.1 and using typical core properties shows that the radial heat flux between neighboring prismatic block nodes is underestimated by only a few percent at this frequency.

Another means of determining the transient accuracy of the finite-difference core conduction model is to compare model transient results with those of finer mesh approximations. Those studies also showed that for perturbation frequencies of interest, errors in temperature calculations incurred from using one node per fuel element were negligible.

The effective radial conductance between elements is accounted for by the geometric factor in the conduction equation. This is treated very simply (and arbitrarily) in GRSAC. Noting that in hexagonal geometry each node has six radial neighbors instead of four, as in slab or square-prism geometry, the equivalent slab geometric conductance term \(G_{\text{slab}}\) needs to be multiplied by \(4/6\):

\[ G_{\text{slab}} = \frac{\text{mean area}}{\text{characteristic length}} = \frac{A}{\Delta X} \approx \frac{D L}{D} = L, \]  

where for a typical prismatic fuel block, for example:

\(D\) is the distance across flats of a hexagonal element (1.18 ft),

\(L\) is the length of a block (2.6 ft).

Therefore, the heat transfer rate \(Q\) (Btu/h) between radially adjacent element blocks with the difference between mean temperatures \(\Delta T\) is determined from:

\[ Q = \frac{4}{6} G_{\text{slab}} k' \Delta T, \]  

where \(k'\) is the effective conductivity, Btu/(h ft \(^\circ\text{F}\)).

The usual form of the energy balance equation for node \(i, j\) is
\[ \rho A \Delta X \frac{dT_{ij}}{dt} = MC_v \frac{dT_{ij}}{dt} = k' \frac{\Delta X}{L} \left[ (T_{r1,j} - T_{ij}) + (T_{r2,j} - T_{ij}) + \ldots \right] \\
+ \frac{k_a A_a}{L} \left[ (T_{l1,j} - T_{ij}) + (T_{l2,j} - T_{ij}) \right] + Q_{se}, \]  

where

- \( T_{r1,j}, T_{r2,j}, \text{etc.} \) are the temperatures of its radial neighbors, °F;
- \( T_{l1,j} \) and \( T_{l2,j} \) are temperatures of its axial neighbors, °F;
- \( Q_{se} \) is the sum of the heat inputs to node \( i, j \) from internal heat generation and convection, Btu/h;
- \( M \) is the mass of the element, lb;
- \( i \) is a subscript denoting radial position;
- \( j \) is a subscript denoting axial position.

(The \( a \) subscripts refer to axial neighbors.)

The effective conductivity terms, \( k' \) and \( k_a \), are dependent on four different parameters: node average temperature, type of material (fuel or reflector), geometry and orientation (radial or axial), and irradiation history. Conductivities are calculated for each node at each time step in function routines RADK (radial) and AXIK (axial). The heat transfer between node \( i, j \) and all of its neighbors could be approximated by multiplying this value of \( k' \) by the summation of the temperature differences; however, because rather large differences in neighboring \( k' \) values can exist, this would lead to heat-balance errors due to significant differences between the calculations of heat transfer in and out. Hence, an average global effective conductivity term for each node is calculated at each time step, which accounts for its own and all its neighbors’ effective conductivities. Then the temperature differences between the node and each neighbor are multiplied by the ratio of the average \( k' \) for those two nodes to the global average. These calculations are done in subroutine ALGEN. This technique permits the use of the efficient core heat transfer solution (Sect. A.10), and it has been tested and shown to give accurate calculated heat balances for conduction-dominated transients.

An option flag (KCH) set in routines RADK and AXIK allows the user a choice of FSV, FSAR, or updated (General Atomics [GA]) MHTGR values of conductivity. The FSV expressions for conductivity, which are generally conservative (low) and do not include differences due to irradiation histories, are simple linear functions of temperature, corrected for node geometry differences.

For PBR cores, the effective conductivities are more related to radiation heat transfer than material (graphite) conductivity. Of the several available options, the one most commonly used is known as the Zehner-Schluender Robold correlation (Fig. A.2). The uncertainty ranges shown represent primarily the differences between unirradiated (top) and irradiated (bottom curve) pebbles. The spread is approximately ±25% around the mean.

The specific heat terms for the composite core and reflector elements are also included as functions of temperature and are calculated in subroutine TPROP. Thus, strictly speaking, the \( MC_v(dT/dt) \) term in Eq. (A.4) should be \( (M) \frac{dh}{dt} \) where \( h \) is the specific enthalpy. However, because \( C_p \) is defined (for small changes in temperature) as \( dh/dt \), the expression for \( C_p \), evaluated at the node temperature, can be used in Eq. (A.4). A new improved correlation for graphite \( C_p \), recommended by the International Nuclear Safety Center (INSC) is used in ORSAC, both for graphite and fuel element node-specific heat (Fig. A.3). For the derivation of the correlation, see http://www.insc.anl.gov/matprop/graphite/ent_hc/index.php.
Fig. A.2. Effective thermal conductivity ranges in the pebble bed.

The approximation to the recommended curve used in GRSAC (as converted to English units) is

\[ C_{\lambda} \left[ \text{Btu/(lb \cdot ^\circ F)} \right] = 0.288 + 0.238 \left( 1.0 - e^{(82-7) \times 10^{-6}} \right) \]

(A.5)

where \( T \) is the core node temperature, \(^\circ \text{F} \).

The uncertainty bands are \( \pm 6\% \) at the lower temperatures and \( \pm 10\% \) in the higher ranges. The function applies between \( \sim 100 \) and 2000\(^\circ \text{C} \) (\( \sim 200 \) and 3600\(^\circ \text{F} \)).
Fig. A.3. Recommended International Nuclear Safety Center (INSC) graphite heat capacities with uncertainties.
A.2 INTERNAL HEAT GENERATION CALCULATIONS

The internal heat generation, \( Q \), in the core region node \( i, j \) is an independent input function of time:

\[
Q_{ii} = Q_{a0} Q_R Q_A Q(t)
\]  \hspace{1cm} \text{(A.6)}

where

\( Q_{a0} \) is the average initial core heat generation rate,
\( Q_R \) is the radial power factor for radial position \( i \),
\( Q_A \) is the axial power factor for axial position \( j \),
\( Q(t) \) is the fraction of initial power vs time.

\( Q_A \) and \( Q_R \) values, which determine the power shaping, are input by means of data input values and are assumed constant. \( Q_{a0} \) is an input number that is determined from the overall core power density.

Treatment of the reflector blocks is similar to that of the active core block. The power fraction vs time in the side reflector blocks \( Q_{SR}(t) \) is assumed to have the same shape as \( Q(t) \) for the core.
A.3 CORE CONVECTION HEAT TRANSFER MODELING

Because the flow in the coolant channels varies over a wide range in emergency cooling situations, it is necessary to consider all three flow regimes (turbulent, transition, and laminar) and upflow as well as the normal downflow direction.

1. Turbulent \((Re > 4000)\)

The Dittus-Boelter heat transfer correlation used for prismatic cores was approximated for the turbulent regime as follows:

\[
h = (0.023)(0.88) \frac{k}{D} (Re)^{0.8},
\]

where

- \(h\) = heat transfer coefficient from gas-to-fuel-element block, Btu/(h·ft²·F);
- \((0.88)\) is approximately the 1/3 power of the Prandtl number for helium in the range of interest;
- \(k\) = conductivity of helium, Btu/(h·ft²·F);
- \(D\) = coolant channel diameter, ft;
- \(Re\) = Reynolds number, \(DG/\mu\);
- \(G\) = helium mass flow per unit area lb/(ft²·h);
- \(\mu\) = helium viscosity, lb/(ft·h).

2. Laminar \((Re < 2100)\)

For laminar flow, the average value for \(h\) over the length \(L\) of a channel was derived from Ref. A.4:

\[
h = \frac{2k}{D} \left( \frac{WC_L}{kL} \right)^{0.333},
\]

where

- \(W\) = channel flow rate, lb/h;
- \(C_p\) = helium specific heat, 1.241 Btu/(lb·°F);
- \(L\) = length of channel, ft.

3. Transition \((2100 < Re < 4000)\)

The value of \(h\) is computed as a linear function of the Reynolds number between the values of \(h\) (laminar) at \(Re = 2100\) and \(h\) (turbulent) at \(Re = 4000\).

For pebble-bed core coolant heat transfer correlations \((h_e)\), GRSAC uses a consensus function of KFA (Jülich Research Centre, Germany) and others:

\[
h_e = \frac{Nu}{k_e} / d_p,
\]

where

- \(h_e\) = heat transfer coefficient,
- \(Nu\) = Nusselt number.
\[ k = 0.09 + 7.67 \times 10^{-5} T, \]
\[ \mu = 0.054 + 4.125 \times 10^{-5} T, \]

where instead of the average helium temperature, the adjacent block temperature \( T (\^\circ F) \) is used as an approximation.

Sensitivity studies have shown that the safety significance of the results is not sensitive to the expected range of uncertainties in the heat transfer correlations, including the Reynolds number flow regime transition points.

The calculation of the heat exchanged between a solid node and a coolant gas can be approximated in a variety of ways. When the solid is represented as a point mass at temperature \( T_s \) (assumed to be uniform over the node), the heat transferred from solid to coolant \( Q_{sw} \) is often calculated by:

\[ Q_{sw} = h_d (T_s - T_c). \]

An arithmetic average coolant temperature \( T_c \) can be used, for example:

\[ T_c = (T_{ci} + T_{co}) / 2, \]

where \( T_{ci} \) and \( T_{co} \) are the coolant inlet and outlet temperatures, and \( T_{co} \) is determined from:

\[ Q_{sw} = W C_p (T_{co} - T_c). \]

However, this approach may seriously overestimate the amount of heat transfer and give values of \( T_o \) greater than \( T_s \) (when the gas is being heated), especially at low flow rates. It can also result in a “wrong-way” response to rapid changes of the inlet coolant temperature. To avoid these (nonphysical) situations, the value of the quantity \((h_d / W C_p)\) for the nodal approximation must be \(< 2.0\). Because this often cannot be achieved for very low dimensionless characteristic length flows, the end point weighting (EPW) or “well-mixed” approximation is sometimes used such as is assumed in the GA RECA code.\(^1\) In the EPW
approximation, the mean temperature of the coolant is assumed equal to the outlet temperature (i.e., \( T = T_{\infty} \)). This avoids both the overestimation of the heat transferred and the wrong-way response problem, but in the general case of slug flow, it underestimates the heat transfer rates over the entire flow range. Also, for initial full-power conditions, it overestimates the stored energy and peak fuel temperatures in the core.

The model for heat exchange from the coolant to the adjacent solid node at temperature \( T_s \) used in the GRSAC is known as the “exponential approach” method,

where

\[
T_{\infty} = T_s + (T_s - T_\infty) \left(1 - e^{-k/\alpha C_p}\right).
\]  

(A.15)

![Diagram of heat transfer approximations](image)

**Fig. A.4.** Comparison of coolant heat transfer approximations.

This model gives an “exact” solution for the heat transfer rate for the case where the solid temperature is assumed to be uniform over the entire length of the node, the coolant transit time is negligible, and the physical properties are constant. It also avoids the wrong-way response problem. A comparison of the steady-state values of percent approach vs \( \alpha A/W C_p \) for the arithmetic mean EPW and exponential approach models is shown in Fig. A.4.

The percent approach concept commonly used in heat exchanger design characterization is defined as:
A.4 CORE FLOW RATE EQUATIONS

Unlike the Fort St. Vrain reactor and later designs of large HTGRs, modular HTGR fuel-element flows (and pebble bed flow regions) cannot be adjusted by core inlet orifice valves. Hence, the flow distributions are governed by the temperature-dependent flow resistance, which in turn depends on element or nodal peaking factors, and in the case of the pebble bed, by local pebble packing fractions. Over the operating power and flow range, the element-by-element flows are approximately proportional to the total core flow rate. However, with temporary or long-term flow stoppages and with low flows typical of emergency cooling situations, the region flow distributions become quite sensitive to temperature effects, buoyancy forces, and other factors and are thus very important in determining maximum fuel temperatures. Hence, one must solve for all of the element or nodal flows simultaneously to determine any one of them.

A.4.1 Prismatic Cores:

The flow equation in GRSAC for prismatic cores is very similar to its counterpart in ORECA. It is a one-dimensional momentum equation for incompressible flow in a channel and is applicable to all cases of interest except during periods of very rapid depressurization.

\[ \Delta P = W_i^2 \left( \frac{R}{g \cdot \Delta P} \right) \left[ K_i + \sum_{j=1}^{N_i} \frac{T_{ij} - T_{ij}^*}{T_{ij}^*} + \frac{2fL}{D} \right] \frac{gLP}{gR \sum_{j=1}^{N_i} \frac{1}{T_{ij}^*}}. \]  
(A.17)

where

- \( \Delta P \) = core plenum-to-plenum pressure drop, lb/ft²;
- \( W_i \) = channel i flow rate, lb/s;
- \( R \) = gas constant for helium, 386 ft³/(lb·°R·lbₘ);
- \( g \) = acceleration due to gravity, 32.2 ft/s²;
- \( g_c \) = conversion factor, 32.2 ft³/(lb·°R·lbₘ);
- \( A_i \) = fuel-element cross-section area, ft²;
- \( P \) = average channel pressure, lb/s²;
- \( K_i \) = lumped resistance coefficient for inlet flow distribution and other restrictions;
- \( T_{P_i} \) = inlet plenum temperature, °R;
- \( i \) = index of axial element;
- \( N_i \) = number of axial elements;
- \( T_{ij} \) = outlet temperature, element \( i_j \), °R;
- \( T_{ij}^* \) = inlet temperature, element \( i_j \), °R;
- \( f \) = Fanning friction factor;
- \( L \) = axial element length, ft;
- \( D \) = mean hydraulic diameter, ft;
- \( T_{ij}^* \) = average temperature, element \( i_j \), °R.

The temperature difference terms (\( T_{ij} \) and \( T_{ij}^* \)) account for the losses due to acceleration, the friction factor \( f \) is a function of flow regime, and the summation term on the right-hand side is the buoyancy or static head term.

The friction factor \( f \) in the turbulent region (Re = 4291) is approximated by:
\[ f = 0.0014 + 0.125 \text{Re}^{-0.32}, \quad \text{(A.18)} \]

and in the laminar region (\( \text{Re} < 1600 \)) by:

\[ f = \frac{16}{\text{Re}}. \quad \text{(A.19)} \]

In between these two regions, \( f \) is assumed to be constant at 0.01. Sensitivity studies to determine the effects of assuming higher friction factors and splitting the entrance and exit loss terms (rather than lumping them at the inlet) show that in general, these considerations have little effect on the maximum predicted fuel temperatures.

GRSAC assumes that the total core flow \( \sum W_i \text{ is specified as an input function and then uses an iterative scheme to find the “correct” overall core } \Delta P \text{ to satisfy the total flow conditions, except for the case where natural convection flow (in air ingress accidents) occurs.} \]

### A.4.2 Pressure Drop in a Pebble Bed Reactor Core

Operational experience in the Arbeitsgemeinschaft Versuchsreaktor (AVR), THTR, and other experiments provides the following empirical relation for PBR core pressure drop (\( \Delta P \)). It is a slight variation of the Ergun equation for pressure drop \( \Delta P \) in a bed of packed spheres. (The pressure drop is very sensitive to void fraction changes.)

\[ \Delta P = \Delta \rho = \psi \left( 1 - \varepsilon \right) \frac{1}{\varepsilon} \frac{H}{d_p} \frac{1}{2 \rho} \left( \frac{W}{A} \right)^2, \quad \text{(A.20)} \]

where

\[ \psi = \frac{320}{\text{Re}} + \frac{6}{\left( \frac{\text{Re}}{1 - \varepsilon} \right)^{0.1}}. \quad \text{(A.21)} \]

The parameters for the relationship above are defined as:

- Reynolds number: \( \text{Re} = \frac{W \cdot d_p}{\eta A} \),

where

- \( H \) = core height (m);
- \( A \) = cross section of the core (m²);
- \( d_p \) = pebble diameter (m);
- \( \varepsilon \) = pebble bed void fraction;
- \( \rho \) = density of helium (kg/m³);
- \( \eta \) = dynamic viscosity of the helium (kg/[m·s]);
- \( W \) = helium mass flow (kg/s).
Furthermore, the range of applicability is specified by the following:

\[ 1 \leq \frac{Re}{1 - \varepsilon} \leq 10^3, \]
\[ 0.36 \leq \varepsilon \leq 0.42. \]

### A.5 Plenum Models with Radiation Heat Transfer

While detailed models were used in the original ORECA models, simplifications were introduced (and tested) in the MORECA modeling of radiation heat transfer between the upper and lower core surfaces and the vessel thermal shields. Rather than calculating radiation heat transfer between each of the fuel-element and reflector upper surface nodes, and the individual nodal surfaces of the vessel thermal shields, a concentric-ring approximation was used. Each ring represents a single ring of elements with a ring of corresponding projected area of the thermal shield directly opposite. Hence, seven rings are used to represent each surface. Equations for ring-to-ring heat transfer were derived from view factor equations for opposing disks given in Ref. A.7.

The view factor \( F_{12} \) for opposing disks 1 and 2 with radii \( R_1 \) and \( R_2 \) separated by length \( L \) is:

\[
F_{12} = 0.5 \left( x - \sqrt{x^2 - 4 \left( \frac{R_2}{R_1} \right)^2} \right), \tag{A.22}
\]

where

\[
x = 1 \pm \frac{1 + R_2^2}{R_1^2},
\]

\[
R_1 = \frac{r_1}{L},
\]

\[
R_2 = \frac{r_2}{L}.
\]

The view factors for concentric disks to rings can be obtained by subtracting out the doughnut centers from disk-to-disk view factors. For example, to solve for the view factor for ring 2 to disk 3, \( F_{23} \) (see Fig. A.5):

\[
A_1 F_{13} + A_2 F_{23} = A_{i+2} F_{(i+2)3}, \tag{A.23}
\]

\[
F_{23} = \frac{A_{i+2} F_{(i+2)3} - A_i F_{13}}{A_2}, \tag{A.24}
\]

where

\[
A_1, A_2 = \text{disk areas},
\]

\[
A_2 = \text{ring area}.
\]

Likewise, ring-to-ring view factors can be calculated by subtracting out the area-weighted view factor of a ring to an inner disk from the ring to the outer disk.

Radiant heat transfer from the upper core surface to the sidewalls was also found to be significant. The view factor for each ring to the sidewalls is simply calculated by noting that the sum of the view factors for any ring should be 1.0 so the difference between 1.0 and the sum of its view factors to the opposing rings is its sidewall view factor.
Fig. A.5. Ring-to-disk view factors.

View factor calculations are done in subroutine VFRING and employed in subroutine TOPTEM (upper plenum) and BOTTEM (lower plenum).
A.6 CORE BARREL AND VESSEL TEMPERATURE MODELING

The nodalization scheme used for the core barrel and vessel accounts for azimuthal asymmetries by splitting the fuel and reflector sections into quadrants. Axial noding allocates one each to the upper and lower reflector areas and one for every two of the ten fuel-element sections. Hence, in the fuel-reflector region, there are 4 quadrants times 7 axial sections or 28 nodes each for the core barrel and vessel. In each of the upper and lower plenums, there is one core barrel node and one vessel node for the side walls. The upper plenum ceiling and lower plenum floor are each represented by seven concentric ring nodes as described in the preceding section. The top of the vessel is represented by a single node. The bottom of the lower plenum floor is assumed to be well-insulated with heat transfer to the lower vessel head neglected.

Insulated thermal shields can be utilized in various regions to protect the vessel from over-temperature; however, the placement of the shields and their thicknesses are design considerations that must account for the fact that the RCCS’ heat removal effectiveness in a heatup accident requires high vessel temperatures. Insulation design must also consider that for pressurized heatup accidents, the maximum temperatures occur near the top wall for depressurized scenarios, they are near the vessel midplate.

In an example GRSAC model, for instance, the insulation inside the FSV vessel top head is assumed to consist of a thin thermal shield plate plus 1.25 in. of Kaowool. Insulation in the upper plenum sidewall area and in the region adjacent to the upper reflectors is assumed to consist of a shield plate plus 0.75 in. of Kaowool. Radiation shield plates (without Kaowool) are assumed to be used in the lower plenum sidewall region.

The calculation of heat transfer through radiation shields with conduction through insulation would normally involve iterations needed to determine the intermediate shield temperature. Instead, a straightforward explicit approximation was developed that gives good accuracy in the temperature ranges of interest. Equivalent heat transfer coefficients \( h \) (for assumed emissivities of 0.8 for the core barrel, shield and vessel surfaces along with unity view factors) are simple functions of the hot surface temperature \( T \) and the difference \( \Delta T \) between the hot and cold surfaces. Using the conductivity expression for Kaowool as:

\[
k = 0.1507 + T(1.349E-4 + 3.496E-8T), \tag{A.25}
\]

the approximate \( h \)'s for two different example insulation thicknesses are

\[
h_{25\text{in}} = 0.14 + 0.00231 T = 0.0014 \Delta T \tag{A.26}
\]

and

\[
h_{0.75\text{in}} = 0.00375 T - 0.0023 \Delta T, \tag{A.27}
\]

where

- \( T \) = temperature, °F,
- \( k \) = conductivity, Btu/(hr·ft²·°F),
- \( h \) = heat transfer coefficient, Btu/(hr·ft²·°F).

In another example, the model for heat transfer between the core barrel and the vessel in the core region, the “view” between the two is \(-50\%\), obscured by the rectangular inlet coolant ducts. Because a
full radiation shield would cut the heat transfer rate by half, it is assumed that with 50% of the view obscure, the heat transfer rate is reduced by 25%.

Core barrel-to-vessel heat transfer calculations are made in subroutines TOPEM for the upper plenum region, BOTTEM for the lower plenum, and CONVEC for the middle vessel regions.

In response to a review critique, heat conduction between vessel nodes was added to the model. This was found to have negligible effects on computed vessel node temperatures (~1°F maximum) during core heatup accident scenarios. The calculations, which use temperature-dependent steel conductivities, are done in subroutine VESCON.

### A.7 PRIMARY SYSTEM PRESSURE MODELING

Changes in primary system pressure are calculated by accounting for changes in primary system gas temperatures and inventories. The estimated pressure changes are strongly dependent on bulk gas temperatures in the power conversion unit and in relatively “dead” spaces that together could account for up to ~75% of the total mass of the primary system gas. Hence, the pressure calculated in core heatup transients is an “approximate” method and depends strongly on the details of gas turbine, intermediate heat exchanger (IHX), or steam generator cooldown operations. Those operations may be crucial to the outcome of pressurized core heatup accidents because some (unlikely) scenarios might lead to pressures exceeding the relief valves’ setpoint.

The primary system pressure calculation (as a function of temperature) is approximated in function PRESS by dividing the gas volume into four regions and solving the perfect gas law equation. For a given initial pressure $P_0$ and volume absolute temperatures ($T_1$–$T_4$ for volumes $V_1$–$V_4$), the constant $RMT$ for a fixed inventory is defined as:

$$RMT = P_0 \left( \frac{V_1}{T_1} + \frac{V_2}{T_2} + \frac{V_3}{T_3} + \frac{V_4}{T_4} \right), \quad (A.28)$$

Subsequently, the primary pressure $P$ is calculated from:

$$P = \frac{RMT}{\frac{V_1}{T_1} + \frac{V_2}{T_2} + \frac{V_3}{T_3} + \frac{V_4}{T_4}}. \quad (A.29)$$

Using depressurization options, the pressure $P$ can be ramped down to a new level at a specified rate, and when that target pressure is reached, $RMT$ is recalculated. Thereafter, the new value of $RMT$ corresponding to the reduced inventory is used to calculate $P$.

The four volumes are associated with the core coolant, core inlet plenum, core outlet plenum, and power conversion unit (PCU) or steam generator cavity. For very low residual primary system flows (<10%), it could be assumed that the cavity gas temperature would approach the nominal cooler or feedwater temperatures exponentially (arbitrarily assumed as a 30 min time constant); otherwise, it is computed as the average of the core inlet and outlet plenums.
A.8 SHUTDOWN COOLING SYSTEM MODELING

The shutdown cooling system (SCS) heat exchanger is typically a tube-in-shell design with pressurized water coolant in the tubes. For heat exchanger modeling, it is convenient to use dimensionless parameters and time constants for the heat transfer between each fluid and the tube.\(^1\) First, we define the “section length” \(n\) and the time constant, based on heat transfer to the surface being heated or cooled, \(\tau:\)

\[
n = \frac{hA}{WC_p}.
\]

(A.30)

and

\[
\tau = \frac{MC_p}{hc},
\]

(A.31)

where

\(h\) = fluid-to-surface heat transfer coefficient, Btu/(h ft\(^2\)°F);

\(A\) = surface area of tube, ft\(^2\);

\(WC_p\) = mass flow rate of the fluid, lb/hr, times its specific heat, Btu/(lb °F);

\(\tau\) = time constant referenced to tube, h;

\(MC_p\) = heat capacity of tube, Btu °F.

Because the time response of the SCS is fast compared to that of the core in a shutdown cooling mode, it is reasonable to employ steady-state solutions and the concept of heat exchanger effectiveness. For example, the cooling effectiveness \(\varepsilon_c\) is defined as:

\[
\varepsilon_c = \frac{T_{H0} - T_{H0}}{T_{H0} - T_{cl}},
\]

(A.32)

where if the hot helium outlet temperature \(T_{H0}\) were equal to the cooling water inlet temperature \(T_{cw}\), the device would be 100% effective \(\varepsilon_c = 1.0\). For a countercflow heat exchanger, \(\varepsilon_c\) can be calculated explicitly by:\(^6\)

\[
\varepsilon_c = \frac{1 - \exp[-(1 - N_1)N_s]}{1 - N_s \exp[-(1 - N_1)N_s]}
\]

(A.33)

In terms of the quantities defined previously,

\[
N_1 = n_c \tau_c / n_H \tau_H
\]

(A.34)

\[
N_1 = n_H / (1 + \tau_c / \tau_H)
\]

(A.35)

A heat exchanger’s heating effectiveness \(\varepsilon_h\) can be calculated in a similar fashion. These equations are solved in subroutine CAHE in that for given helium and water flows and inlet temperatures, the

(steady-state) outlet temperatures can be computed directly. The SCS model in GRSAC allows (user input) specification of the water and helium flows and the water inlet temperature. There is also a built-in automatic control function model (typically corresponding to the process controller design) that reduces the hot helium flow below the user input value if the cooling water outlet temperature exceeds 400°F (to prevent boiling). This model is in the function routine FLOW.
A.9 REACTOR CAVITY COOLING SYSTEM MODELING

For all reactor operating conditions, the reactor vessel (RV) will transfer heat by radiation and natural convection through the reactor cavity to reactor cavity cooling system (RCCS) panels (Fig. A.6), where the heated air (or water) coolant flow inside these panels is typically induced by buoyant forces (the chimney effect). The RCCS typically has no moving parts, or at least does not rely on any mechanical actions. Currently, the default model in GRSAC is for a water-cooled RCCS.

In a typical air-cooled RCCS, there are four quadrants of panels, each with an active heat transfer length of approximately 17 m. There also are four inlet/outlet structures with coaxial ducts where the inner duct carries the hotter air from the reactor cavity and the outer duct carries the cooler ambient air. The height of interconnecting ducts in a typical modular HTGR design is approximately 33 m above the panels. Redundancy is provided by interconnecting ducts and plenums to ensure that a natural convection flow of ambient air is available at all times.

The dynamic simulation of an air-cooled RCCS is described in detail in a companion report (see Ref. A.9). The GRSAC model is implemented in a subroutine RCCS. The equations governing the air flow and the air heat transfer in the RCCS are coupled. Further coupling via radiation and convection occurs by the transfer of heat from the outer surface of the RV to the outer surface of the RCCS panels. For dynamic modeling of the heat transfer process, the simplifying assumption is made that there is negligible thermal and mass inertia on the air side relative to the thermal inertia of the metal panels. The use of this “quasi-static” assumption greatly simplifies the analysis and can be rigorously justified.\textsuperscript{A.11} The same assumption is made for the primary coolant in the core.

The conservation of energy equation for each of the nine RCCS panel nodes is a simple heat balance of the heat transferred by radiation and natural convection from the vessel and the heat convected to the air flowing upward in the channel. Air flow through the RCCS ductwork, including the hot riser section of the panels, is modeled with the one-dimensional momentum equation adapted from Ref. A.2 for the core cooling channels. The outlet air temperature from each node is computed by using the exponential approach model that is an exact solution of the differential equation for conservation of energy where the panel temperature is uniform over the node length, the air transit time is negligible, and the air thermophysical properties are constant. Thermal radiation heat transfer from the front face to the sides or back of the internal hot riser channel is neglected. The convective heat transfer from the sidewalls to the flowing air is modeled as an extended surface\textsuperscript{A.11} The back face of the panel duct is treated as an adiabatic surface. The computed heat transfer to ambient conditions was found to be relatively insensitive to the value of the heat transfer coefficient on the air side of the RCCS panels.

The heat transfer process inside the reactor cavity from the uninsulated outer wall of the RV to the RCCS hot riser panels consists of natural convection and thermal radiation. Participating media thermal radiation heat transfer in the annular space between the RV and the RCCS panels is neglected in the analyses presented here but is being considered for postulated accidents in which steam or aerosols are present. The net heat transferred by radiation from the RV to the RCCS panels is modeled with the assumption that all surfaces are gray and diffuse (i.e., the emissivities are independent of wavelength). Natural convection of heat across the cavity is also modeled but is much less than the radiant heat transfer across the annulus.

For natural-convection flow analysis, the conservation of energy and momentum equations for the fluid are coupled so that simultaneous solution is usually required. However, because the dynamics of the RCCS panel are much slower than the dynamics of the air, values of the air temperatures and flows will not appreciably change over a reasonably short time step. Therefore, panel temperatures from a previous time step are used in the equation to compute air flow.
Fig. A.6. Example passive reactor cavity cooling system (air-cooled).
A.10 SPECIAL SOLUTION TECHNIQUES

To avoid the consumption-intensive task of solving all of the core heat conduction equations as a set of ~3000 coupled differential equations, a “component isolation” technique was implemented. The basis of this method is the assumption of a model in which the component (i.e., the fuel-element block) sees neighboring blocks with fixed temperatures over the time period \( \Delta t \), the computation time interval. For example, consider the coupled equations for node temperatures \( T_i \) where in hex geometry each node is coupled to six radial neighbors:

\[
\frac{dT_i}{dt} = -6\alpha T_i + \alpha (T_2 + T_3 + T_4 + T_5 + T_6 + T_7) + \frac{Q_i}{MC_p},
\]

\[
\frac{dT_2}{dt} = -6\alpha T_2 + \alpha (T_1 + T_3 + T_4 + \ldots) + \frac{Q_2}{MC_p},
\]

etc., where

\[
\alpha = \frac{D_H}{(\Delta x)^2}, \text{min}^{-1} (D_H = \text{heat diffusivity}, \Delta x = \text{spacing interval}),
\]

\[
Q_i = \text{heat generation rate in node } i, \text{ Btu/min};
\]

\[
MC_p = \text{heat capacity of the node, Btu}^\circ\text{F}.
\]

Expressed in matrix form,

\[
\frac{dT}{dt} = AT + Z.
\]  \hspace{1cm} (A.37)

The exact form of a recursive solution to Eq. (A.37), assuming \( Z \) stays constant over the time interval \( \Delta t \) is (Ref. A.12):

\[
T(t + \Delta t) = e^{\Delta t}T(t) + \left(e^{\Delta t} - I\right)A^{-1}Z_t.
\]  \hspace{1cm} (A.38)

The isolation technique incorporates the coupling to the adjacent nodes as part of the forcing function \( Z_t \);

\[
T_i(t + \Delta t) = e^{-6\alpha\Delta t}T_i(t) + \frac{e^{6\alpha\Delta t} - 1}{6\alpha} \left[\alpha(T_2 + \ldots + T_7) + \frac{Q_i}{MC_p}\right].
\]  \hspace{1cm} (A.39)

This method is similar to an Euler explicit solution; the major difference is that the first-order equations are solved exactly.
Another approximation used in GRSAC is the sequential, rather than simultaneous, solution of the conduction and convection cooling equations. The dependence of the flow equations on temperature is derived from temperatures calculated at the previous time step. This approximation is verified by reducing the computation time interval $\Delta t$ until there are no further significant changes in the results. The individual fuel column flow rates are computed at each time step by an iterative scheme that was developed by trial and error. The criteria to be satisfied are: the percentage error of the calculated total flow $\Sigma W_i$ compared with the specified total flow must be less than PERR, or the absolute error must be less than AERR (where PERR and AERR are input via DATA statements). Convergence is usually achieved within three or more iterations for reference values of PERR and AERR. The iteration scheme is as follows.

1. For the first try, if the total flow specified ($WT$) is equal to the value of $WT$ at the last time step, set the overall core $\Delta P(DP)$ equal to the last value of $DP$; if not, compute:

   \[ DP = (SDPW \cdot WT)^2 = BT, \]  

   where

   \[ SDPW = \left[ \frac{\sqrt{(DP + BT) / WT}}{\omega} \right], \]  

   $BT = \text{summation of all static head terms in Eq. (A.16),}$

   \[ BT = \frac{LP}{R} \sum_{i} \frac{1}{T_i}. \]

2. For the second try, compute:

   \[ DP = \left( \frac{WT}{\sum W_i} \right)^2 (DPL + BT) - BT, \]

   where $DPL$ is the last try value of $DP$.

   This prediction tends to overreact for fast flow transients; so to compensate for this, a lower limit value of 0.1 is used for ($WT/\Sigma W_i$). Also, if ($WT/\Sigma W_i$) = 0, then

   \[ DP = DPL + 0.7 \left[ SDPW \left( WT - \sum W_i \right) \right]^2. \]

3. For the third try and thereafter, a linear interpolation scheme is used:

   \[ DP = DP_1 + \left( \frac{WT - W_{A1}}{W_{A2} - W_{A1}} \right) (DP_2 - DP_1), \]

   where

   $DP_1, 2$ equal two previous try values of $DP$,
   $W_{A1}, 2$ equal two previous try values of $\Sigma W_i$.

The program stops if convergence is not attained in MAXIT tries ($MAXIT$ is specified in a DATA statement, typically ~20).
A.11 FUEL FAILURE MODES

Currently, GRSAC has two different fuel failure models. The first is a simple temperature-only failure-dependence model that calculates the fraction of the total fuel that has, at any time, exceeded a user-specified “failure temperature.” A second, more detailed model is based on work by D. T. Goodin of General Atomics (GA).\textsuperscript{A13} This model predicts cumulative fuel failure fractions (CFF) that are dependent on the time the fuel spends at a given temperature. The failure rate is assumed to be a function of two processes: a nonlinear mechanism due to decomposition and diffusion, and a linear mechanism due to corrosion and diffusion. Because of the nonlinear dependence of the CFF on time at a certain temperature, the original Goodin equations had to be approximated by a linear model to accommodate arbitrary fuel temperature histories. Although this model includes the effects of time at temperature, it assumes that failures are independent of fuel age or burnup. The burnup effects are included in later models, which are not as yet implemented in GRSAC.

The GRSAC implementation of the Goodin model is in subroutine GOODVT. Characterization of the nonlinear decomposition term (the B component of Goodin’s equation) by the sum of two (linear) exponentials improved the versatility of the model and allowed for decreasing temperatures. In the original model, “self healing” would occur (i.e., the fraction of failed fuel would decrease) if the fuel temperature decreased. The coefficients in the exponential approximation were determined by a gradient search routine, which found what appeared to be a global optimum set of coefficients. The resulting expression is

\[
\text{Component } B = FB \left[ C_1 \left( 1 - e^{-C_2 x} \right) + (1 - C_1) \left( 1 - e^{-C_3 x} \right) \right], \tag{A.46}
\]

where

\[
\begin{align*}
FB &= \text{Goodin’s } f_0 \text{ term for the nonlinear failure mechanism;} \\
x &= at; \\
a &= \text{Goodin’s } a \text{ term;} \\
t &= \text{incremental time, } h; \\
C_1, C_2, C_3 &= \text{coefficients in exponential approximation } (= 0.237, 32.8, 1.35).
\end{align*}
\]

The rest of the fuel failure fraction calculation (made for each fuel-element node) is taken directly from the Goodin reference.

From sensitivity studies, it was found that fuel failure calculations could use large computation time steps (many hours) with little degradation in accuracy, as long as the average of the temperature-dependent functions is representative of the true average (i.e., average values of failure rates are computed by using initial and final values of the individual component functions as opposed to using the function values computed at the average temperature over the interval).
A.12 REFERENCES


