Multiphysics Coupling of PROTEUS-NODAL and SAM for Molten Salt Reactor Simulation

Nuclear Science and Engineering Division
About Argonne National Laboratory
Argonne is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC under contract DE-AC02-06CH11357. The Laboratory’s main facility is outside Chicago, at 9700 South Cass Avenue, Argonne, Illinois 60439. For information about Argonne and its pioneering science and technology programs, see www.anl.gov.

DOCUMENT AVAILABILITY

Online Access: U.S. Department of Energy (DOE) reports produced after 1991 and a growing number of pre-1991 documents are available free via DOE’s SciTech Connect (http://www.osti.gov/scitech/)

Reports not in digital format may be purchased by the public from the National Technical Information Service (NTIS):
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312
www.ntis.gov
Phone: (800) 553-NTIS (6847) or (703) 605-6000
Fax: (703) 605-6900
Email: orders@ntis.gov

Reports not in digital format are available to DOE and DOE contractors from the Office of Scientific and Technical Information (OSTI):
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062
www.osti.gov
Phone: (865) 576-8401
Fax: (865) 576-5728

Disclaimer
This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor UChicago Argonne, LLC, nor any of their employees or officers, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of document authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof, Argonne National Laboratory, or UChicago Argonne, LLC.
Multiphysics Coupling of PROTEUS-NODAL and SAM for Molten Salt Reactor Simulation

prepared by
M. K. Jaradat, G. Yang, H. Park, and W. S. Yang
Department of Nuclear Engineering and Radiological Sciences
University of Michigan

C. H. Lee and Y. S. Jung
Nuclear Science and Engineering Division
Argonne National Laboratory

February 28, 2020
EXECUTIVE SUMMARY

In order to enhance the accuracy and efficiency of the PROTEUS-NODAL code for MSR simulation, several new capabilities have been implemented in the variational nodal $P_1$ solver and the PROTEUS-NODAL code has been coupled with the system analysis module SAM. First of all, to eliminate the error caused by approximating a cylindrical MSR core by a hexagonal core, a variational nodal $P_1$ method was developed for cylindrical geometries. The developed capability was verified against the FDM solver of the DIF3D code using 2D and 3D steady state problems derived from the MSFR benchmark. Relative to the fine hexagonal geometry model that yields a comparable accuracy, the computational time was reduced about 100 times by an R-Z model and about 7 to 8 times by an R-$\theta$-Z model. Although the computational gain depends on the specific problem, it is obvious that a cylindrical geometry model is more efficient and accurate than a hexagonal geometry model for cylindrical geometry problems.

The transient analysis capability that had been limited to the $S_P^3$ solver in hexagonal geometry was extended to the $P_1$ solvers for Cartesian, triangular, hexagonal, and cylindrical geometries. The TFSP solver was extended to the $P_1$ solvers, and a FDM solver for the delayed neutron precursor equation was developed. To improve the computational efficiency for transient analyses, a CMFD acceleration scheme was also implemented. A capability to calculate the kinetics parameters in flowing fuel reactors was also added by implementing a steady state adjoint equation solver. The developed $P_1$ transient solver was verified against the $S_P^3$ transient solver that was verified last year. The test results showed that the $P_1$ solutions agree very well with the $S_P^3$ solutions. These tests also showed that the CMFD acceleration scheme reduces the computational time for TFSP a few tens of times. The $P_1$ transient solver in R-Z geometry was also tested by comparing the results with those of the hexagonal geometry option of the $P_1$ solver and those of the $S_P^3$ solver. Various unprotected transient scenarios of the MSFR benchmark problem were solved with thermal feedback, including UTOP, UPOS, ULOF, ULOHS, and UFSOC accidents. Despite the differences in geometrical models and transport approximations (diffusion vs. $S_P^3$), the three solutions showed very good agreement in the power and core-averaged fuel salt temperature. With comparable accuracies, the $P_1$ solver in R-Z geometry reduces the computational time about 10 to 60 times relative to the $S_P^3$ solver and about 5 to 10 times relative to the $P_1$ solver in hexagonal geometry.

In order to enhance the thermal-hydraulics modeling capabilities by overcoming the limitation of the standalone thermal-hydraulics solver, PROTEUS-NODAL has been coupled with SAM under the MOOSE framework. A MOOSE sub-application named TreeFrog was developed as the wrapper for PROTEUS-NODAL to communicate with other MOOSE applications. A MOOSE master application named TreeKangaroo was also developed to control the coupling calculation of TreeFrog and SAM. Different time step sizes between PROTEUS-NODAL and SAM are allowed in the coupled transient calculations by using the “subcycling” option of the MOOSE transient executioner. The Picard iteration was used in the coupled steady state calculation, and the operator-splitting method was used in the coupled transient calculations.
Verification tests of the coupled system of PROTEUS-NODAL and SAM were performed using the steady state and transient problems derived from the MSFR benchmark problem. Since the radial crossflow is neglected in the current SAM model, the effect of this simplification was first examined by comparing the steady state results with those obtained by a manually coupled calculation of PROTEUS-NODAL and ANSYS CFX. The SAM calculation used four parallel axial channels and CFX performed the full 3D CFD calculation in the cylindrical geometry of the MSFR core. Due to the neglect of the radial velocity field in the SAM calculation, SAM underestimated the axial velocity at the core center slightly, and this resulted in a slightly top-skewed power distribution: 0.1% overestimation in the upper part and 0.2% underestimation in the lower part of the core. The UTOP, ULOF, and ULOHS accidents of the MSFR transient benchmark were analyzed by including the outer loop in the SAM model. The results were compared with the PSI solutions from a coupled PARCS and TRACE calculation and the TUDelft solutions obtained from a coupled neutron diffusion and CFD calculation. In general, the power and core-averaged fuel temperature solutions of the coupled PROTEUS-NODAL and SAM calculations agreed well with the other solutions.
## REVISION HISTORY

<table>
<thead>
<tr>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0        | Release date: February 28, 2020  
Initial release |
| 1        | Release date: April 10, 2020  
Updates  
- TreeFlog/TreeKangaroo coupling with SAM 3D |
# TABLE OF CONTENTS

Executive SUMMARY .................................................................................................................... i
Table of Contents ........................................................................................................................... iv
List of Figures ................................................................................................................................ vi
List of Tables ................................................................................................................................ vii
1. Introduction ............................................................................................................................... 1
2. Development of Variational Nodal $P_1$ Solver in R-$\theta$-Z Geometry ................................. 4
   2.1. Variational Nodal Method for Diffusion Theory ............................................................. 4
   2.2. Variational Nodal $P_1$ Solver in R-$\theta$-Z Geometry ............................................................. 8
      2.2.1. Generation of Basis Functions .................................................................................. 8
      2.2.2. Generation of Response Matrices ........................................................................... 10
      2.2.3. Boundary Conditions in $\theta$ Direction ....................................................................... 11
   2.3. Verification Tests of Variational Nodal $P_1$ Solver for R-$\theta$-Z Geometry .................. 11
      2.3.1. R-Z Geometry Problem .......................................................................................... 11
      2.3.2. R-$\theta$-Z Geometry ...................................................................................................... 15
3. Development of Transient Analysis Capabilities of $P_1$ Solver ............................................... 18
   3.1. Transient Fixed Source Problem .................................................................................... 18
   3.2. Solution of Delayed Neutron Precursor Equation for Flowing Fuel .......................... 20
   3.3. Implementation of CMFD Acceleration in Variational Nodal $P_1$ Solver .................. 22
      3.3.1. CMFD Acceleration Method .................................................................................. 22
      3.3.2. Performance of CMFD Acceleration ...................................................................... 23
   3.4. Evaluation of Kinetics Parameters ................................................................................. 25
      3.4.1. Adjoint Flux and Kinetics Parameters .................................................................... 25
      3.4.2. Verification Tests of Kinetics Parameters .............................................................. 27
   3.5. Thermal Feedback Calculation ...................................................................................... 30
      3.5.1. Cross Section Functionalization ............................................................................. 30
      3.5.2. Calculation Flow ..................................................................................................... 31
   3.6. Verification Tests of $P_1$ Transient Solver .................................................................... 32
      3.6.1. Verification Tests without Thermal Feedback ....................................................... 33
      3.6.2. Verification Tests with Thermal Feedback ............................................................ 36
4. Coupling of PROTEUS-NODAL and SAM ........................................................................... 45
   4.1. Coupled Calculation Scheme ......................................................................................... 45
      4.1.1. PROTEUS-NODAL Wrapper TreeFrog ................................................................. 46
      4.1.2. Overall Computational Procedure of TreeKangaroo .............................................. 48
   4.2. Verification Test Results ................................................................................................ 52
      4.2.1. Comparison of Steady State Results of SAM with ANSYS CFX .......................... 52
      4.2.2. Transient Tests of Coupled System of PROTEUS-NODAL and SAM .......... 56
5. Summary ................................................................................................................................. 62

References ..................................................................................................................................... 64
Appendix A. Description of MSFR Benchmark Problem ............................................................. 67
Appendix B. Comparison of Three Solution Methods for Delayed Neutron Precursor Equation of Flowing Fuel .................................................................................................................... 70
Appendix C. Standard MSR Thermal-Hydraulics Solver of PROTEUS ................................. 78
Appendix D. PROTEUS-NODAL Input Data for MSR Analysis .............................................. 102
Appendix E. Input Data for TreeFrog and TreeKangaroo Applications .......................... 109
LIST OF FIGURES

Fig. 2.1. Simplified R-Z geometry of MSFR. ................................................................. 12
Fig. 2.2. MSFR core models in original cylindrical geometry (left) and in approximate hexagonal geometry of 8 rings of hexagons of 16.5 cm pitch (right). .................. 13
Fig. 2.3. Approximated hexagonal geometry core models of MSFR with 12 rings of hexagons of 11.3 cm (left) and 15 rings of hexagons of 9.0 cm pitch (right). ......................... 13
Fig. 2.4. Cross sectional view at an azimuthal angle where control rod is inserted in the simplified MSFR. ........................................................................................................ 16
Fig. 2.5. Modified core models with control rods in R-θ-Z geometry with Δθ = 13.3° for the control rod regions and 17.8° for other regions (left) and in hexagonal geometry with 15 rings of hexagons of 9.0 cm pitch (right). ................................................... 16
Fig. 3.1. Comparison of error reduction in fission source with and without CMFD for eigenvalue problems of (a) small and (b) large MSFR cores. ........................................ 24
Fig. 3.2. Comparison of (a) the computational time and (b) the number of outer iterations at each time step with and without CMFD for the modified TWIGL benchmark problem. ................................................................................................................. 24
Fig. 4.1. MOOSE MultiApps system [12]. ....................................................................... 46
Fig. 4.2. Axial nodes of PROTEUS-NODAL and axial meshes of TreeFrog. ...................... 48
Fig. 4.3. Data flow and coupling scheme between PROTEUS-NODAL and SAM under the MOOSE framework. ......................................................................................... 49
Fig. 4.4. Coupling scheme with Picard iteration for steady state calculations. .................. 50
Fig. 4.5. Coupling scheme with operator splitting for transient calculations. ..................... 51
Fig. 4.6. Axial (left) and radial (right) velocity fields of the fuel salt obtained from CFX. The cylinder axis is in the middle of both figures. ............................................................ 54
Fig. 4.7. Comparison of axial velocity profiles of SAM and CFX. ..................................... 54
Fig. 4.8. (a) Power distribution (W/cc) of coupled PROTEUS-NODAL and SAM calculation and (b) % difference from that of coupled PROTEUS-NODAL and CFX calculation. ................................................................................................................. 55
Fig. 4.9. Comparison of axial temperature profiles of SAM and CFX. ................................ 55
Fig. 4.10. SAM model of the MSFR benchmark problem. ............................................... 56
Fig. 4.11. Power evolution during the UTOP transient. ................................................... 57
Fig. 4.12. Average core temperature increase during UTOP transient. ............................. 58
Fig. 4.13. Power evolution during ULOF transient. .......................................................... 59
Fig. 4.14. Average inlet velocity evolution during ULOF transient. ................................. 60
Fig. 4.15. Power evolution during ULOHS transient. ....................................................... 60
Fig. A.1. Schematic representation of the molten salt fast reactor. ................................. 67
Fig. B.1. Illustration of characteristic lines for precursor concentration calculation. ........... 71
Fig. B.2. Fuel salt path at each time step from a starting point to a node outlet. .................. 73
Fig. B.3. Power evolution in null transient without thermal feedback. ............................... 76
Fig. B.4. Power evolution in pump start-up transient without thermal feedback. ............... 76
Fig. B.5. Power evolution in pump coast-down transient without thermal feedback. ........... 77
Fig. C.1. Radial configuration of unit cell. ........................................................................ 80
Fig. C.2. Main and momentum control volumes and state variables. ............................... 83
Fig. C.3. Mesh structure for heat conduction calculations. .............................................. 84
Fig. C.4. Schematic diagram of the heat removal from the primary system of MSFR. ......... 98
LIST OF TABLES

Table 2.1. Surface Numbering Scheme in the R-θ-Z Geometry ................................................................. 9
Table 2.2. Geometry Dependent Matrix Scaling Factors .............................................................................. 11
Table 2.3. Eigenvalue and Power Comparison Results of Unrodded Case .................................................. 14
Table 2.4. Eigenvalue and Power Comparison Results of Rodded Case ..................................................... 17
Table 3.1. Performance of CMFD for Eigenvalue Problems .......................................................................... 23
Table 3.2. Calculated Safety Parameters for MSFR Benchmark Problem .................................................. 28
Table 3.3. Eigenvalues of Steady State MSFR Benchmark Problem .......................................................... 37
Table 3.4. Performance Comparison of Three Solvers for MSFR Transient Problems .............................. 44
Table A.1. Characteristics of the Molten Salt Fast Reactor ......................................................................... 68
Table A.2. Initial Composition of the Fuel Salt of MSFR .......................................................................... 68
Table A.3. TRU-element Fuel Composition ............................................................................................... 68
Table A.4. Thermo-physical Properties of LiF-ThF4 Fuel Salt ................................................................... 69
Table A.5. Decay Heat Parameters of MSFR ............................................................................................ 69
Table D.1. Input Cards in Driver Input File ................................................................................................. 102
Table D.2. Keywords Added to Geometry Input for R-θ-Z Geometry ....................................................... 103
Table D.3. Input Cards in Material Assignment File .................................................................................. 103
Table D.4. Input Cards in Standalone Thermal-Hydraulics Input File ......................................................... 104
Table E.1. Input Description of Mesh Block ............................................................................................... 109
Table E.2. Input Description of AuxVariables Block .................................................................................. 110
Table E.3. Input Description of Function Block ........................................................................................ 110
Table E.4. Input Description for Problem Block in TreeKangaroo ............................................................ 111
Table E.5. Input Description for Problem Block in TreeFrog .................................................................. 112
Table E.6. Input Description for Steady Executioner Block ...................................................................... 113
Table E.7. Input Description for SteadyWithPicardCheck Executioner Block ......................................... 113
Table E.8. Input Description for Transient Executioner Block .................................................................. 113
Table E.9. Input Description for MultiApps Block .................................................................................... 114
Table E.10. Input Description for Transfers Block ....................................................................................... 115
Table E.11. Input Description for Outputs Block with Short-cut Syntax .................................................... 117
Table E.12. Input Description for Outputs Block with Sub-block Syntax ................................................. 117
1. Introduction

The molten salt reactor (MSR) with the fuel dissolved into the liquid salt has been selected as one of the six Generation-IV reactors due to its excellent characteristics in the aspects of sustainability, economy, passive safety, and resource utilization [1-3]. In a liquid fuel MSR, the fuel salt is circulated throughout the primary system and thus some of the delayed neutron precursors generated in the core decay in positions of low importance or even out of the core. The MSR can be configured in both thermal and fast spectrum reactors, depending on the use of a moderator. Thermal spectrum MSRs such as the Molten Salt Reactor Experiment (MSRE) and the Molten Salt Breeder Reactor (MSBR) design use the graphite moderator and the fuel flows through the channels in the graphite moderator while fast spectrum MSRs such as the Molten Salt Fast Reactor (MSFR) design have no moderator.

Recently both thermal and fast spectrum MSRs are gaining increasing interest from the industry. This increases the need for advanced design analysis capabilities for MSR applications. In order to meet this need for MSR modeling and simulation capabilities, it was decided to extend the capabilities of the PROTEUS-NODAL code [4,5] being developed at Argonne National Laboratory for the steady state and transient analyses of MSRs. PROTEUS-NODAL is a three-dimensional (3D) neutron transport code based on a homogeneous assembly model for various nuclear reactor applications. The code has two solution options: the $P_1$ solver based on the DIF3DVARIANT methodology [6] and the simplified $P_3$ (SP$_3$) solver based on the triangle-based polynomial expansion nodal (TPEN) method for the radial direction and the one-dimensional nodal expansion method (NEM) for the axial direction [7]. The $P_1$ solver is applicable to Cartesian, triangular, and hexagonal geometry problems, while the SP$_3$ solver is applicable to hexagonal geometry problems only.

Last year, both the $P_1$ and SP$_3$ solvers were extended to model MSRs by taking into account the precursor drift in a flowing fuel [8-10]. However, the transient analysis capability was limited to the SP$_3$ solver for hexagonal geometry problems. As a result, cylindrical cores of molten salt fast reactors had to be approximated by hexagonal geometry models. In addition, the standalone thermal-hydraulics solver implemented in the PROTEUS-NODAL code was based on a parallel one-dimensional (1D) multi-channel model for the core and simple lumped parameter models for the components outside of the core [10]. In order to improve the analysis accuracy by eliminating the unnecessary modeling approximations, it was decided to extend the $P_1$ solver to cylindrixal geometry problems and to couple the PROTEUS-NODAL code with the System Analysis Module (SAM) [11], which is a modern system analysis tool being developed at Argonne National Laboratory for advanced non-LWR safety analyses.
The system analysis code SAM aims to provide fast-running, whole-plant transient analysis capability with improved-fidelity for sodium-cooled fast reactors, lead-cooled fast reactors, and molten salt reactors or fluoride-cooled high-temperature reactors. SAM utilizes the Multi-physics Object-Oriented Simulation Environment (MOOSE) framework and its underlying meshing and finite-element library (libMesh) and linear and non-linear solvers (PETSc), to leverage the modern advanced software environments and numerical methods [12]. The MOOSE framework has been developed and maintained at Idaho National Laboratory under the NEAMS integration product line. The MOOSE framework is originally designed for fully coupled implicit multi-physics simulations based on finite element and the Jacobian-free Newton-Krylov (JFNK) methods in a quick and efficient manner. For loosely coupled systems or simulations involving external codes such as PROTEUS-NODAL, the MOOSE framework uses the operator splitting method to solve the coupled system of equations.

To apply the P1 solver to cylindrical geometry problems, a new P1 solver for R-Z and R-θ-Z geometries has been developed based on the variational nodal method. The P1 solver for cylindrical geometry problems has been verified against the DIF3D finite difference code [13]. In addition, transient solvers have been developed for all the P1 solvers using the transient fixed source problem (TFSP) formulation. The computational efficiency has also been improved significantly for transient problems by implementing the coarse mesh finite difference (CMFD) acceleration scheme into all the P1 solvers for Cartesian, triangular, hexagonal, and annular geometry problems. The steady state and transient P1 and SP3 solvers for MSR applications have been coupled with the SAM code under the MOOSE framework. The coupled system of PROTEUS-NODAL and SAM has been verified by comparing the MSFR benchmark solutions with the solutions found in the open literature and the PROTEUS-NODAL solutions obtained with the standalone thermal-hydraulics solver.

The purpose of this report is to document the new features and capabilities that have been implemented in the PROTEUS-NODAL code and the coupling procedure of PROTEUS-NODAL with SAM to analyze MSRs. This report is organized as follows. Section 2 presents the variational nodal P1 solver for cylindrical geometries and its verification test results. Section 3 discusses the transient analysis capability implemented in the P1 solvers of PROTEUS-NODAL. The solution methods for the coupled system of TFSP and precursor drift equations, the coarse mesh finite difference (CMFD) acceleration, the kinetics parameter evaluation method, and the thermal feedback are discussed. The verification test results against the SP3 solutions are also discussed. Section 4 describes the coupling scheme of PROTEUS-NODAL and SAM under the MOOSE framework for MSR applications. The developed MOOSE applications named “TreeFrog” and “TreeKangaroo”
are discussed. Verification test results for the coupled system are also discussed by comparing the MSFR steady state solution with that obtained from a manually coupled PROTEUS-NODAL and ANSYS CFX [14] computational fluid dynamics (CFD) calculations and the MSFR transient analysis results with the open literature solutions and the PROTEUS-NODAL solutions obtained with the standalone thermal-hydraulics solver. Section 5 provides conclusions and future works.
2. Development of Variational Nodal P₁ Solver in R-θ-Z Geometry

The original implementation of the P₁ and SP3 solvers of PROTEUS-NODAL has no solution option for cylindrical geometry problems. The P₁ solver is applicable to Cartesian, hexagonal-Z, and triangular-Z geometries, and the SP3 solver is only for the hexagonal-Z geometry. As a result, the MSFR benchmark problem in cylindrical geometry has been analyzed using approximate hexagonal geometry models [9, 10]. In order to eliminate the unnecessary modeling approximation, a variation nodal P₁ solver for R-θ-Z geometry has been developed. This section describes the variational nodal method in R-θ-Z geometry and its implementation in PROTEUS-NODAL. The verification test results against the finite difference method (FDM) solutions of DIF3D [13] are also discussed along with the computational gain achieved with the cylindrical geometry model compared to the hexagonal geometry model of the MSFR.

The general variational nodal method for the diffusion theory is first described. This is followed by the discussion on the implemented R-θ-Z geometry solver, including the basis functions and the resulting response matrices. Then, the verification test results against the FDM solver of DIF3D are presented.

2.1. Variational Nodal Method for Diffusion Theory

In the variational nodal method for the diffusion theory, the functional for an internal node \( v \) can be written as follows.

\[
F_v[\phi_g, \chi_g] = \int_v dV \left\{ D_g(r) \nabla \phi_g(r) \cdot \nabla \phi_g(r) + \Sigma_{r,g}(r) \left( \phi_g(r) \right)^2 - 2 \phi_g(r) S_g(r) \right\} + 2 \sum_{\gamma} \int_{\Gamma_{\gamma}} d\Omega \left( \nabla \cdot \hat{n}_\gamma \right) \phi_g(r) \chi_{r,g}(r, \Omega),
\]

where \( V \) is the node volume, \( \Gamma_{\gamma} \) is the area of surface \( \gamma \), \( g \) is the energy group index, \( \phi_g(r) \) is the zeroth moment of the even parity flux (i.e., the scalar flux), \( \chi_{r,g}(r, \Omega) \) is the first moment of the odd parity flux on surface \( \gamma \), and \( S_g(r) \) is the isotropic source. Carrying out the angular integration in Eq. (2.1) yields

\[
F_v[\phi_g, \chi_g] = \int_v dV \left\{ D_g(r) \nabla \phi_g(r) \cdot \nabla \phi_g(r) + \Sigma_{r,g}(r) \left( \phi_g(r) \right)^2 - 2 \phi_g(r) S_g(r) \right\} + 2 \sum_{\gamma} \int_{\Gamma_{\gamma}} d\Gamma_r \phi_g(r) j_{r,g}(r),
\]
where \( j_{r,g}(r) \) is the net surface current normal to the surface \( \gamma \). By expanding the scalar flux and the surface current with orthogonal spatial basis functions, \( f_i(r) \) and \( h_{r,j}(r) \), respectively, they can be expressed as

\[
\phi_g(r) = \sum_{i=1}^{N_v} f_i(r) \xi_{g,i},
\]
\[
j_{r,g}(r) = \sum_{j=1}^{N_s} h_{r,j}(r) \chi_{r,g,j},
\]

where \( \xi_{g,i} \)'s are the expansion coefficients of \( N_v \) volumetric spatial basis functions \( f_i(r) \) and \( \chi_{r,g,j} \)'s are the \( N_s \) expansion coefficients of surface spatial basis functions \( h_{r,j}(r) \). The source is expanded with \( f_i(r) \) and can be expressed as

\[
S_g(r) = \sum_i f_i(r) s_{g,i}.
\]

Assuming constant cross sections in each node and using Eq. (2.3) and Eq. (2.4), Eq. (2.2) can be rewritten with the summation convention rule as

\[
F_v[\phi_g, \chi_g] = D_g \int_v dV \nabla_x f_i(r) \xi_{g,i} \nabla_x f_j(r) \xi_{g,j} + \sum_{r,g} \int_v dV f_i(r) \xi_{g,i} f_j(r) \xi_{g,j} + \sum_{r,g} \int_{\Gamma} f_i(r) \xi_{g,i} h_{r,j}(r) \chi_{r,g,j} - 2 \int_v dV f_i(r) \xi_{g,i} s_{g,j} + 2 \int_{\Gamma} f_i(r) \xi_{g,i} h_{r,j}(r) \chi_{r,g,j},
\]

where \( \alpha \) sums over \( x, y, \) and \( z \) for Cartesian, hexagonal, and triangular-z geometries and over \( r, \theta, \) and \( z \) for cylindrical geometries. Using matrix and vector notations, Eq. (2.5) can be expressed as

\[
F_v[\phi_g, \chi_g] = \xi_g^T (D_g \bar{P} + \sum_{r,g} \bar{F}) \xi_g - 2 \xi_g^T \bar{F} \xi_g + 2 \sum_{r,g} \xi_g^T \bar{M}_\gamma \chi_{r,g},
\]

where \( \xi_g = [\xi_{g,1}, \ldots, \xi_{g,N_v}]^T \), \( \chi_{r,g} = [\chi_{r,g,1}, \ldots, \chi_{r,g,N_s}]^T \), and \( s_g = [s_{g,1}, \ldots, s_{g,N_v}]^T \). The elements at the \( i \)-th row and the \( j \)-th column of the matrices \( \bar{P}, \bar{F}, \) and \( \bar{M}_\gamma \) are given by

\[
\bar{P}_{ij} = \sum_{\alpha} \int_v dV \nabla_x f_i(r) \nabla_x f_j(r),
\]
\[
\bar{F}_{ij} = \int_v dV f_i(r) f_j(r) = V \delta_{ij},
\]
\[
\bar{M}_{r,ij} = \int_{\Gamma} d\Gamma f_i(r) h_{r,j}(r).
\]
The spatial basis functions are orthogonalized with the Gram-Schmidt orthogonalization procedure so that \( \tilde{F}_g = V_v \delta_{ij} \) \cite{15}. As a result, the first element of the coefficient vector of the scalar flux becomes the node-averaged scalar flux yielding
\[
\zeta_{g,i} = \frac{1}{V_v} \int dV \phi_g(r) f_i(r) = \frac{1}{V_v} \int dV \phi_g(r) = \overline{\phi}_g .
\] (2.8)

Requiring the reduced functional in Eq. (2.6) be stationary with respect to the variation of the scalar flux results in
\[
\zeta_g = \tilde{\mathbf{A}}^{-1}_g \left( \tilde{\mathbf{F}} s_g - \sum_j \tilde{\mathbf{M}}_{j,g}^T \chi_j \right) ,
\] (2.9)

where \( \tilde{\mathbf{A}}_g = D_g \tilde{\mathbf{P}} + \sum_{r,g} \tilde{\mathbf{F}} \). In order to use a nodal response matrix approach, an additional relation that couples interfacial scalar fluxes (or interfacial partial currents) and node average scalar fluxes between nodes is needed. This relation can be obtained by requiring the surface current to be continuous across each interface. To represent the surface scalar flux \( \phi_{g,j}(r) = \phi_g(r \in \Gamma_j) \) in terms of the node average scalar flux, \( \phi_{g,j}(r) \) can be projected onto each surface basis function as
\[
\int_{\gamma} d\Gamma_j h_{j,j}(r) \phi_{g,j}(r) = \int_{\gamma} d\Gamma_j h_{j,j}(r) \phi_{g}(r \in \Gamma_j) .
\] (2.10)

By orthogonalizing the surface basis functions such that \( \int_{\gamma} d\Gamma_j h_{j,j}(r) h_{j,j}(r) = \Gamma_j \delta_{ij} \) and by expanding the surface scalar flux with the orthogonalized surface basis functions as \( \phi_{g,j}(r) = \sum_j h_{j,j}(r) \phi_{g,j} \), the following relation is obtained.
\[
\Gamma_j \phi_{g,j} = \sum_j \int_{\gamma} d\Gamma_j h_{j,j}(r) f_j(r) \zeta_{g,j} = \tilde{\mathbf{M}}_{g,j}^T \zeta_g ,
\] (2.11)

where \( \phi_{g,j} = [\phi_{g,j,1}, \ldots, \phi_{g,j,N_s}]^T \) is the coefficient vector of the surface flux. Because of the orthogonality of the surface basis functions, the first element of the coefficient vector of the surface current becomes the surface-averaged current as
\[
\chi_{g,1} = \frac{1}{\Gamma_j} \int_{\gamma} d\Gamma_j h_{1,j}(r) f_1(r) = \frac{1}{\Gamma_j} \int_{\gamma} d\Gamma_j \phi_{g,j}(r) = \overline{f}_{g,j} .
\] (2.12)

By inserting Eq. (2.9) into Eq. (2.11), the surface flux is expressed as
\[
\phi_{g,j} = \tilde{\mathbf{M}}_{g,j}^T \tilde{\mathbf{A}}^{-1}_g \tilde{\mathbf{F}} s_g - \tilde{\mathbf{M}}_{g,j}^T \tilde{\mathbf{A}}^{-1}_g \sum_j \tilde{\mathbf{M}}_{j,g}^T \chi_{j,g} ,
\] (2.13)
where $\tilde{M}_r^T = \tilde{M}_r^T / \Gamma_r$. In the diffusion approximation, $\varphi_{r,g} = 2\left(\chi_{r,g}^+ + \chi_{r,g}^-\right)$ where $\chi_{r,g}^\pm = [\chi_{r,g,1}^\pm, \ldots, \chi_{r,g,N_s}^\pm]^T$ are the surface partial currents. Therefore, Eq. (2.13) can be rewritten in terms of partial currents as

$$
\chi_{r,g}^+ + \chi_{r,g}^- = \frac{1}{2} \tilde{M}_r^T \tilde{A}_g^{-1} \tilde{F}_g - \frac{1}{2} \tilde{M}_r^T \tilde{A}_g^{-1} \sum_{r'} \tilde{M}_{r'} \left(\chi_{r',g}^+ - \chi_{r',g}^-\right).
$$

(2.14)

Eq. (2.9) and Eq. (2.14) can be rewritten in the following response matrix equations:

$$
\zeta_g = \tilde{H}_s s_g - \overline{C}_g \left(J_g^+ - J_g^-\right),
$$

(2.15)

$$
J_g^+ = \tilde{B}_g s_g + \overline{R}_g J_g^-,
$$

(2.16)

where

$$
J_g^\pm = [\chi_{g,1}^\pm, \ldots, \chi_{g,N_s}^\pm]^T,
$$

$$
\tilde{H}_s = \tilde{A}_s^{-1} \tilde{F},
$$

(2.17)

$$
\tilde{B}_g = (\tilde{G}_g + I)^{-1} \frac{1}{2} \tilde{M}_r^T \tilde{A}_g^{-1} \tilde{F},
$$

$$
\overline{R}_g = (\tilde{G}_g + I)^{-1} (\tilde{G}_g - I).
$$

The block matrices $\tilde{G}_g$, $\overline{C}_g$, and $\tilde{M}$ in Eq. (2.17) are defined as

$$
\tilde{G}_g = \begin{bmatrix}
G_{11,g} & \cdots & G_{1N_s,g} \\
\vdots & \ddots & \vdots \\
G_{N_s,1,g} & \cdots & G_{N_s,N_s,g}
\end{bmatrix},
$$

$$
\overline{C}_g = [C_{1,g}, \ldots, C_{N_s,g}],
$$

$$
\tilde{M} = [\tilde{M}_1 / \Gamma_1, \cdots, \tilde{M}_{N_r} / \Gamma_{N_r}]
$$

(2.18)

where $G_{r,g}^\pm = 1/2 \tilde{M}_r^T \tilde{A}_g^{-1} \tilde{M}_r$, $C_{r,g} = \tilde{A}_g^{-1} \tilde{M}_r$, and $N_s$ is the number of surfaces.

For a given source, Eq. (2.16) is solved iteratively. For each node, by using the incoming surface partial currents as an input, the outgoing surface partial currents are updated. Then, the outgoing surface partial currents are used as the incoming surface partial currents for adjacent nodes using the Red-Black sweeping scheme. Once the currents converge, node-average scalar fluxes are updated.
2.2. Variational Nodal $P_1$ Solver in R-$\theta$-Z Geometry

In this section, the orthogonal basis functions $f_i(r)$ and $h_{\gamma,j}(r)$ for R-$\theta$-Z geometry are first derived. Then the procedure to generate the response matrices based on these basis functions to be compatible for use with the CMFD acceleration scheme is discussed. The method to treat the periodic boundary condition to the $\theta$ direction is also discussed.

2.2.1. Generation of Basis Functions

The basis functions for the R-$\theta$-Z geometry are polynomials with a user-specified order expressed with monomials as

$$f_i(r) = \sum_p \sum_q \sum_w C_{i}^{pqw} r^p \theta^q z^w,$$
$$h_{\gamma,j}(r) = \sum_p \sum_q C_{\gamma,j}^{pq} r^p \alpha^q \beta^q, \quad \alpha, \beta = \text{coordinate variables on surface } \gamma.$$ (2.19)

The coefficients $C_{i}^{pqw}$ and $C_{\gamma,j}^{pq}$ are determined by the Gram-Schmidt orthogonalization procedure. In Cartesian, hexagonal, and triangular geometries, these coefficients are the same for all the nodes because all the nodes have the translational symmetry with arbitrary scaling. However, there is no translational symmetry in cylindrical geometry, and thus the volumetric basis functions and surface basis functions in R-$\theta$ and R-Z surfaces are unique and generated for each radial node.

For outer nodes of annular sector shape, the sequence of $p$, $q$, and $w$ and the coefficients in Eq. (2.19) are determined in the same way as the other geometry types. However, the central nodes of sector shape need a special care. At the center, the volumetric basis functions must not depend on $\theta$ and their derivatives with respect to $r$ should be zero. Thus, the following constraints should be imposed to the volumetric basis functions for central nodes.

$$\left. \frac{\partial f_i(r, \theta, z)}{\partial r} \right|_{r=0} = 0,$$
$$f_i(r, \theta, z) = g(0, z).$$ (2.20)

These are equivalent to the following constraints on the monomials of $r^p \theta^q z^w$:

1. Any monomials with $p = 1$ are not allowed.

2. For $p = 0$, only $q = 0$ is allowed.

Thus, the monomials allowed for the volumetric basis functions of the central nodes become
The same constraints are applied to the relevant surface basis functions. The allowed monomials for the surface basis functions are

Surface 1, 2: \(1, \theta, z, \theta z, z^2, \theta^2 z, z^3, \theta^3 z, \theta^2 z^2, \theta^3 z^3, \ldots \)  
Surface 3, 4: \(1, z, r^2, z^2, r^3 z, z^3, r^4, r^3 z^2, z^4, \ldots \)  
Surface 5, 6: \(1, r^2, r^3, r^4, r^3 \theta, r^2 \theta^2, \ldots \)

with the surface numbering scheme present in Table 2.1. Note that a central node has only five surfaces from Surface 2 to Surface 6.

Table 2.1. Surface Numbering Scheme in the R-\(\theta\)-Z Geometry

<table>
<thead>
<tr>
<th>Surface Number</th>
<th>Surface Type per Geometry Type</th>
<th>(R - \theta - Z)</th>
<th>(R - \theta)</th>
<th>(R - Z)</th>
<th>(R)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(\theta - z) @ -r</td>
<td>(\theta) @ -r</td>
<td>(z) @ -r</td>
<td>@ -r</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(\theta - z) @ +r</td>
<td>(\theta) @ +r</td>
<td>(z) @ +r</td>
<td>@ +r</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(r - z) @ -(\theta)</td>
<td>(r) @ -(\theta)</td>
<td>(r) @ -z</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(r - z) @ +(\theta)</td>
<td>(r) @ +(\theta)</td>
<td>(r) @ +z</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>(r - (\theta)) @ -z</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>(r - (\theta)) @ +z</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Variables \(\gamma\) represents a surface depending on “variables” at a constant \(\gamma\) with “+” for the larger value and “−” for the smaller value of variable \(\gamma\).

To keep the same number of moments for the volumetric and surface basis functions for all the nodes, the monomials in Eq. (2.22) for the central nodes can include higher order terms than the user specification. That is, the basis functions for the surfaces 3 to 6 that are constrained by Eq. (2.20) will contain monomials of higher orders.

Using the selected set of monomials, the volumetric and surface basis functions are generated with the Gram-Schmidt orthogonalization procedure. The normalization condition for the volumetric basis functions is given by

\[
\bar{F}_{ij}^{\text{code}} = \int_{\eta} d\eta \int_{-1/2}^{+1/2} d\theta \int_{-1/2}^{+1/2} dz \tilde{f}_i(r) \tilde{f}_j(r) = \delta_{ij},
\]

where \(\tilde{f}_i\)’s are different from \(f_i\)’s in Eq. (2.3) that are normalized to satisfy Eq. (2.8). The superscript \(\text{code}\) differentiates \(\bar{F}_{ij}^{\text{code}}\) from \(\bar{F}\) in Eq. (2.7). The surface basis functions for R-\(\theta\)-Z geometry are also orthonormalized as
For the other two-dimensional (2D) and one-dimensional (1D) geometries in Table 2.1, the relevant normalization conditions in Eq. (2.24) are used according to their surface types. Again, $\tilde{h}_{r,i}$’s in Eq. (2.24) are different from $h_{r,i}$’s in Eq. (2.3) that are normalized to satisfy the condition $\int d\Gamma \tilde{h}_{r,i} (r) h_{r,i} (r) = \Gamma \delta_{ij}$.

The Gram-Schmidt orthogonalization procedure does not produce accurate results for large $r$ values because of increasing round off errors. Therefore, the quadruple precision was used in the orthogonalization procedure by defining separate routines that deal with 16-byte real variables.

### 2.2.2. Generation of Response Matrices

Once the basis functions are generated, the response matrices can be generated using Eq. (2.7), Eq. (2.17), and Eq. (2.18). In order to comply with the CMFD acceleration, it is better to carry the scalar fluxes and surface currents that satisfy the normalization conditions in Eq. (2.8) and Eq. (2.12). For this, the $\tilde{P}_{\text{code}}$ and $\tilde{M}_{\gamma \text{code}}$ matrices resulting from the basis functions generated with Eq. (2.23) and Eq. (2.24), are multiplied by the scaling factors in Table 2.2 to satisfy the normalization conditions in Eq. (2.8) and Eq. (2.12). Using these scaled $\tilde{F}$, $\tilde{P}$, and $\tilde{M}_{\gamma}$ matrices, the final response matrices are generated using Eq. (2.17) and Eq. (2.18).
### 2.2.2. Boundary Conditions in $\theta$ Direction

The allowed boundary conditions in the $\theta$ direction are the periodic, vacuum and extrapolated ones. The periodic boundary condition was realized by assigning the same surface numbers to the surfaces at the both ends in the $\theta$ direction. In this way, the interface partial currents can be mapped between the nodes at the both ends in the $\theta$ direction. Even though the reflective boundary condition can also be realized with a simple mapping of interface partial currents for the diffusion approximation, it is not the case when the transport theory is considered. Thus, to avoid the loss of generality, the reflective boundary condition to the $\theta$ direction is not allowed.

### 2.3. Verification Tests of Variational Nodal $P_1$ Solver for R-Ω-Z Geometry

Using the MSFR benchmark problem of which description is given in Appendix A, the developed variational nodal $P_1$ solver for R-Ω-Z geometry problems was verified against the FDM solver of the DIF3D code. First, the $P_1$ solver of PROTEUS-NODAL for R-Z geometry was verified by solving the MSFR benchmark problem for a steady state without the dependency on the azimuthal direction. In the second problem, the $P_1$ solver for R-Ω-Z geometry was verified by solving a modified MSFR problem with three control rods located 120° apart from each other. The computational gain of using the cylindrical geometry compared to the hexagonal geometry was also examined by solving the same problems with the $P_1$ solver in hexagonal geometry.

#### 2.3.1. R-Z Geometry Problem

The steady state MSFR benchmark problem was solved using the R-Z geometry shown in Fig. 2.1. The radial configuration at the core level is shown in Fig. 2.2. The left geometry of Fig. 2.2 was used in the $P_1$ solver for R-Z geometry and DIF3D-FDM. The radial node size...
was ~15 cm, and the axial node size was 10 cm in the upper and lower reflectors and 11.275 cm for the core. Two DIF3D-FDM calculations were performed: one with 1 cm × 1 cm mesh size and the other one with 2 cm × 2 cm mesh size. The reference solution was obtained with the Richardson extrapolation of the two results.

The approximate hexagonal geometry problems in the right figure of Fig. 2.2 and in Fig. 2.3 were also solved using the P1 solver in hexagonal geometry. In these models, the pitch was determined to preserve the volume of the fuel region at the core center with a given number of rings. The rest regions were constructed to preserve the volume of each region as close as possible. The model of 9 cm-pitch hexagons was prepared as the smallest hexagon case in order to estimate the computational gain of the R-Z model relative to the hexagonal model of comparable accuracy. In all the calculations, a 60° periodic boundary condition was used.

The 233U-started composition was used in the calculation. The MC²-3 and the TWODANT codes were used to generate an ISOTXS file with the 33-group structure. The same ISOTXS file was used both in DIF3D-FDM and PROTEUS-NODAL.

![Fig. 2.1. Simplified R-Z geometry of MSFR.](image)
Fig. 2.2. MSFR core models in original cylindrical geometry (left) and in approximate hexagonal geometry of 8 rings of hexagons of 16.5 cm pitch (right).

Fig. 2.3. Approximated hexagonal geometry core models of MSFR with 12 rings of hexagons of 11.3 cm (left) and 15 rings of hexagons of 9.0 cm pitch (right).
The zero incoming current boundary condition \((\alpha = 0.5 \text{ in } \alpha = J/\phi)\) was used for all directions: \(A = 0.5\) in the type 05 card in NIP3 for the DIF3D calculation and the VACUUM boundary condition for the PROTEUS-NODAL calculation. For the convergence criteria, \(10^{-7}\) and \(10^{-5}\) were used for the eigenvalue and the fission source convergence, respectively. No acceleration techniques for fission source convergence were applied both in DIF3D-FDM and PROTEUS-NODAL for consistent comparison.

The calculation results are summarized in Table 2.3. The \(\% \Delta P\) column in the table means the relative error in the average power densities in the central core region nodes of \(~15 \text{ cm} \times 11.275 \text{ size}\). To calculate the power distribution error in the hexagonal geometry, the power density at the center of each hexagon was regarded as the point quantity and the radial average values in a ring of interest were obtained with the weighting function \(r\) in the integration range of the inner and the outer radial boundaries of the ring of interest. Thus, the error represents the discretization error of approximating a ring with hexagons.

### Table 2.3. Eigenvalue and Power Comparison Results of Unrodded Case

<table>
<thead>
<tr>
<th></th>
<th>K-eff (error)</th>
<th>% (\Delta P) RMS, MAX</th>
<th>Time (s)</th>
<th># of Outer Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDM (2 cm (\times) 2 cm(^b))</td>
<td>0.98375 (+8 pcm)</td>
<td>0.38, 1.69</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td>FDM (1 cm (\times) 1 cm)</td>
<td>0.98369 (+2 pcm)</td>
<td>0.026, 0.097</td>
<td>824</td>
<td>97</td>
</tr>
<tr>
<td>Extrapolated (Reference)</td>
<td>0.98367</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>R-Z Nodal (~15 cm)</td>
<td>0.98366 (-1 pcm)</td>
<td>0.0087, 0.051</td>
<td>1.5</td>
<td>49</td>
</tr>
<tr>
<td>Hexagonal Nodal (9.0 cm)</td>
<td>0.98365 (-2 pcm)</td>
<td>0.55, 1.47</td>
<td>161.8</td>
<td>44</td>
</tr>
<tr>
<td>Hexagonal Nodal (11.3 cm)</td>
<td>0.98339 (-28 pcm)</td>
<td>1.12, 3.16</td>
<td>91.0</td>
<td>44</td>
</tr>
<tr>
<td>Hexagonal Nodal (16.5 cm)</td>
<td>0.98278 (-89 pcm)</td>
<td>2.59, 7.31</td>
<td>40.6</td>
<td>44</td>
</tr>
</tbody>
</table>

\(^a\) All nodal calculations used 10 cm and 11.275 cm axial node sizes for the axial reflectors and core regions and the sixth and linear expansion orders for volumetric and surface basis functions, respectively.

\(^b\) \(\Delta r \times \Delta z\)

It can be seen that the eigenvalue and power distribution results of the \(P_1\) solver for R-Z geometry agree very well with the reference solutions obtained by Richardson extrapolation of two DIF3D-FDM solutions. When the hexagonal geometry model was used to approximate the cylindrical geometry, a comparable accuracy to the R-Z model solution was obtained with a 9 cm hexagonal pitch. The eigenvalue of this hexagonal geometry model agrees very well with the reference value, but the power distribution error is comparable to the DIF3D-FDM solution of 2 cm meshes. It can be seen that for a comparable computational...
accuracy, the R-Z model reduces the computational time about 100 times relative to the hexagonal geometry model.

2.3.2. R-θ-Z Geometry

To verify the P_1 solver for R-θ-Z geometry, three virtual control rods were introduced 120° apart from each other as shown in Fig. 2.4 and Fig. 2.5. The control rods are about ~10 cm wide, and the volumes are the same in the R-θ-Z geometry and the hexagonal geometry. To introduce the control rods, the third and the fourth rings in the left figure of Fig. 2.2 were merged and subdivided into three rings where the control rods were inserted in the middle ring. The radial node sizes of all the other nodes and the axial node sizes of all nodes were kept the same as in the unrodded case. Two calculations were performed with the R-θ-Z geometry by varying node sizes in the azimuthal direction. One used 13.3° for the nodes containing a control rod and 17.8° for other nodes, and the other used the half-sized nodes. For the hexagonal geometry, only the 9 cm-pitch hexagonal model was used in the calculation. In all the PROTEUS-NODAL calculations, the 120° periodic boundary condition was used. For the DIF3D-FDM calculations, radial and axial meshes were kept the same as in the unrodded case and 120° was divided into 80 and 160 azimuthal meshes. Then, the reference solution was obtained with Richardson extrapolation of the two results. Again, no acceleration scheme for fission source iteration was not used in the PROTEUS-NODAL and DIF3D-FDM calculations.

Table 2.4 shows that the P_1 solution in R-θ-Z geometry agrees very well with the DIF3D-FDM reference result. The use of the azimuthal node size of 17.8° turned out to be sufficiently fine for this problem. As seen in the unrodded case, the P_1 solution in hexagonal geometry with 9 cm hexagonal pitch agrees very well with the reference solution. For a comparable accuracy, the R-θ-Z geometry model reduces the computational time by a factor of 7.8 relative to the hexagonal geometry model. Although the computational gain depends on the specific problem, it is obvious that an R-θ-Z geometry model is more efficient and accurate than a hexagonal geometry model for cylindrical geometry problems.
Fig. 2.4. Cross sectional view at an azimuthal angle where control rod is inserted in the simplified MSFR.

Fig. 2.5. Modified core models with control rods in R-θ-Z geometry with Δθ = 13.3° for the control rod regions and 17.8° for other regions (left) and in hexagonal geometry with 15 rings of hexagons of 9.0 cm pitch (right).
Table 2.4. Eigenvalue and Power Comparison Results of Rodded Case

<table>
<thead>
<tr>
<th></th>
<th>K-eff (error)</th>
<th>% ΔP&lt;sup&gt;a&lt;/sup&gt; RMS, MAX</th>
<th>Time (s)</th>
<th># of Outer Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDM (1.48° × 2 cm × 2 cm)&lt;sup&gt;b&lt;/sup&gt;</td>
<td>0.96329 (+26 pcm)</td>
<td>0.15, 0.80</td>
<td>665.8</td>
<td>78</td>
</tr>
<tr>
<td>FDM (0.74° × 1 cm × 1 cm)</td>
<td>0.96310 (+7 pcm)</td>
<td>0.039, 0.20</td>
<td>6763.9</td>
<td>101</td>
</tr>
<tr>
<td>Extrapolated (Reference)</td>
<td>0.96303</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>R-Z Nodal (8.9° 10 ~ 15 cm)</td>
<td>0.96304 (+1 pcm)</td>
<td>0.019, 0.11</td>
<td>15.3</td>
<td>41</td>
</tr>
<tr>
<td>R-Z Nodal (17.8°, 10 ~ 15 cm)</td>
<td>0.96306 (+3 pcm)</td>
<td>0.039, 0.17</td>
<td>9.9</td>
<td>46</td>
</tr>
<tr>
<td>Hexagonal Nodal (9.0 cm)</td>
<td>0.96306 (+3 pcm)</td>
<td>77.9</td>
<td>42</td>
<td></td>
</tr>
</tbody>
</table>

* All nodal calculations used 10 cm and 11.275 cm axial node sizes for the axial reflectors and core regions and the sixth and linear expansion orders for volumetric and surface basis functions, respectively.

<sup>a</sup> Averaged over a sector (Δθ = 17.8°) of a ring in the fuel region with 11.275 cm in the axial direction and 10 ~ 15 cm in the radial direction.

<sup>b</sup> Δθ × Δr × Δz
3. Development of Transient Analysis Capabilities of P1 Solver

In the original PROTEUS-NODAL code, the transient analysis capability was limited to the SP3 solver in hexagonal geometry. In order to perform transient analyses for reactors in Cartesian (e.g., MSRE) and cylindrical (e.g., MSFR) geometries directly without geometrical approximations, the transient analysis capabilities for stationary and flowing fuels have been added to the variational nodal P1 solver that can now solve problems in Cartesian, hexagonal-Z, triangular-Z, and R-θ-Z geometries. To improve the computational efficiency for transient analyses, a CMFD acceleration scheme was also implemented. The developed P1 transient solver was verified against the SP3 transient solver verified in the last year work using various transient problems in hexagonal geometry. The P1 transient solver in R-Z geometry was also tested by comparing the results with those of the hexagonal geometry option of the P1 solver and those of the SP3 solver.

The method to solve the TFSP for stationary fuel is first discussed. This is followed by the discussion on the modified precursor concentration equation for flowing fuel and its solution scheme. The CMFD acceleration method for transient analyses, the kinetics parameter evaluation, the thermal feedback model, and the coupling scheme of the neutronics and thermal-hydraulics calculations are described in the subsequent subsections. Then, verification test results of the implemented transient capabilities are presented. The descriptions of added input data to PROTEUS-NODAL are provided in Appendix D along with sample input files.

3.1. Transient Fixed Source Problem

Using the conventional notations, the time-dependent neutron diffusion equation in the multigroup form with the delayed neutron precursor concentration equations for stationary fuel can be written as

\[
\frac{1}{v_g} \frac{\partial}{\partial t} \phi_g(\vec{r},t) - \nabla \cdot D_g(\vec{r},t) \nabla \phi_g(\vec{r},t) + \Sigma_{rg}(\vec{r},t) \phi_g(\vec{r},t) = q_g(\vec{r},t), \tag{3.1}
\]

\[
\frac{\partial}{\partial t} C_k(\vec{r},t) + \lambda_k C_k(\vec{r},t) = \lambda \psi_k(\vec{r},t) \quad \text{for } k = 1, 2, \ldots, K, \tag{3.2}
\]

where
\[
q_g(\vec{r}, t) = \sum_{g' \neq g}^G \Sigma_{ngg'}^{g'}(\vec{r}, t) \phi_g^{g'}(\vec{r}, t)
+ \lambda \chi_{pg}(\vec{r}, t) \sum_{g'=1}^G \nu_g \Sigma_{fgg'}^{g'}(\vec{r}, t) \phi_g^{g'}(\vec{r}, t) + \sum_{k=1}^K \chi_{dkg}(\vec{r}, t) \lambda_k C_k(\vec{r}, t). \tag{3.3}
\]

In Eq. (3.3), the delayed neutron precursor generation rate for delayed neutron precursor group \(k\) is given by
\[
\psi_k(\vec{r}, t) = \sum_{g'=1}^G v_{dg} \Sigma_{fgg'}^{g'}(\vec{r}, t) \phi_g^{g'}(\vec{r}, t). \tag{3.4}
\]

For flowing fuels, Eq. (3.3) needs to be modified to include the precursor drift as discussed in the next subsection. In this subsection, the solution scheme for the TFSP is discussed for stationary fuels. Using the implicit Euler method, Eq. (3.1) can be temporarily discretized as
\[
\frac{\phi_g^n(\vec{r}) - \phi_g^{n-1}(\vec{r})}{v_g \Delta t_n} - \nabla \cdot D_g^n(\vec{r}) \nabla \phi_g^n(\vec{r}) + \Sigma_{rgg}^n(\vec{r}) \phi_g^n(\vec{r}) = q_g^n(\vec{r}). \tag{3.5}
\]

By moving the time derivative term to the right hand side, Eq. (3.5) can be written as the TFSP for given delayed neutron precursor concentrations as
\[
-\nabla \cdot D_g^n(\vec{r}) \nabla \phi_g^n(\vec{r}) + \Sigma_{rgg}^n(\vec{r}) \phi_g^n(\vec{r}) = q_g^n(\vec{r}), \tag{3.6}
\]

where
\[
q_g^n(\vec{r}) = \frac{\phi_g^{n-1}(\vec{r}) - \phi_g^n(\vec{r})}{v_g \Delta t_n} + \sum_{g' \neq g}^G \Sigma_{ngg'}^{g'}(\vec{r}) \phi_g^{g'}(\vec{r}) + \sum_{k=1}^K \chi_{dkg}(\vec{r}) \lambda_k C_k^n(\vec{r}). \tag{3.7}
\]

By approximating the precursor production rate by a quadratic function using the fission sources at previous two time nodes, Eq. (3.3) can be integrated analytically as
\[
\lambda_k C_k^n(\vec{r}) = \lambda_k e^{-\Delta t_n} C_k^{n-1}(\vec{r}) + \lambda \left[ \Omega_k^{n-2} \psi_k^{n-2}(\vec{r}) + \Omega_k^{n-1} \psi_k^{n-1}(\vec{r}) + \Omega_k^n \psi_k^n(\vec{r}) \right]. \tag{3.8}
\]

With \(r = \Delta t_{n-1} / \Delta t_n\), the three parameters in Eq. (3.8) are given by
\[ \Omega_{k}^{n-2} = \frac{1}{r_n(1 + r_n)\lambda_n \Delta t_n} \left[ 2(1 - e^{-\lambda_n \Delta t_n}) \left(1 + e^{-\lambda_n \Delta t_n}\right) \right], \]
\[ \Omega_{k}^{n-1} = \frac{1}{r_n\lambda_n \Delta t_n} \left[ 1 + e^{-\lambda_n \Delta t_n} + (1 - e^{-\lambda_n \Delta t_n}) \left( r_n - \frac{2}{\lambda_n \Delta t_n} \right) \right] - e^{-\lambda_n \Delta t_n}, \]
\[ \Omega_{k}^{n} = 1 - \frac{2}{(1 + r_n)\lambda_n \Delta t_n} + \frac{1 - e^{-\lambda_n \Delta t_n}}{(1 + r_n)\lambda_n \Delta t_n} \left( \frac{2}{\lambda_n \Delta t_n} - r_n \right). \] (3.9)

Once the group source \( q_{kn}^n(\vec{r}) \) is calculated, the solution for the group flux in Eq. (3.6) is obtained by solving the response matrix equations in Eq. (2.15) and Eq. (2.16) using the existing steady state solver. Since the within-group flux is included in the group source in Eq. (3.6), a few inner iterations are required between Eq. (2.15) and Eq. (2.16).

3.2. Solution of Delayed Neutron Precursor Equation for Flowing Fuel

For flowing fuels, the drift of delayed neutron precursors is included in the precursor balance equation as a convection term as
\[ \frac{\partial}{\partial t} C_k(\vec{r}, t) + \nabla \cdot [\bar{u}(\vec{r}, t) C_k(\vec{r}, t)] + \lambda_k C_k(\vec{r}, t) = \lambda \psi_k(\vec{r}, t). \] (3.10)

For an axial velocity field \( \bar{u}(\vec{r}) = u(r, z)\hat{e}_z \), Eq. (3.10) is reduced to
\[ \frac{\partial}{\partial t} C_k(r, z, t) + \frac{\partial}{\partial z} [u(r, z) C_k(r, z, t)] + \lambda_k C_k(r, z, t) = \lambda \psi_k(r, z, t) \] (3.11)

In this study, three different ways were investigated to solve Eq. (3.11). The first approach is the analytical scheme which solves the partial differential equation (PDE) using the method of characteristics (MOC). The second approach is the semi-analytical scheme or the so-called method of lines (MoL). The basis of this method relies on converting the partial differential equation into an ordinary differential equation (ODE) and solving it using a proper ODE solver. The third approach is the finite difference method (FDM) in which the delayed neutron precursor equation is discretized in both time and space. Test results showed that only the FDM produces the null transient result correctly. The MOC and MoL solutions diverged from the steady state solution in the opposite directions although the power deviations were only fractions of percent. Thus, only the FDM is discussed in this section. The MOC and MoL are summarized in Appendix B along with some comparison of results.

In the FDM approach, the spatial derivative is approximated with the upwind finite difference scheme, while the time derivative is approximated with the forward (explicit) or backward Euler (implicit) scheme. With the backward Euler scheme for temporal
Multiphysics Coupling of PROTEUS-NODAL and SAM for Molten Salt Reactor Simulation
February 28, 2020

discretization and the upwind finite difference scheme for spatial discretization, Eq. (3.11) becomes

\[
\frac{\bar{C}_{k,ij}^{n} - \bar{C}_{k,ij}^{n-1}}{\Delta t_n} + \frac{\bar{u}_y^{n} \bar{C}_{k,ij}^{n} - \bar{u}_{y,i-1,j}^{n} \bar{C}_{k,ij-1}^{n}}{\Delta z_j} + \lambda_k \bar{C}_{k,ij}^{n} = \lambda \psi_{k,ij}^{n}. \tag{3.12}
\]

Solving for \( \bar{C}_{k,ij}^{n} \) yields

\[
\bar{C}_{k,ij}^{n} = \left(1 + \lambda_k \Delta t_n + \frac{\bar{u}_y^{n} \Delta t_n}{\Delta z_j} \right)^{-1} \left[ \frac{\bar{u}_y^{n-1} \Delta t_n}{\Delta z_j} \bar{C}_{k,ij-1}^{n} + \lambda \Delta t_n \psi_{k,ij}^{n} \right]. \tag{3.13}
\]

The precursor concentration at the core outlet is assumed equal to the average precursor concentration of the last node in the core (upwind scheme) or it can be evaluated using the linear extrapolation of the last two nodes in the core. By assuming that precursors leaving the core are uniformly mixed and they flow back to the core except for those that decay outside the core, the boundary condition at the core inlet is evaluated as

\[
C_k(r,0,t) = \frac{\int_{A_o} dA u(r,H,t-\tau) C_k(r,H,t-\tau) e^{-\lambda \tau}}{\int_{A_o} dA u(r,0,t)} , \tag{3.14}
\]

where \( \tau \) is the time taken for precursors to pass through the core outlet and return to the core inlet.

This scheme provides an unconditionally stable solution with no restrictions on the time step size and preserves the initial delayed neutron precursor concentrations since the equation is solved in the same way for both steady state and transient analyses. Also, the main advantage of this approach is that it can be extended easily to a general velocity field in order to account for the precursor drift in the radial direction. The discretized time-dependent delayed neutron precursor equation for a general velocity field can be written as

\[
\frac{\bar{C}_{k,ij}^{n} - \bar{C}_{k,ij}^{n-1}}{\Delta t_n} + \frac{\bar{u}_{x,ij}^{n} \bar{C}_{k,ij}^{n} - \bar{u}_{x,i-1,j,l}^{n} \bar{C}_{k,ij-1,j}^{n}}{\Delta x_i} + \frac{\bar{u}_{y,ij}^{n} \bar{C}_{k,ij}^{n} - \bar{u}_{y,j-1,i,l}^{n} \bar{C}_{k,ij-j,1}^{n}}{\Delta y_j} + \frac{\bar{u}_{z,ij}^{n} \bar{C}_{k,ij}^{n} - \bar{u}_{z,j-1,i,l}^{n} \bar{C}_{k,ij-j,1}^{n}}{\Delta z_j} + \lambda_k \bar{C}_{k,ij}^{n} = \lambda \psi_{k,ij}^{n}. \tag{3.15}
\]

For each time step, Eq. (3.15) can be solved iteratively with the Gauss-Seidel method considering the axial and radial velocity fields. Note that if there is no radial velocity, Eq. (3.15) is reduced to Eq. (3.12). For a fast spectrum MSR, it would be important to consider the radial velocity field for transients in which the fuel velocity is decreasing as in the loss of flow accident scenario.
3.3. Implementation of CMFD Acceleration in Variational Nodal $P_1$ Solver

The coarse mesh finite difference (CMFD) acceleration scheme was implemented in the $P_1$ solvers to enhance the fission source convergence for the TFSP. In this section, the CMFD method for the TFSP and its performance are presented.

3.3.1. CMFD Acceleration Method

The CMFD acceleration method can be applied to the TFSP as in the eigenvalue problem. In this method, the neutron flux preserving interfacial net currents determined from a higher order calculation – the nodal calculation here – is sought for a given source from the previous time step. To preserve the net current, the current correction coefficients are obtained for all surfaces and groups as

$$\dot{D}_{g,i\rightarrow u}^n = -\frac{(J_{g,iightarrow u}^n)_{\text{nodal}} + \dot{D}_{g,i\rightarrow u}^n (\phi_{g,i}^{n,n} - \phi_{g,u}^{n,n})_{\text{nodal}}}{(\phi_{g,i}^{n,n} + \phi_{g,u}^{n,n})_{\text{nodal}}} , \tag{3.16}$$

where $\dot{D}_{g,i\rightarrow u}^n$ is the coupling coefficient between nodes $i$ and $u$ of the conventional finite difference diffusion formulation. Then, the following CMFD equations for the TFSP are constructed by moving all the terms containing quantities at the current time step to the left hand side as

$$-\sum_u A_{iu} (\dot{D}_{g,i\rightarrow u}^n + \dot{D}_{g,u\rightarrow i}^n) \phi_{g,i}^{n,n} + \sum_u A_{iu} (\dot{D}_{g,i\rightarrow u}^n - \dot{D}_{g,u\rightarrow i}^n) + \left( \sum_{r,g} + \frac{1}{v_g \Delta t_n} \right) V_i \phi_{g,i}^{n,n} , \tag{3.17}$$

where $A_{iu}$ is the area of the interface between nodes $i$ and $u$, $V_i$ is the volume of node $i$, and

$$q_{i,g}^{n,n} = \sum_{g'\neq g} \chi_{g'}^i \phi_{g'}^{i,n} + \lambda \chi_{g}\sum_{g'\neq g} V_{p} \sum_{l,g} \phi_{g'}^{i,n} + \sum_{k} \chi_{g,k} \beta_{g,k} \phi_{g,k}^{i,n} , \tag{3.18}$$

where

$$q_{i,g}^{n-1} = \sum_{k} \chi_{g,k} \beta_{g,k} \left( \lambda_k e^{-\lambda_k \Delta t_n} C_k^{i,n-1} + \lambda \left( \Omega_k^{i,n-2} \psi_k^{i,n-2} + \Omega_k^{i,n-1} \psi_k^{i,n-1} \right) \right) + \frac{1}{v_g \Delta t_n} \phi_{g,i}^{n-1} .$$

The entire linear system of equations for all energy groups is formed and the resulting fixed source problem is solved using the Krylov subspace method with no source iteration. The resulting fission source is used in the next outer iteration step for the nodal method to determine interfacial net currents.
3.3.2. Performance of CMFD Acceleration

The performance of the CMFD acceleration was examined for steady state and transient calculations. For a steady state eigenvalue calculation, small and large MSFR core problems were solved with and without the CMFD acceleration. The results in Table 3.1 and Fig. 3.1 show that the CMFD acceleration significantly reduces the number of outer iterations for both problems. However, the total computational time is almost similar to that of the calculation without the CMFD acceleration since the mesh size of the CMFD and NODAL calculations are the same and the CMFD implementation requires a longer computational time than that of the NODAL calculations.

For transient calculations, the TWIGL benchmark problem modified by converting the original Cartesian geometry to the hexagonal one [16] was solved. At each time node, a significant reduction is achieved in both the number of outer iterations and the computational time as presented in Fig. 3.2. The number of outer iterations is reduced by a factor of 20 to 60 and the computational time is reduced by a factor of 40. The reason for this large computational gain for transient calculations is due to the fact that the entire CMFD linear system is solved only once without source iterations.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Small Core</th>
<th>Large Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Hexagonal Assemblies</td>
<td>19</td>
<td>625</td>
</tr>
<tr>
<td>No. Energy Groups</td>
<td>9</td>
<td>33</td>
</tr>
<tr>
<td>w/o CMFD</td>
<td>1.000903</td>
<td>1.022056</td>
</tr>
<tr>
<td>w/ CMFD</td>
<td>0.815</td>
<td>304.55</td>
</tr>
<tr>
<td>Eigenvalue</td>
<td>56</td>
<td>11</td>
</tr>
<tr>
<td>Total Time (s)</td>
<td>0.972</td>
<td>287.15</td>
</tr>
<tr>
<td>Number of Outer Iterations</td>
<td>11</td>
<td>17</td>
</tr>
</tbody>
</table>
Fig. 3.1. Comparison of error reduction in fission source with and without CMFD for eigenvalue problems of (a) small and (b) large MSFR cores.

Fig. 3.2. Comparison of (a) the computational time and (b) the number of outer iterations at each time step with and without CMFD for the modified TWIGL benchmark problem.
3.4. Evaluation of Kinetics Parameters

In order to compare with the reported kinetic parameters of benchmark problems, the capabilities to calculate the steady state adjoint flux and the effective delayed neutron fraction and the prompt neutron generation time have been added to the PROTEUS-NODAL code.

3.4.1. Adjoint Flux and Kinetics Parameters

At the steady state, the neutron diffusion equation in Eq. (3.1) and the precursor concentration equation for a flowing fuel in Eq. (3.10) are reduced to

\[-\nabla \cdot D_r(\vec{r})\nabla \phi_g(\vec{r}) + \Sigma_f(\vec{r})\phi_g(\vec{r}) - \sum_{g'=1}^{G} \Sigma_{g'\rightarrow g}(\vec{r})\phi_{g'}(\vec{r}) = \lambda \chi_{fg} \sum_{g'=1}^{G} \nabla \cdot \phi_{g'} \phi_g(\vec{r}) + \sum_{k=1}^{K} \chi_{dkg} \lambda_k \phi_g(\vec{r})\]

\[\nabla \cdot [\vec{u}(\vec{r}) C_k(\vec{r})] + \lambda \psi_k(\vec{r}) = \lambda \psi_k(\vec{r}), \quad k = 1, 2, \ldots, K,\]  

where

\[\psi_k(\vec{r}) = \sum_{g'=1}^{G} \nu_{dk} \Sigma_{fg'}(\vec{r}) \phi_{g'}(\vec{r}).\]  

The adjoint system of equations to the coupled system of equations in Eq. (3.19) and Eq. (3.20) can be obtained as

\[-\nabla \cdot D_r(\vec{r})\nabla \phi^*_g(\vec{r}) + \Sigma_f(\vec{r})\phi^*_g(\vec{r}) - \sum_{g'=1}^{G} \Sigma_{g'\rightarrow g}(\vec{r})\phi^*_{g'}(\vec{r}) = \lambda \chi_{fg} \sum_{g'=1}^{G} \nabla \cdot \phi_{g'} \phi^*_g(\vec{r}) + \sum_{k=1}^{K} \chi_{dkg} \lambda_k \phi^*_g(\vec{r})\]

\[-\nabla \cdot [\vec{u}(\vec{r}) C^*_k(\vec{r})] + \lambda \psi^*_k(\vec{r}) = \lambda \psi^*_k(\vec{r}), \quad k = 1, 2, \ldots, K,\]  

where

\[\psi^*_k(\vec{r}) = \sum_{g'=1}^{G} \chi_{dkg} \phi_{g'}^*(\vec{r}).\]  

Because of the negative sign in the precursor convection term in Eq. (3.23), this equation needs to be solved in the opposite direction with a boundary condition at the core outlet that can be represented as
where the position vector \( \vec{r} \) is separated into the radial and axial variables. For a 1D axial flow, the adjoint precursor concentration equation can be solved analytically in terms of the adjoint precursor concentration at the outlet as

\[
\int_{A_{\text{in}}} dA u(r,0) C^*_k(r,0) e^{-\lambda_t r} \int_{A_{\text{out}}} dA u(r,H) . \tag{3.25}
\]

For a multi-dimensional flow, Eq. (3.23) can be solved using the FDM discussed in Section 3.2.

Using the adjoint flux, the importance-weighted quasi-stationary source of neutrons from fission and precursor decay can be obtained as

\[
F(t) = \sum_i V_i \sum_{g=1}^{G} \phi^*_{g,i} \chi_{pg,i} \sum_{g=1}^{G} V_p \sum_{f} \phi_{f,i}'(t) \phi_{g,i}'(t) + \sum_i V_i \sum_{g=1}^{G} \phi^*_{g,i} \sum_{k=1}^{K} \chi_{dkg,i} \lambda_k C_{k,i}(t), \tag{3.27}
\]

where \( V_i \) is the volume of node \( i \) and the other notations are conventional. On the right hand side of Eq. (3.27), the first term represents the total importance of prompt neutrons produced by fission and the second term represents the total importance of delayed neutrons produced by precursor decay. Therefore, the effective delayed neutron fraction for a circulating fuel can be calculated as [17,18]

\[
\beta_{\text{eff}}^\text{circ}(t) = \frac{\sum_i V_i \sum_{g=1}^{G} \phi^*_{g,i} \sum_{k=1}^{K} \chi_{dkg,i} \lambda_k C_{k,i}(t)}{\sum_i V_i \sum_{g=1}^{G} \phi^*_{g,i} \chi_{pg,i} \sum_{g=1}^{G} V_p \sum_{f} \phi_{f,i}'(t) \phi_{g,i}'(t) + \sum_i V_i \sum_{g=1}^{G} \phi^*_{g,i} \sum_{k=1}^{K} \chi_{dkg,i} \lambda_k C_{k,i}(t)}. \tag{3.28}
\]

Similarly, the prompt neutron generation time for a circulating fuel can be determined as

\[
\Lambda_{\text{circ}}(t) = \frac{\sum_i V_i \sum_{g=1}^{G} \phi^*_{g,i} V^{-1}_g \phi_{g,i}'(t)}{\sum_i V_i \sum_{g=1}^{G} \phi^*_{g,i} \chi_{pg,i} \sum_{g=1}^{G} V_p \sum_{f} \phi_{f,i}'(t) \phi_{g,i}'(t) + \sum_i V_i \sum_{g=1}^{G} \phi^*_{g,i} \sum_{k=1}^{K} \chi_{dkg,i} \lambda_k C_{k,i}(t)}. \tag{3.29}
\]

It is noted that in an MSR, the delayed neutron production by precursor decay and the precursor production from fission should be differentiated since delayed neutron precursors move to other positions before they decay.
For a stationary fuel, delayed neutron precursors decay at the same position where they are produced, and thus the number of delayed neutrons produced at a position is equal to the number of precursors produced by fission. Therefore, the total number of neutrons can be determined from the quasi-stationary total fission neutrons. As a result, the effective delayed neutron fraction and the prompt neutron generation time can be determined as

$$\beta_{\text{eff}}(t) = \frac{\sum_i V_i \sum_{g=1}^G \phi_{g,i}^* \sum_{k=1}^K Z_{dkg} \sum_{g'=1}^G V_{dk} \sum_{g''=1}^G \phi_{g'',i}}{\sum_i V_i \sum_{g=1}^G \phi_{g,i}^* \sum_{g'=1}^G v \Sigma_{fg'} \phi_{g',i}}, \quad (3.30)$$

$$\Lambda(t) = \frac{\sum_i V_i \sum_{g=1}^G \phi_{g,i}^* \sum_{g'=1}^G \phi_{g',i}}{\sum_i V_i \sum_{g=1}^G \phi_{g,i}^* \sum_{g'=1}^G v \Sigma_{fg'} \phi_{g',i}}. \quad (3.31)$$

The reactivity is the importance-weighted neutron balance, i.e., the importance-weighted net neutron production due to neutron interactions with nuclides, leakage, and precursor decays, relative to the importance-weighted total neutron source. The reactivity for a circulating fuel can be divided into two components: one is the prompt fission neutron production minus the loss due to absorption and leakage, and the other is the delayed neutron production as

$$\rho_{\text{circ}}(t) = \frac{\sum_{i} \sum_{g=1}^G \phi_{g,i}^* \left[ \chi_{pg,i} \sum_{g'=1}^G v \sum_{g''=1}^G \phi_{g'',j}^* - \sum_{g''=1}^G \phi_{g'',i} + \sum_{g''=1}^G \Sigma_{g'' \to g,i} \phi_{g',i} \right] V_i - \sum_{g} J_{g,i} A_{g,i} \gamma_i}{\sum_i V_i \sum_{g=1}^G \phi_{g,i}^* \sum_{g'=1}^G \sum_{g''=1}^G v \Sigma_{fg'} \phi_{g',i} + \sum_i V_i \sum_{g=1}^G \phi_{g,i}^* \sum_{k=1}^K Z_{dkg,i} \lambda_k C_k(t) + \beta_{\text{eff}}^\text{circ}(t)}, \quad (3.32)$$

where $J_{g,i}$ is the net current of the surface $\gamma$ of node $i$ and $A_{g,i}$ is the surface area. It can be seen that the change in the effective delayed neutron fraction due to the precursor decay outside of the core is directly reflected in the reactivity.

### 3.4.2. Verification Tests of Kinetics Parameters

The adjoint flux solution capability implemented in PROTEUS-NODAL was verified by calculating the safety parameters of the MSFR benchmark problem and by comparing the resulting values with the other code results reported in the EVOL report [19]. The MSFR benchmark problem is described in Appendix A. The effective delayed neutron fractions of...
stationary ($\beta_{\text{eff}}$) and flowing fuel salts ($\beta_{\text{circ}}$), and the prompt neutron generation time were calculated for the $^{233}\text{U}$-started and TRU-started MSFR.

Table 3.2 compares the kinetics parameters calculated with PROTEUS-NODAL with the values reported in the EVOL report. The effective delayed neutron fraction of the flowing fuel salt is almost half of that of the stationary fuel salt. This is because the residence time of the fuel salt in the core equals to the transit time of the fuel salt in the external loop. Both the effective delayed neutron fraction and the prompt neutron generation time calculated with PROTEUS-NODAL are consistent with other reported values.

Table 3.2. Calculated Safety Parameters for MSFR Benchmark Problem

<table>
<thead>
<tr>
<th>Institute</th>
<th>LPSC</th>
<th>POLITO</th>
<th>POLIMI</th>
<th>TUDelft</th>
<th>Purdue</th>
<th>PROTEUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>XS library</td>
<td>ENDF/B-VI</td>
<td>JEFF-3.1.1</td>
<td>JEFF-3.1</td>
<td>ENDF/B-VII</td>
<td>ENDF/B-VII.1</td>
<td>ENDF/B-VII.0</td>
</tr>
<tr>
<td>$\beta_{\text{eff}}$ (pcm)</td>
<td>320.0</td>
<td>305.0</td>
<td>305.0</td>
<td>290.0</td>
<td>318.8</td>
<td>317.7</td>
</tr>
<tr>
<td>$\beta_{\text{circ}}$ (pcm)</td>
<td>169.5</td>
<td>117.3</td>
<td>146.0</td>
<td>124.6</td>
<td>142.1</td>
<td>141.2</td>
</tr>
<tr>
<td>$\beta_{\text{circ}}/\beta_{\text{eff}}$</td>
<td>0.529</td>
<td>0.385</td>
<td>0.479</td>
<td>0.430</td>
<td>0.446</td>
<td>0.444</td>
</tr>
<tr>
<td>$\beta_{\text{loss}}$ (pcm)</td>
<td>150.5</td>
<td>187.7</td>
<td>159.0</td>
<td>165.4</td>
<td>176.7</td>
<td>176.6</td>
</tr>
<tr>
<td>$\beta_{\text{loss}}/\beta_{\text{eff}}$</td>
<td>0.470</td>
<td>0.615</td>
<td>0.521</td>
<td>0.570</td>
<td>0.554</td>
<td>0.556</td>
</tr>
<tr>
<td>$\Lambda$ (μs)</td>
<td>1.200</td>
<td>0.971</td>
<td>1.090</td>
<td>1.150</td>
<td>–</td>
<td>1.103</td>
</tr>
</tbody>
</table>

TRU-Started

<table>
<thead>
<tr>
<th>Institute</th>
<th>LPSC</th>
<th>POLITO</th>
<th>POLIMI</th>
<th>TUDelft</th>
<th>Purdue</th>
<th>PROTEUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>XS library</td>
<td>ENDF/B-VI</td>
<td>JEFF-3.1.1</td>
<td>JEFF-3.1</td>
<td>ENDF/B-VII</td>
<td>ENDF/B-VII.1</td>
<td>ENDF/B-VII.0</td>
</tr>
<tr>
<td>$\beta_{\text{eff}}$ (pcm)</td>
<td>312.8</td>
<td>301.0</td>
<td>302.0</td>
<td>–</td>
<td>298.8</td>
<td>294.6</td>
</tr>
<tr>
<td>$\beta_{\text{circ}}$ (pcm)</td>
<td>165.5</td>
<td>–</td>
<td>147.0</td>
<td>–</td>
<td>138.5</td>
<td>135.0</td>
</tr>
<tr>
<td>$\beta_{\text{circ}}/\beta_{\text{eff}}$</td>
<td>0.529</td>
<td>–</td>
<td>0.487</td>
<td>–</td>
<td>0.464</td>
<td>1.464</td>
</tr>
<tr>
<td>$\beta_{\text{loss}}$ (pcm)</td>
<td>147.3</td>
<td>–</td>
<td>155.0</td>
<td>–</td>
<td>160.3</td>
<td>159.6</td>
</tr>
<tr>
<td>$\beta_{\text{loss}}/\beta_{\text{eff}}$</td>
<td>0.471</td>
<td>–</td>
<td>0.513</td>
<td>–</td>
<td>0.536</td>
<td>0.542</td>
</tr>
<tr>
<td>$\Lambda$ (μs)</td>
<td>0.900</td>
<td>0.783</td>
<td>0.650</td>
<td>–</td>
<td>–</td>
<td>0.623</td>
</tr>
</tbody>
</table>

Additionally, the forward and adjoint delayed neutron precursor distributions are compared for stationary and flowing fuels. Fig. 3.3 compares the axial distributions of
precursor concentrations for all the six precursor groups, and Fig. 3.4 compares the axial distributions of adjoint precursor concentrations. Fig. 3.3 shows that the precursor concentration of the stationary fuel is symmetric about the core mid-plane for all the six precursor groups. On the other hand, for the flowing fuel, the first two precursor groups show almost a uniform distribution of precursor concentrations due to their relatively large half-lives, but the precursor concentration of group five varies significantly from the core inlet to the core outlet.

Fig. 3.3. Delayed neutron precursor distribution in stationary and flowing fuels.

Fig. 3.4 also shows that the adjoint precursor concentration of the stationary fuel is symmetric about the core mid-plane for all the six precursor groups. It is noted that for the stationary fuel, the axial importance distributions of the six precursor groups are similar to each other since the delayed neutron spectra are similar. On the other hand, in the flowing
fuel, the axial importance distributions of the six precursor groups are very different because of the different decay constants. The adjoint precursor concentrations of groups 3 to 6 show significant axial variations. For the groups 1 and 2, the adjoint precursor distributions are almost constant throughout the active core.

3.5. Thermal Feedback Calculation

This section explains the coupled neutronics and thermal-hydraulics calculation scheme to account for thermal feedback. The cross section functionalization scheme for thermal feedback is discussed first, and then the overall computational procedure of the coupled calculation for thermal feedback is discussed.

3.5.1. Cross Section Functionalization

The calculation scheme for thermal feedback has been developed for fast and thermal spectrum reactor applications. To functionalize cross sections, multigroup cross sections were prepared using the multigroup cross section generation code MC2-3 [20] or Monte Carlo codes Serpent [21] and OpenMC [22]. The isotopic cross section dataset ISOTXS and the delayed neutron cross section dataset DLAYXS are generated using MC2-3 directly or using the utility code GenISOTXS [23] that converts the output data of Serpent or OpenMC to ISOTXS. In this study, a capability to convert the Serpent or OpenMC output to DLAYXS was added to GenISOTXS. Cross sections for the fast and thermal spectrum MSRs were functionalized differently based on the temperature dependence of cross sections for the Doppler effects and the existence of a moderator.

In a fast spectrum MSR, the reactivity feedback is due to the Doppler effects and the change of fuel density. In large fast reactors, the Doppler effects is approximately proportional to $1/T$ [24] and thus the microscopic cross section can be represented as a linear function of $\ln(T)$.

$$\frac{d\sigma}{dT} = \frac{\alpha}{T} \to \sigma(T) = \sigma(T_0) + \alpha \ln(T/T_0). \tag{3.33}$$

Microscopic cross sections are prepared at two temperatures $T_L$ and $T_H$. After each thermal-hydraulics calculation, the cross sections are updated as

$$\Sigma(T) = \frac{\rho(T)}{\rho(T_0)} \sum_i N_i(T_0)[\varepsilon_L\sigma_i(T_L) + \varepsilon_H\sigma_i(T_H)] , \tag{3.34}$$

where

$$\varepsilon_L = \frac{\ln(T_H/T)}{\ln(T_H/T_L)}, \quad \varepsilon_H = 1 - \varepsilon_L . \tag{3.35}$$
In a thermal spectrum MSR, cross sections depend on the moderator temperature as well as the fuel temperature. The resonance integral in thermal reactors is approximately proportional to $\sqrt{T}$ and thus the microscopic cross sections of fuel isotopes are represented as a linear function of $\sqrt{T}$. On the other hand, the moderator temperature dependency is represented as a linear function of $T$. After each thermal-hydraulics calculation, the cross sections are updated as

$$\Sigma(T_F, T_M) = \sum \frac{\rho_F(T_F)}{\rho_F(T_{F0})} \left[ N_i^F(T_{F0}) \left( \varepsilon_{L}^F \sigma_{i}^F(T_{L}^F) + \varepsilon_{H}^F \sigma_{i}^F(T_{H}^F) \right) \right] + \sum \left[ N_j^M \left( \varepsilon_{L}^M \sigma_{j}^M(T_{L}^M) + \varepsilon_{H}^M \sigma_{j}^M(T_{H}^M) \right) \right],$$

where

$$\varepsilon_{L}^F = \left( \sqrt{T_{F}^H} - \sqrt{T_{F}^L} \right) / \left( \sqrt{T_{F}^H} - \sqrt{T_{F}^L} \right), \quad \varepsilon_{L}^M = \left( T_{M}^H - T_{M}^L \right) / \left( T_{M}^H - T_{M}^L \right),$$

$$\varepsilon_{H}^F = 1 - \varepsilon_{L}^F, \quad \varepsilon_{H}^M = 1 - \varepsilon_{L}^M.$$

### 3.5.2. Calculation Flow

The coupling schemes of the neutronics and the thermal-hydraulics calculations for steady state and transient analyses are illustrated in Fig. 3.5. In steady state calculations, the neutronics and thermal-hydraulics equations are tightly coupled and solved iteratively until the power and temperature solutions converge. The initial power distribution in the core is determined by performing the nodal calculation using PROTEUS-NODAL with uniform temperature and velocity fields. At the end of each power iteration of the nodal calculation, a thermal-hydraulics calculation is performed to determine the fuel salt temperature, density, and velocity fields in the core region using the updated power distribution. Then, the cross sections in the core region are updated using the calculated fuel salt temperature distribution. Using the updated cross sections and the salt velocity field, the nodal calculation is performed again. This process continues until the power and temperature distributions converge.

In transient calculations, the neutronics and thermal-hydraulics calculations are performed once for each time step. At the beginning of each time step, the temperature, velocity and density distributions at the previous time step are used to determine the cross sections. Using the cross sections, the neutronics calculation is performed to update the power distribution. Using the updated power distribution, the thermal-hydraulics calculation is performed to calculate the new fuel salt temperature, density, and velocity fields. Then, these updated
temperature, density, and velocity distributions are used in the feedback calculation in the next time step until the end of the time step is reached.

3.6. **Verification Tests of $P_1$ Transient Solver**

In this section, verification test results of the $P_1$ transient solver are discussed. As the $SP_3$ transient solver was verified in the last year work [9], the results of the $P_1$ transient solver were compared to those of the $SP_3$ transient solver. For consistent comparison, using the $P_1$ and $SP_3$ solvers, transient calculations were first performed in hexagonal geometry without invoking the thermal feedback module. These tests were performed for both stationary and flowing fuels. Then, the same transients were analyzed in R-Z geometry with thermal feedback and the results were compared with those of the $P_1$ and $SP_3$ transient solvers for
hexagonal geometry. The thermal feedback calculation was performed using the standalone thermal-hydraulics solver of PROTEUS-NODAL, which is discussed in Appendix C.

3.6.1. Verification Tests without Thermal Feedback

The $P_1$ transient solver was first verified with the modified TWIGL benchmark problem [16]. The same hexagonal geometry core of six rings of 22.5 cm-pitch hexagons was used for both the $P_1$ and SP3 solver. The Doppler feedback effects were taken into account using the provided sets of cross sections without invoking the standalone thermal-hydraulics calculation module. Fig. 3.6 compares the calculation results. The $P_1$ solution agrees very well with the SP3 solution within 1% error. As the same hexagonal geometry model was used, this difference is solely due to the transport effect. As the diffusion approximation overestimates leakage in general, the $P_1$ transient solver yields a 1% smaller peak power than the SP3 solver.

Next tests were performed with a MSFR problem of reduced size in order to reduce the computational time. Step reactivity insertion transients were simulated for both stationary and flowing fuels. For the stationary fuel, two step-reactivity insertions of +100 pcm and -100 pcm were investigated by adjusting the fuel concentration. Fig. 3.7 and Fig. 3.8 show the power changes in the first one second for the positive and negative reactivity insertion problems, respectively. For both the positive and negative reactivity insertion problems, the $P_1$ transient solution agrees well with the SP3 transient result. An about 1% difference is observed during the prompt jump period and the error remains within 1.5%.

For a flowing fuel, the reactivity insertion was simulated by adjusting the fuel velocity in the core. An increased fuel velocity increases the loss of delayed neutrons and thus decreases the reactivity. On the contrary, a decreased fuel velocity increases the reactivity due to a decreased loss of delayed neutrons. Fig. 3.9 shows the power response to the step increase in the fuel velocity. The power decreases initially and then shows an oscillatory behavior of 4.0 s period because the fuel leaving the core flows back to the core after 2.0 s. The $P_1$ solution agrees well with the SP3 solution within 1.0%. The power response to the step velocity decrease is shown in Fig. 3.10. The decreased fuel velocity introduces a positive reactivity and thus the power increases exponentially. The $P_1$ solution agrees well with the SP3 solution within 0.6%. These results demonstrate that the $P_1$ transient solver is working properly.
Fig. 3.6. Power change with time for the modified TWIGL benchmark problem.

Fig. 3.7. Power comparison after the positive step reactivity insertion for the stationary fuel.
Fig. 3.8. Power comparison after the negative step reactivity insertion for the stationary fuel.

Fig. 3.9. Power comparison after the step increase in the fuel velocity.
3.6.2. Verification Tests with Thermal Feedback

The performance of $P_1$ solver with thermal feedback was verified by solving the steady state problem and various transient scenarios of the MSFR benchmark problem [19]. The standalone thermal-hydraulics module was used for the thermal feedback calculation. The results of three solvers were compared: SP$_3$ solver for hexagonal geometry, $P_1$ solver for hexagonal geometry, and $P_1$ solver for R-Z geometry. A hexagonal geometry core model of six rings of 22.5 cm-pitch hexagons was used.

![Graph](image.png)

Fig. 3.10. Power comparison after the step decrease in the fuel velocity.

Table 3.3 compares the eigenvalue results of three solvers for the steady state MSFR benchmark problem. The results without thermal feedback obtained with the core-averaged temperature are also included for comparison. As already shown in Table 2.3, a hexagonal geometry core model with large hexagons underestimates the eigenvalue by ~100 pcm due to the geometrical approximation. The transport effect partially recovered by the SP$_3$ solver is about +50 pcm. The thermal feedback effect is almost the same for all three solvers with a value of about 85 pcm. Although not included in this report, the power density, temperature, pressure, fuel density, and velocity solutions obtained with the three solvers agreed very well with each other.

Computational time was also compared although it is not consistent because of the non-negligible errors introduced by the approximate hexagonal geometry model. However, this comparison would provide a useful information since the selected hexagonal geometry model with a relatively large hexagon size would be a practical one to avoid a huge computational
burden. It can be seen that relative to the SP$_3$ solver for hexagonal geometry, the P$_1$ solver for R-Z geometry reduces the computational time by ~53 times for the problem without thermal feedback and 8 times for that with thermal feedback with a marginal eigenvalue difference of ~50 pcm.

### Table 3.3. Eigenvalues of Steady State MSFR Benchmark Problem

<table>
<thead>
<tr>
<th></th>
<th>No. of Iterations</th>
<th>Calculation Time (s)</th>
<th>Time Ratio</th>
<th>k-eff</th>
<th>a) Diff. (pcm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP$_3$-Hex</td>
<td>32</td>
<td>30.69</td>
<td>52.9</td>
<td>0.979492</td>
<td>-53.9 (+51.8)</td>
</tr>
<tr>
<td>P$_1$-Hex</td>
<td>17</td>
<td>10.82</td>
<td>18.7</td>
<td>0.978995</td>
<td>-105.7</td>
</tr>
<tr>
<td>P$_1$-RZ</td>
<td>15</td>
<td>0.58</td>
<td>1.0</td>
<td>0.980009</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>No. of Iterations</th>
<th>Calculation Time (s)</th>
<th>Time Ratio</th>
<th>k-eff</th>
<th>c) Diff. (pcm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP$_3$-Hex</td>
<td>42</td>
<td>42.01</td>
<td>8.0</td>
<td>0.980351</td>
<td>85.9</td>
</tr>
<tr>
<td>P$_1$-Hex</td>
<td>19</td>
<td>19.83</td>
<td>3.8</td>
<td>0.979839</td>
<td>84.4</td>
</tr>
<tr>
<td>P$_1$-RZ</td>
<td>15</td>
<td>5.24</td>
<td>1.0</td>
<td>0.980213</td>
<td>86.8</td>
</tr>
</tbody>
</table>

a) Difference in eigenvalue from the P$_1$-RZ result  
b) Difference in eigenvalue from the P$_1$-Hex result  
c) Difference of eigenvalue with thermal feedback from that without thermal feedback

For transient problems, unprotected transient overpower (UTOP), unprotected pump over speed (UPOS), unprotected loss of flow (ULOF), unprotected loss of heat sink (ULOHS), unprotected fuel salt overcooled (UFSOC) scenarios were solved using the three solvers with the same geometries used for the steady state problem. The description of each transient and the simulation methodology can be found in [9,25,26]. For each transient case, the power and core-averaged fuel salt temperature changes with time are compared in Fig. 3.11 through Fig. 3.22. It can be seen that despite the differences in geometrical models and transport approximations (diffusion vs. SP$_3$), the differences in power and fuel salt temperature of the three solutions are small. The maximum difference in power of the two P$_1$ solutions from the SP$_3$ solution is less than 0.9% for all the transients except for the super-prompt critical UTOP of a reactivity insertion of 200 pcm. In the super-prompt critical UTOP, the maximum difference is 2.0% for the P$_1$ solver with R-Z geometry and 5.0% for the P$_1$ solver with hexagonal geometry. The maximum difference in the core-averaged fuel salt temperature is less than 0.8% for all scenarios. With these comparable accuracies, the P$_1$ solver for R-Z geometry reduces the computational time about 10 to 60 times relative to the SP$_3$ solver and about 4.5 to 9.5 times relative to the P$_1$ solver for hexagonal geometry. The solution accuracy
and computational efficiency of three solvers are compared in Table 3.4 for all the six transient scenarios.

Fig. 3.11. Power evolution during the UTOP transient with 50 pcm reactivity insertion.

Fig. 3.12. Core average temperature rise across core during the UTOP transient with 50 pcm reactivity insertion.
Fig. 3.13. Power evolution during the UTOP transient with 200 pcm reactivity insertion.

Fig. 3.14. Core average temperature rise across core during the UTOP transient with 200 pcm reactivity insertion.
Fig. 3.15. Power evolution during the UPOS transient.

Fig. 3.16. Average temperature rise of fuel salt across core during the UPOS transient.
Fig. 3.17. Power evolution during the ULOF transient.

Fig. 3.18. Average temperature rise of fuel salt across core during the ULOF transient.
Fig. 3.19. Power evolution during the ULOHS transient.

Fig. 3.20. Average temperature rise of fuel salt across core during the ULOHS transient.
Fig. 3.21. Power evolution during the UFSOC transient.

Fig. 3.22. Average temperature rise of fuel salt across core during the UFSOC transient.
### Table 3.4. Performance Comparison of Three Solvers for MSFR Transient Problems

<table>
<thead>
<tr>
<th>Transient Scenario</th>
<th>No. Time Steps</th>
<th>End Time (sec)</th>
<th>Max. No. of Iterations</th>
<th>Computational Time (hr)</th>
<th>Time Ratio</th>
<th>Max. Power Diff. (%)</th>
<th>Max. Core Avg. Temperature Diff. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SP$_3$-Hex</td>
<td>P$_1$-Hex</td>
<td>P$_1$-RZ</td>
<td>SP$_3$-Hex</td>
</tr>
<tr>
<td>UTOP-50</td>
<td>4600</td>
<td>100</td>
<td>451</td>
<td>29.93</td>
<td>2.75</td>
<td>0.449</td>
<td>66.70</td>
</tr>
<tr>
<td>UTOP-200</td>
<td>11800</td>
<td>100</td>
<td>442</td>
<td>22.80</td>
<td>4.79</td>
<td>1.106</td>
<td>20.61</td>
</tr>
<tr>
<td>UPOS</td>
<td>5000</td>
<td>50</td>
<td>142</td>
<td>25.48</td>
<td>4.22</td>
<td>0.460</td>
<td>55.43</td>
</tr>
<tr>
<td>ULOF</td>
<td>7000</td>
<td>150</td>
<td>108</td>
<td>8.83</td>
<td>4.40</td>
<td>0.742</td>
<td>11.91</td>
</tr>
<tr>
<td>UFSOC</td>
<td>8000</td>
<td>200</td>
<td>82</td>
<td>9.32</td>
<td>4.78</td>
<td>0.498</td>
<td>18.72</td>
</tr>
<tr>
<td>ULOHS</td>
<td>7000</td>
<td>150</td>
<td>323</td>
<td>21.85</td>
<td>4.10</td>
<td>0.726</td>
<td>30.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SP$_3$-Hex</td>
<td>P$_1$-Hex</td>
<td>P$_1$-RZ</td>
<td>SP$_3$-Hex</td>
</tr>
<tr>
<td>UTOP-50</td>
<td></td>
<td></td>
<td></td>
<td>0.386</td>
<td>0.291</td>
<td>-</td>
<td>0.012</td>
</tr>
<tr>
<td>UTOP-200</td>
<td></td>
<td></td>
<td></td>
<td>5.772</td>
<td>2.312</td>
<td>-</td>
<td>0.028</td>
</tr>
<tr>
<td>UPOS</td>
<td></td>
<td></td>
<td></td>
<td>0.228</td>
<td>0.220</td>
<td>-</td>
<td>0.017</td>
</tr>
<tr>
<td>ULOF</td>
<td></td>
<td></td>
<td></td>
<td>0.844</td>
<td>0.054</td>
<td>-</td>
<td>0.756</td>
</tr>
<tr>
<td>UFSOC</td>
<td></td>
<td></td>
<td></td>
<td>0.651</td>
<td>0.854</td>
<td>-</td>
<td>0.028</td>
</tr>
<tr>
<td>ULOHS</td>
<td></td>
<td></td>
<td></td>
<td>0.667</td>
<td>2.719</td>
<td>-</td>
<td>0.029</td>
</tr>
</tbody>
</table>
4. Coupling of PROTEUS-NODAL and SAM

In a liquid fuel MSR, the core reactivity is directly related to the speed of flowing fuel because of the precursors that decay outside the core and the thermal expansion of the liquid fuel introduces large negative reactivity feedback. As a result, the neutronics and thermal-hydraulics are strongly coupled. In order to enhance the thermal-hydraulics modeling capabilities by overcoming the limitation of the standalone thermal-hydraulics solver, PROTEUS-NODAL has been coupled with the system analysis code SAM [11] under the MOOSE framework [12]. Since PROTEUS-NODAL is not an application of the MOOSE framework, a MOOSE wrapper named “TreeFrog” was developed. Using the MOOSE framework, a master application named “TreeKangaroo” was developed as well to control the coupling calculations between PROTEUS-NODAL and SAM. At the moment, the coupling is limited to the 1D core models of SAM. The coupled calculation of PROTEUS-NODAL and SAM was first verified by comparing the steady state analysis results with those obtained by manual iteration between PROTEUS-NODAL and the computational fluid dynamics (CFD) code ANSYS CFX [14]. Then the MSFR transient analysis results of the coupled system of PROTEUS-NODAL and SAM were compared with the other solutions found in the open literature.

The coupling calculation scheme of PROTEUS-NODAL and SAM is discussed first. This is followed by the verification test results of the SAM code with the ANSYS CFX code for the steady state MSFR problem. Then the MSFR transient analysis results of the coupled system of PROTEUS-NODAL and SAM are compared with those of other code systems. The descriptions of TreeFrog and TreeKangaroo are provided in Appendix E along with sample input files.

4.1. Coupled Calculation Scheme

Since PROTEUS-NODAL is not an application of the MOOSE framework, the mesh system and associated variables used in PROTEUS-NODAL cannot be directly communicated with other codes under the MOOSE framework. Thus, the MOOSE application called TreeFrog was developed as the wrapper for the PROTEUS-NODAL code. The communications between different MOOSE applications rely on the MOOSE “MultiApps” and “Transfer” systems. In the MOOSE framework, multiple independent applications can be defined in a tree-like multi-level structure, as shown in Fig. 4.1. In the MultiApps system, a master application controls its sub-applications. In the coupled system of PROTEUS-NODAL and SAM, TreeFrog (i.e., the MOOSE wrapper of PROTEUS-NODAL) and SAM are independently run as two sub-applications controlled by a master
application called TreeKangaroo. The Transfer system provides the functions to transfer the solution data between applications in the MultiApp system. The Transfer system consists of three main types of transfers: field mapping, post-processed spatial data, and scalar values [12]. In this work, the data transfer functions between PROTEUS-NODAL and SAM are based on the transfer of post-processed spatial data.

![MOOSE MultiApps system](image)

Fig. 4.1. MOOSE MultiApps system [12].

4.1.1. **PROTEUS-NODAL Wrapper TreeFrog**

The PROTEUS-NODAL code is wrapped inside TreeFrog with the “ExternalProblem” class of the MOOSE framework. This class contains mainly two methods: one for an executioner and the other for a data transferrer. The executioner runs PROTEUS-NODAL, while the data transferrer transfers data between PROTEUS-NODAL and TreeFrog. The MOOSE-based TreeFrog mesh is defined to store data to be passed from and to PROTEUS-NODAL such as power density, temperature, density, and velocity of the fuel salt. TreeFrog communicates with PROTEUS-NODAL before and after running PROTEUS-NODAL at each time step to ensure that the data are up to date.

PROTEUS-NODAL is compiled as a dynamic library such that the subroutines of PROTEUS-NODAL can be accessed from TreeFrog. At the beginning of a calculation, TreeFrog calls the subroutines that read the PROTEUS-NODAL input files and initializes the
relevant variables. Then the executioner runs PROTEUS-NODAL at each time step in a transient simulation or at each Picard iteration in a steady state simulation. The memory is freed at the final step of the TreeFrog run.

To facilitate the data transfer between TreeFrog and SAM, the axial mesh size of TreeFrog is set to equal to that of the SAM channel. The variables in the TreeFrog meshes and those in the nodes of PROTEUS-NODAL are transferred with appropriate interpolation and integration. The power densities in TreeFrog meshes are obtained by integrating the intra-nodal power density of the corresponding node of the PROTEUS-NODAL code over the axial direction. In both the SP3 and P1 solvers, the intra-nodal flux is represented by polynomials of a specified order as

\[ \phi_z(\xi) = \sum_{j=0}^{n_z} a_{g,j} f_j(\xi), \quad \xi = 2 \frac{z - Z_{in}}{Z_{out} - Z_{in}} - 1. \]  

(4.1)

where \( Z_{in} \) and \( Z_{out} \) are the lower and upper coordinates of the node. In the SP3 solver, \( n_z = 2 \) and \( f_j(\xi) \) is the Legendre polynomial of order \( j \). For the P1 solver, \( n_z \) is the user-specified order, \( a_{g,j} \) corresponds to \( \zeta_{g,j} \) in Eq. (2.3), and \( f_j(\xi) \) is obtained by radially averaging the basis function \( f_j(r) \) in Eq. (2.3). With the axial nodes of PROTEUS-NODAL and the axial meshes of TreeFrog shown in Fig. 4.2, the power density in the \( m \)-th axial mesh of TreeFrog in the \( i \)-th channel of SAM is calculated as

\[ P_{i,m} = \sum_{j=0}^{n_z} b_j f_j(\xi) d\xi, \quad b_j = \sum a_{g,j} k_{g} \Sigma_{f,g}, \]  

(4.2)
Fig. 4.2. Axial nodes of PROTEUS-NODAL and axial meshes of TreeFrog.

where $\xi_b$ and $\xi_u$ are the normalized positions for the lower and upper bounds of the $m$-th TreeFrog axial mesh, respectively. For the hexagonal geometry, $P_{i,m}$ is determined by averaging the power densities of the hexagonal assemblies that belong to the $i$-th channel of SAM.

The average fuel density in a node of PROTEUS-NODAL is evaluated by integrating the fuel densities in the corresponding TreeFrog meshes based on the piece-wise linear interpolation. The average fuel temperature and velocity in a node of PROTEUS-NODAL are obtained with the density-weighted integration of each variable based on a piece-wise linear interpolation.

4.1.2. Overall Computational Procedure of TreeKangaroo

TreeKangaroo was developed as the master application to control TreeFrog and SAM as independent sub-applications. Fig. 4.3 shows the structure and the coupling scheme of TreeKangaroo. The data transfer between the TreeFrog and SAM meshes are done through the TreeKangaroo mesh using the MOOSE function “MultiAppCopyTransfer,” which supports the transfer of all kinds of variables between two MultiApps if their meshes are identical.
Fig. 4.3. Data flow and coupling scheme between PROTEUS-NODAL and SAM under the MOOSE framework.

Two functions “MultiAppPowerTransferToSam” and “MultiAppVPPTransferFromSam” were implemented in TreeKangaroo for the data transfer between TreeKangaroo and SAM. The function “MultiAppPowerTransferToSam” is used to transfer the power density from TreeKangaroo to SAM. To store the power density transferred from TreeKangaroo, a new heat source object was defined in SAM as an “AuxVariables” of MOOSE. The function MultiAppPowerTransferToSam determines the mapping between the TreeKangaroo mesh and the SAM mesh. The function “MultiAppVPPTransferFromSam” is used to transfer the velocity, temperature, and density data from the SAM mesh to the TreeKangaroo mesh. In SAM, all the velocity, temperature, and density data of a channel are output to one “VectorPostprocessors”. In the “MultiAppVPPTransferFromSam” function, the velocity, temperature, and density data are read from a “VectorPostprocessors” of SAM, and the transferred data are stored as “AuxVariables” in the TreeKangaroo mesh.

For steady state calculations, the Picard iteration is used as illustrated in Fig. 4.4. Within each iteration, TreeFrog and SAM are executed sequentially. The PROTEUS-NODAL calculation is performed using the temperature, density, and velocity of SAM in the last Picard iteration. Then the power distribution calculated by PROTEUS-NODAL is mapped to the
Fig. 4.4. Coupling scheme with Picard iteration for steady state calculations.

TreeFrog mesh and is transferred to SAM. Using the updated power, the SAM calculation is performed. At the end of each iteration, the SAM solution is transferred to PROTEUS-NODAL. The convergence is checked by comparing the maximum relative error in the power distribution between two successive Picard iterations. The Picard iteration is continued until the convergence criteria is met.

For transient calculations, the conventional operator splitting technique is used instead of the Picard iteration as shown in Fig. 4.5. Transient calculations are performed after SAM and PROTEUS-NODAL read their restart files that contain the steady state solutions as an initial condition. The restart file of SAM is the standard file defined in the MOOSE framework. The subroutines in PROTEUS-NODAL were modified to allow the time step size to be controlled by TreeFrog, not by the PROTEUS-NODAL input. As the time step size of the SAM calculation needs not be as small as that of the PROTEUS-NODAL calculation, the “subcycling” option defined in the transient executioner of the MOOSE framework is utilized to allow different time step sizes between PROTEUS-NODAL and SAM. With this option, sub-applications can take multiple time steps during a single time step of the master application. Thus, the time step size of TreeKangaroo is selected as the larger one of those
specify in the PROTEUS-NODAL and SAM inputs. Within a one-time step of TreeKangaroo, the sub-application that has the larger time step size will be executed once and the other with smaller time step sizes will be executed multiple times until the time reaches the time of TreeKangaroo. Since the characteristic time in thermal-hydraulics calculations is usually larger than that in neutronics calculations, the time step size of TreeKangaroo is set to the same as that of SAM. If the SAM result does not converge, the SAM calculation is carried out again by halving the original time step size.

Fig. 4.5. Coupling scheme with operator splitting for transient calculations.
4.2. Verification Test Results

Verification tests of the coupled system of PROTEUS-NODAL and SAM were performed using the steady state and transient problems derived from the MSFR benchmark problem [10]. Since the radial crossflow is neglected in the current SAM model coupled with PROTEUS-NODAL, the effect of this simplification was first examined by comparing the results of the coupled PROTEUS-NODAL and SAM calculation for a steady state problem with those obtained by a manually coupled calculation of PROTEUS-NODAL and the commercial CFD software ANSYS CFX [14]. In this comparison, the outer loop was not modeled, and the same inlet and outlet boundary conditions were used in both SAM and ANSYS CFX. Then, the MSFR transient benchmark problems were analyzed using the coupled system of PROTEUS-NODAL and SAM by including the outer loop in the SAM model. The results were compared with the other code system results found in the open literature.

4.2.1. Comparison of Steady State Results of SAM with ANSYS CFX

The coupled calculation results of PROTEUS-NODAL and SAM were compared with the manually coupled calculation results of PROTEUS-NODAL and CFX. For a consistent comparison, the outer loop was not included and only the core channels were used in the SAM model. The inlet velocity and temperature and the outlet pressure were given as the boundary conditions for both SAM and CFX calculations. The core was modeled with multiple parallel axial one-dimensional (1D) channels with the neglect of radial crossflow in the SAM calculation, while the full 3D Navier-Stokes equations were solved in the cylindrical domain in the CFX calculation. In both the SAM and CFX calculations, the thermo-physical properties of the fuel salt were taken from Reference [19] as provided in Appendix A for LiF (78 mol%)-ThF₄ (22 mol%) properties. For both coupled calculations, the PROTEUS-NODAL calculation was performed using the P₁ solver in R-Z geometry. The active core was divided into 20 axial regions and 6 radial rings. The first and second radial rings correspond to the first and second channels in the SAM model, respectively, the third and fourth rings correspond to the third channel, and the remaining fifth and sixth rings correspond to the fourth channel. Region-dependent cross-sections were generated using the MC²-3 code [20] with the ENDF/B-VII.0 data.

In both coupled calculations, the power densities of two successive Picard iterations were used to determine the convergence with a relative tolerance of $10^{-4}$. About three iterations were needed for convergence for this problem. In the manually coupled calculation of PROTEUS-NODAL and CFX, the initial power distribution was obtained from the coupled calculation of PROTEUS-NODAL and SAM. The power distribution of PROTEUS-NODAL was fitted to a polynomial function. The third and fourth orders were used in the
fitting for the radial and axial power distributions, respectively. Using this power distribution, the CFD simulation was performed to determine the flow fields of the fuel salt. In the subsequent PROTEUS-NODAL calculation, the radial velocity field of the CFX calculation was ignored and only the axial velocity field was used. Despite the different thermal-hydraulics modeling, the multiplication factor for the coupled calculation of PROTEUS-NODAL and CFX and that of PROTEUS-NODAL and SAM agreed very well within 2 pcm (0.98117 with CFX and 0.98115 with SAM).

Fig. 4.6 shows the converged axial and radial velocity fields obtained from the coupled calculation of PROTEUS-NODAL and CFX. The axial velocity increases along the axial direction because of heating, and the axial velocity at the core outlet is higher in the central region than in the periphery region because of the higher power density. The high power density in the central region increases the fuel salt temperature and thus reduces its density, which in turn increases the flow velocity. The radial crossflow from the periphery region to the central region is also observed although the magnitude is smaller than 2% of the axial flow. (Both the positive velocity in the left half and the negative velocity in the right half are the flow from the periphery toward the core center.) Fig. 4.7 compares the axial velocity profiles of SAM and CFX. The axial velocity fields of SAM and CFX are quite different due to the neglect of the radial velocity field in the SAM calculation. The axial channel velocity of SAM increases along the axial direction in all the four channels, whereas that of CFX increases in the first three central channels and decreases in the fourth periphery channel. It is noted that the axial channel velocity of SAM is smaller than that of CFX in the three central channels.

Fig. 4.8 shows the power distribution obtained from the coupled PROTEUS-NODAL and SAM calculation and its relative difference from that obtained from the coupled PROTEUS-NODAL and CFX calculation. The smaller axial velocities of SAM at the upper part of the three central channels reduce the delayed neutrons loss outside of the core and thus yield a relatively top-skewed power distribution. The relative difference from -0.2% to 0.2% in the power distribution results in the difference in the fuel temperature distribution as shown in Fig. 4.9. The fuel temperature of SAM is slightly higher at the upper part of the core than that of CFX.
Fig. 4.6. Axial (left) and radial (right) velocity fields of the fuel salt obtained from CFX. The cylinder axis is in the middle of both figures.

Fig. 4.7. Comparison of axial velocity profiles of SAM and CFX.
Fig. 4.8. (a) Power distribution (W/cc) of coupled PROTEUS-NODAL and SAM calculation and (b) % difference from that of coupled PROTEUS-NODAL and CFX calculation.

Fig. 4.9. Comparison of axial temperature profiles of SAM and CFX.
4.2.2. Transient Tests of Coupled System of PROTEUS-NODAL and SAM

Three transient scenarios of the MSFR benchmark problem were analyzed using the coupled system of PROTEUS-NODAL and SAM: Unprotected Transient Over-Power (UTOP), Unprotected Loss of Flow (ULOF), and Unprotected Loss of Heat Sink (ULOHS) accidents. The PROTEUS-NODAL calculations were performed using the P₁ solver in R-Z geometry. The results were compared to the solutions in the open literature obtained with different code systems. One is the solution obtained from the coupled calculation with the nodal neutronics code PARCS and the system analysis code TRACE conducted at the Paul Scherrer Institute (PSI) [25]. The other is the solution obtained from the coupled calculation of a neutronics code based on the finite volume diffusion theory method and a CFD code conducted at the Delft University of Technology (TUDelft) [26].

Fig. 4.10 shows the SAM model for the MSFR benchmark problem. The core was divided into four axial 1D channels as in the steady state calculation. These channels were connected to zero-dimensional (0D) branch components. The heat exchanger and pump were connected to each other and the core channels were connected by pipes. Because of the lack of detailed design information, the heat exchanger and pump parameters were determined to yield the desired temperature increase across the core. The coolant inlet temperature at the secondary side of the heat exchanger was assumed 550°C. The multigroup cross sections and thermo-physical properties of the fuel salt used in the steady state calculation were used for the transient analyses.

Fig. 4.10. SAM model of the MSFR benchmark problem.
The UTOP transient was simulated by inserting a 50 pcm step reactivity at the beginning of the transient. This was achieved by increasing the fuel salt concentration in the core. Because of the rapid increase of the power at the beginning of the transient, the time step size for the PROTEUS-NODAL calculation was set to 0.0001 s. On the other hand, since the response of thermal-hydraulics variables to the power change was relatively much slower, the larger time step size of 0.01 s was used in the SAM calculation. Fig. 4.11 compares the three power solutions from the coupled PROTEUS-NODAL and SAM calculation, the standalone PROTEUS-NODAL calculation with its thermal-hydraulics module, and Reference [26]. The power increases at the beginning of the transient and reaches its maximum (~1.6 times of the nominal power) at around 4.9 ms. The increased power increases the fuel salt temperature, which in turn introduces a negative reactivity due to the Doppler effects and the reduced fuel salt density. As a result, the power decreases after 4.9 ms. It can be seen that the power evolution of the coupled PROTEUS-NODAL and SAM calculation reasonably agrees well with the TUDelft results up to around 1 s. However, the peak power predicted by PROTEUS-NODAL is slightly lower than the TUDelft result. This difference is attributed to the more negative temperature feedback coefficient of PROTEUS-NODAL (-7.68 pcm/K) than that of TUDelft (-6.79 pcm/K) [9] due to the use of different cross section sets.

![Power evolution during the UTOP transient.](image)

Fig. 4.12 shows the increase in the core-averaged temperature from the steady state as a function of time. The result of the coupled PROTEUS-NODAL and SAM calculation shows an increasing temperature after ~2 s since the heated fuel salt comes back to the core inlet.
On the other hand, the TUDelft result shows a decreasing temperature after ~3 s, which seems to be due to the decreasing power around 2 s. The core-averaged fuel temperature predicted by PROTEUS-NODAL is about 3K lower than the TUDelft result at the end of 10 s transient because of the lower peak power caused by the more negative temperature coefficient. For this UTOP transient, the PROTEUS-NODAL standalone solution agrees very well with the result of the coupled PROTEUS-NODAL and SAM calculation until 2 s. After 2 s, the standalone PROTEUS-NODAL solution shows some deviation from the coupled calculation result because of the simplified outer loop model.

The ULOF transient is initiated by the pump failure. Since no secondary pump was included in the SAM model, the primary pump head and the secondary flow rate were decreased exponentially with a time constant of 5 s. The decreased flow velocity reduces the number of delayed neutrons lost in the outer loop and thus introduces a positive reactivity. On the other hand, the reduced flow rate increases the core average temperature and thus induces a negative feedback due to the Doppler effects and the reduced fuel salt density. The negative feedback was larger than the positive feedback for this problem and thus the power decreases with time.

Fig. 4.13 compares the four power solutions from the coupled PROTEUS-NODAL and SAM calculation, the standalone PROTEUS-NODAL calculation with its thermal-hydraulics module, the coupled PARCS and TRACE calculation [25], and Reference [26]. The result of the coupled PROTEUS-NODAL and SAM calculation agrees well with that of TUDelft in
the first 30 s. The power decreases monotonically and reaches about 20% of the nominal full power at 30 s. After 30 s, the power obtained from the coupled PROTEUS-NODAL and SAM calculation keeps decreasing while the TUDelft power remains at about 20% of the nominal full power. This is because natural circulation was considered in the TUDelft model while not in the coupled PROTEUS-NODAL and SAM calculation.

The TUDelft power and that from coupled PROTEUS-NODAL and SAM calculation decrease more slowly than the other two solutions. This is due to the different outer loop models. While the inlet fuel salt velocity was directly decreased exponentially with a time constant of 5 s in the standalone PROTEUS-NODAL calculation with the thermal-hydraulics module and the coupled PARCS and TRACE calculation, the pump head was decreased in the calculation at TUDelft and in the coupled PROTEUS-NODAL and SAM calculation. As showed in Fig. 4.14, the average inlet velocity of the standalone PROTEUS-NODAL calculation with its thermal-hydraulics module decreases more rapidly than that of the coupled PROTEUS-NODAL and SAM calculation. The more rapid decrease in the inlet velocity leads to the more rapid decrease in the power.
Fig. 4.14. Average inlet velocity evolution during ULOF transient.

Fig. 4.15. Power evolution during ULOHS transient.
The ULOHS transient was simulated by the exponential decrease in the inlet flow rate of the secondary side of the heat exchanger with a time constant of 1 s in the SAM calculation. For the standalone PROTEUS-NODAL calculation, the heat removal rate of the heat exchanger was manually decreased exponentially with a time constant of 1 s. Fig. 4.15 shows the power evolution during the ULOHS from the time when the uncooled fuel salt flows into the core inlet. Whereas the standalone PROTEUS-NODAL calculation result agrees well with the PARCS-TRACE and TUDelft results, the power obtained from the coupled PROTEUS-NODAL and SAM calculation decreases at a slower rate than others. This is due to the difference in the heat removal rate of the heat exchanger.
5. Summary

In order to enhance the accuracy and efficiency of the PROTEUS-NODAL code for MSR simulations, several new capabilities have been implemented in the variational nodal $P_1$ solver and the PROTEUS-NODAL code has been coupled with the system analysis module SAM. First of all, to eliminate the error caused by approximating a cylindrical MSR core by a hexagonal core, a variational nodal $P_1$ method was developed for cylindrical geometries, i.e., $R-\theta-Z$, $R-\theta$, $R-Z$ and $R$ geometries. The developed capability was verified against the FDM solver of the DIF3D code using 2D and 3D steady state problems derived from the MSFR benchmark. The eigenvalue and power distribution results of the $P_1$ solver in cylindrical geometries agreed very well with the reference solutions obtained by Richardson extrapolation of two DIF3D-FDM solutions. Relative to the fine hexagonal geometry model that yields a comparable accuracy, the computational time was reduced about 100 times by an $R-Z$ model and about 7 to 8 times by an $R-\theta-Z$ model. Although the computational gain depends on the specific problem, it is obvious that a cylindrical geometry model is more efficient and accurate than a hexagonal geometry model for cylindrical geometry problems.

The transient analysis capability limited to the $P_3$ solver in hexagonal geometry was then extended to the $P_1$ solvers for Cartesian, triangular, hexagonal, and cylindrical geometries. The TFSP solver was extended to the $P_1$ solvers, and a FDM solver for the delayed neutron precursor equation was developed to overcome the limitations of the previously developed MOC and MoL solvers and to facilitate the coupling with the multi-dimensional flow models of SAM in the near future. To improve the computational efficiency for transient analyses, a CMFD acceleration scheme was also implemented. A capability to calculate the kinetics parameters in flowing fuel reactors was also added by implementing a steady state adjoint equation solver. The developed $P_1$ transient solver was verified against the $P_3$ transient solver verified in the last year work using various transient problems in hexagonal geometry. The test results showed that the $P_1$ solutions agree very well with the $P_3$ solutions. These tests also showed that the CMFD acceleration scheme reduces the computational time for the TFSP a few tens of times.

The $P_1$ transient solver in $R-Z$ geometry was also tested by comparing the results with those of the hexagonal geometry option of the $P_1$ solver and those of the $P_3$ solver. Various unprotected transient scenarios of the MSFR benchmark problem were solved with thermal feedback, including UTOP, UPOS, ULOF, ULOHS, and UFSOC accidents. Despite the differences in geometrical models and transport approximations (diffusion vs. $P_3$), the three solutions showed very good agreement in the power and core-averaged fuel salt temperature. The maximum difference in power was less than 0.9% for all the transients except for a super-prompt critical UTOP of a reactivity insertion of 200 pcm. In the latter case, the maximum difference was 2.0% for the $P_1$ solver with $R-Z$ geometry and 5.0% for the $P_1$ solver with hexagonal geometry. The maximum difference in the core-averaged fuel salt
temperature was less than 0.8% for all the five scenarios. With these comparable accuracies, the P₁ solver in R-Z geometry reduces the computational time about 10 to 60 times relative to the SP₃ solver and about 5 to 10 times relative to the P₁ solver in hexagonal geometry.

In order to enhance the thermal-hydraulics modeling capabilities by overcoming the limitation of the standalone thermal-hydraulics solver, PROTEUS-NODAL has been coupled with SAM under the MOOSE framework. A MOOSE sub-application named TreeFrog was developed as the wrapper for PROTEUS-NODAL to communicate with other MOOSE applications. A MOOSE master application named TreeKangaroo was also developed to control the coupling calculation of TreeFrog and SAM. Different time step sizes between PROTEUS-NODAL and SAM are allowed in the coupled transient calculations by using the “subcycling” option of the MOOSE transient executioner. The Picard iteration was used in the coupled steady state calculation, and the operator-splitting method was used in the coupled transient calculations.

Verification tests of the coupled system of PROTEUS-NODAL and SAM were performed using the steady state and transient problems derived from the MSFR benchmark problem. Since the radial crossflow is neglected in the current SAM model, the effect of this simplification was first examined by comparing the steady state results with those obtained by a manually coupled calculation of PROTEUS-NODAL and ANSYS CFX. In this comparison, the outer loop was not modeled, and the same inlet and outlet boundary conditions were used in both SAM and ANSYS CFX. The SAM calculation used four parallel axial channels and CFX performed the full 3D CFD calculation in the cylindrical geometry of the MSFR core. Due to the neglect of the radial velocity field in the SAM calculation, SAM underestimated the axial velocity at the core center slightly, and this resulted in a slightly top-skewed power distribution: 0.1% overestimation in the upper part and 0.2% underestimation in the lower part of the core. The UTOP, ULOF, and ULOHS accidents of the MSFR transient benchmark were analyzed by including the outer loop in the SAM model. The results were compared with the PSI solutions from a coupled PARCS and TRACE calculation and the TUDelft solutions obtained from a coupled neutron diffusion and CFD calculation. In general, the power and core-averaged fuel temperature solutions of the coupled PROTEUS-NODAL and SAM calculations agreed well with the other solutions.

Future work will focus on the coupling of PROTEUS-NODAL with the recently developed 3D SAM to consider the radial crossflow of the fuel salt and the implementation of the transport kernel of the variational nodal method in the P₁ solver to model the transport effects more accurately. In addition, validation tests will be performed using the experimental data of the molten salt reactor experiment (MSRE).
References

14. ANSYS® CFX, Release 18.0, ANSYSYS, Inc.
Appendix A. Description of MSFR Benchmark Problem

The MSFR design specification and relevant material properties are mainly obtained from the EVOL (Evaluation and Viability of Liquid Fuel Fast Reactor) project report [19]. The MSFR design has a thermal power of 3000 MW\textsubscript{th} and utilizes a fast neutron spectrum based on the thorium fuel cycle. The MSFR utilizes a binary fluoride salt, composed of LiF (77.5 molar %) and a heavy nuclei (HN) mixture (22.5 molar %) initially composed of fertile thorium and fissile material. The fuel salt is circulated in the core and in the external loop, which consists of 16 branches each with the heat exchanger, pump, and associated instruments and pipes. A schematic representation of the MSFR core is shown in Fig. A.1.

Fig. A.1. Schematic representation of the molten salt fast reactor.

The MSFR is operated between 650 °C and 750 °C with a circulation time of 4.0 s, which can be controlled based on the power level and the temperature rise in the core. The fuel salt temperature rise across the core during normal operation is fixed at 100 °C. The active reactor core has a cylindrical shape with the same height and diameter of 2.255 m to improve the breeding ratio and reduce the neutron leakage. The total fuel salt volume is 18 m\textsuperscript{3}, half of which is in the active core. The radial reflector includes a fertile blanket of 50 cm thick to increase the breeding ratio and to shield the external components, and it is filled with a fertile salt of LiF-\textsubscript{Th}F\textsubscript{4} (same type as the core salt). The walls of the blanket are surrounded by a 20 cm thick layer of B\textsubscript{4}C to enhance the shielding of the external components. The external core
structures and the heat exchangers are protected by thick reflectors made of nickel-based alloys for corrosion resistance. Table A.1 provides the main characteristics of the MSFR core.

**Table A.1. Characteristics of the Molten Salt Fast Reactor**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermal/electric power</td>
<td>3000 MWth / 1300 MWe</td>
</tr>
<tr>
<td>Fuel salt</td>
<td></td>
</tr>
<tr>
<td>Temperature increase in the core</td>
<td>100 °C</td>
</tr>
<tr>
<td>Fuel salt melting point</td>
<td>565 °C</td>
</tr>
<tr>
<td>Fuel Inlet/outlet temperature</td>
<td>650 °C / 750 °C</td>
</tr>
<tr>
<td>Mean fuel salt temperature</td>
<td>700 °C</td>
</tr>
<tr>
<td>Fuel salt density</td>
<td>4.1249 g/cm³</td>
</tr>
<tr>
<td>Core Height / Core Radius</td>
<td>2.255 m / 1.1275 m</td>
</tr>
<tr>
<td>Fuel salt volume</td>
<td>18 m³</td>
</tr>
<tr>
<td>Total fuel salt cycle in the fuel circuit</td>
<td>4.0 s</td>
</tr>
<tr>
<td>Flow rate</td>
<td>4.5 m³/s</td>
</tr>
</tbody>
</table>

Fuel salt in its initial composition has a fissile element either $^{233}\text{U}$ ($^{233}\text{U}$-started) or the transuranic elements produced by PWRs (TRU-started). The $^{233}\text{U}$-started fuel salt is composed of LiF-ThF$_4$-$^{233}$UF$_3$, and the TRU-started fuel salt is composed of LiF-ThF$_4$-(TRU)F$_3$ with a TRU mixture of Pu (87.5%), Np (6.3%), Am (5.3%), and Cm (0.9%). The initial fuel compositions of both options are provided in Table A.2, and the isotopic compositions of the TRU element are listed in Table A.3.

**Table A.2. Initial Composition of the Fuel Salt of MSFR**

<table>
<thead>
<tr>
<th>Element</th>
<th>$^{233}\text{U}$-started</th>
<th>TRU-started</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mole fraction (%)</td>
<td>Mole fraction (%)</td>
<td></td>
</tr>
<tr>
<td>Th</td>
<td>19.985</td>
<td>16.068</td>
</tr>
<tr>
<td>U-233</td>
<td>2.515</td>
<td>5.628</td>
</tr>
<tr>
<td>Pu</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Np</td>
<td>0.405</td>
<td></td>
</tr>
<tr>
<td>Am</td>
<td>0.341</td>
<td></td>
</tr>
<tr>
<td>Cm</td>
<td>0.058</td>
<td></td>
</tr>
</tbody>
</table>

**Table A.3. TRU-element Fuel Composition**

<table>
<thead>
<tr>
<th>Isotope</th>
<th>Mole fraction (%)</th>
<th>Isotope</th>
<th>Mole fraction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pu-238</td>
<td>2.7</td>
<td>Np-237</td>
<td>6.3</td>
</tr>
<tr>
<td>Pu-239</td>
<td>45.9</td>
<td>Cm-244</td>
<td>0.8</td>
</tr>
<tr>
<td>Pu-240</td>
<td>21.5</td>
<td>Cm-245</td>
<td>0.1</td>
</tr>
<tr>
<td>Pu-241</td>
<td>10.7</td>
<td>Am-241</td>
<td>3.4</td>
</tr>
<tr>
<td>Pu-242</td>
<td>6.7</td>
<td>Am-243</td>
<td>1.9</td>
</tr>
</tbody>
</table>
The MSFR utilizes 16 heat exchangers and 16 pumps. Each heat exchanger has the capability of extracting 187 MW during normal operation, and each pump provides a flow rate of about 0.28 m$^3$/s to maintain the temperature rise in the core. The heat exchanger considered in the following simulations consists of about 8000 tubes with a length of 2.0 m and an inner diameter of 0.4 cm, with a fuel salt volume of 0.203 m$^3$. The inlet and outlet temperatures of the heat exchanger are 750 °C and 650 °C, respectively. The MSFR design includes an intermediate circuit that is used to separate the radioactive fuel salt from the energy conversion system that uses FLiNaK as a coolant with a temperature of 550 °C. The heat exchanger is assumed to be of shell and tube type as the MSRE and MSBR designs developed at the Oak Ridge National Laboratory. The temperature range of the MSFR design is between 838 K (the salt freezing point) and 1600 K (the melting point of the nickel alloy of the core structures).

Table A.4 provides the thermo-physical properties for salt of LiF (78 mol%)-ThF$_4$ (22 mol%) as provided in reference [27]. It is assumed that the fission products and new heavy nuclei produced during reactor operation will not impact the fuel salt thermo-physical properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Correlation</th>
<th>Validity Range [°C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density [kg/m$^3$]</td>
<td>$\rho = 4094 - 0.882(T[K]-1008)$</td>
<td>620-850</td>
</tr>
<tr>
<td>Kinematic Viscosity [m$^2$/s]</td>
<td>$\nu = 5.54 \times 10^{-8} e^{3689/T[K]}$</td>
<td>625-846</td>
</tr>
<tr>
<td>Dynamic Viscosity [Pa.s]</td>
<td>$\mu = \left[2.268 - 4.886 \times 10^{-4} (T[K]-1008)\right] \times 10^{-4} e^{3689/T[K]}$</td>
<td>625-846</td>
</tr>
<tr>
<td>Thermal Conductivity [W/m.K]</td>
<td>$k = 0.928 + 8.397 \times 10^{-5} T[K]$</td>
<td>618-747</td>
</tr>
</tbody>
</table>

In order to perform decay heat calculations, the decay heat fractions, and decay constants should be known. The parameters in Reference [28] were adopted in this study. Table A.5 provides the decay heat fractions and decay constants for TRU and $^{233}$U started MSFR.

<table>
<thead>
<tr>
<th>Group</th>
<th>$^{233}$U-started</th>
<th>TRU-Started</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fraction (%)</td>
<td>Decay Constant (s$^{-1}$)</td>
</tr>
<tr>
<td>1</td>
<td>1.28</td>
<td>0.2160</td>
</tr>
<tr>
<td>2</td>
<td>1.23</td>
<td>0.0182</td>
</tr>
<tr>
<td>3</td>
<td>2.08</td>
<td>2.590×10$^{-4}$</td>
</tr>
<tr>
<td>Total</td>
<td>4.59</td>
<td></td>
</tr>
</tbody>
</table>
Appendix B. Comparison of Three Solution Methods for Delayed Neutron Precursor Equation of Flowing Fuel

As discussed in Section 3.2, three different numerical methods were examined to solve the delayed neutron precursor equation for 1D axial flow: the method of characteristics (MOC), the method of lines (MoL), and the finite difference method (FDM). FDM was selected and implemented in PROTEUS-NODAL since it produces the null transient result correctly and it can be easily extended to multi-dimensional flows. The FDM method was discussed in Section 3.2, and thus in this appendix, MOC and MoL are discussed and their results for three transient scenarios are compared with the FDM solutions.

B.1. Solution Methods for 1D Axial Flow

B.1.1. Method of Characteristics

Under the assumption that the fuel velocity is constant within a node, the precursor equation for 1D axial flow given in Eq. (3.11) can be solved analytically using the MOC. Along a characteristic line that can be represented by a parameter \( s \), the terms including the temporal and spatial derivatives can be represented by the total derivative with respect to \( s \).

Denoting the precursor concentration of family \( k \) in a radial mesh \( i \) by \( C_{k,i} \), its derivative with respect to \( s \) becomes

\[
\frac{d}{ds} C_{k,i}(z(s),t(s)) = \frac{\partial C_{k,i}}{\partial t} \frac{dt}{ds} + \frac{\partial C_{k,i}}{\partial z} \frac{dz}{ds},
\]

where \( i \) refers to the index for a radial mesh. Therefore, Eq. (3.11) can be written as an ordinary differential equation (ODE) as

\[
\frac{d}{ds} C_{k,j}(s) + \lambda_k C_{k,j}(s) = \lambda \psi_{k,j}(s),
\]

with the following auxiliary ODEs for the characteristic line:

\[
\frac{dt}{ds} = 1, \quad \frac{dz}{ds} = u_i.
\]

The characteristic lines can be determined by solving Eq. (B.3) as

\[
\begin{cases} 
  t = s + t_0 \\
  z = u_i s + z_0
\end{cases}
\]

where \( t_0 \) and \( z_0 \) are the coordinates for the starting point where the initial or boundary condition is given. Along a characteristic line, the ODE in Eq. (B.2) can be solved as
\[ C_{k,j}(s) = C_{k,j}(0)e^{-\lambda_k s} + \lambda \int_0^s \psi_{k,j}(s')e^{-\lambda_k (s-s')} \, ds'. \]  

(B.5)

In Eq. (B.5), \( C_{k,j}(0) = C_{k,j}(z_0,t_0) \) is the initial or boundary condition given at the starting point of the characteristic line. For example, as illustrated in Fig. B.1, the precursor concentration in an axial node \( j \) with an interval \([z_{j-1}, z_j]\) during a time interval \([t_{n-1}, t_n]\) can be obtained starting from the initial condition \( C_{k,j}^{n-1}(z) \), \( z_{j-1} \leq z \leq z_j \) given at \( t = t_{n-1} \) or the inlet boundary condition \( C_{k,j}^{in}(t) \), \( t_{n-1} \leq t \leq t_n \) given at \( z = z_{j-1} \).

\[ \begin{align*}
\text{Fig. } B.1. & \text{ Illustration of characteristic lines for precursor concentration calculation.} \\
\text{Along a characteristic line starting from the line } t = t_{n-1}, & \quad z_0 = z - u_i (t - t_{n-1}) > z_{j-1} \text{ and thus } t - t_{n-1} < (z - z_{j-1}) / u_i. \text{ On the other hand, along a characteristic line starting from the line } \\
z = z_{j-1}, & \quad t_0 = t - (z - z_{j-1}) / u_i > t_{n-1} \text{ and thus } t - t_{n-1} > (z - z_{j-1}) / u_i. \text{ Therefore, the precursor concentration in the axial node } j \text{ during a time interval } [t_{n-1}, t_n] \text{ can be determined as} \\
C_{k,j}(z,t) = & \begin{cases} \\
C_{k,j}^{in}(t)e^{-\lambda_z z_{j-1} / u_i} + \lambda \int_{z_{j-1}}^z \psi_{k,i} \left[ z' - t - z' / u_i \right] e^{-\lambda_z z' / u_i} \, dz', & t - t_{n-1} > z - z_{j-1} / u_i \\
C_{k,j}^{n-1}(z)e^{-\lambda_z (t-t_{n-1})} + \lambda \int_{z_{j-1}(t-t_{n-1})}^{z_{j-1}} \psi_{k,i} \left[ z' - t - z' / u_i \right] e^{-\lambda_z z' / u_i} \, dz', & t - t_{n-1} < z - z_{j-1} / u_i \\
\end{cases} \quad (B.6)
\end{align*} \]

At each time node, the precursor concentrations at the outgoing surface of each node are determined using Eq. (B.6). The precursor production and decay during the time interval \( \Delta t_n \) are evaluated along the characteristic line from a starting point \( z_{\text{start}}^n \) to an ending point at
\[ z = z_{\text{out}}. \] For an ending point at \( z = z_{\text{out}} \) at \( t = t_n \), the starting point at \( t = t_{n-1} \) can be determined as

\[ z_{\text{start}}^n = z_{\text{out}} - u_i \Delta t_n. \quad (B.7) \]

The precursor concentration at the starting position is determined by the linear interpolation of precursor concentrations at the previous time node as

\[ C_{k,i}(z_{\text{start}}^n) = \gamma C_{k,i}^{n-1}(z_{\text{in}}) + (1 - \gamma) C_{k,i}^{n-1}(z_{\text{out}}), \quad (B.8) \]

where \( z_{\text{in}} \) is the inlet coordinate of the node of interest and

\[ \gamma = \frac{z_{\text{out}} - z_{\text{start}}^n}{z_{\text{out}} - z_{\text{in}}}. \quad (B.9) \]

In order to evaluate the precursor generation rate in Eq. (B.6), the unknown flux at the current time node \( t_s \) is approximated by assuming the flux varies exponentially in the time interval \([t_{n-1}, t_n]\) as

\[ \phi_{k,i}(z, t) = \phi_{k,i}^{n-1}(z) e^{\alpha_n (t - t_{n-1})}. \quad (B.10) \]

The inverse period \( \alpha_n \) is approximately determined using the power levels at \( t_{n-2} \) and \( t_{n-1} \) as

\[ \alpha_n = \frac{1}{\Delta t_{n-1}} \ln \frac{P_{n-1}}{P_{n-2}}. \quad (B.11) \]

Fig. B.2 illustrates the method to calculate the precursor concentration at \( t = t_n \) on the outlet surface of a node \( j \) (i.e., at \( z = z_j \)). If the fuel salt moves through \( L \) nodes during a time interval \([t_{n-1}, t_n]\), then starting from the node \( j - (L - 1) \), the outgoing precursor concentrations of the nodes between \( z_{\text{start}}^n \) to \( z_j \) are calculated as
Multiphysics Coupling of PROTEUS-NODAL and SAM for Molten Salt Reactor Simulation
February 28, 2020

\[
C_{k,i,j-1}^{n} = \begin{cases} 
C_{k,i,j-1}^{n} - \frac{z_{\text{start}}}{\hat{u}_{i,j}^{n}} \int_{z_{\text{start}}}^{z_{i-1}} \psi_{k,i,j-1}(z') e^{-\left(\lambda_{k} + \alpha_{k}\right) \frac{z_{j-1} - z'}{\hat{u}_{i,j}^{n}}} dz' , & z_{j-1} < z_{\text{start}} \\
C_{k,i,j-1}^{n} + \lambda_{k} \int_{z_{j-1}}^{z_{i-1}} \psi_{k,i,j-1}(z') e^{-\left(\lambda_{k} + \alpha_{k}\right) \frac{z_{j-1} - z'}{\hat{u}_{i,j}^{n}}} dz' , & z_{j-1} > z_{\text{start}} 
\end{cases} , \quad l = 1, \ldots, L - 1
\]

where \( \hat{u}_{i,j}^{n} \) is the average fuel velocity during the time step \([t_{n-1}, t_{n}]\) in an axial node \( j \) in a radial mesh \( i \), and the outgoing concentrations from a node are used as the incoming concentrations for the next node.

Fig. B.2. Fuel salt path at each time step from a starting point to a node outlet.

With this method, the precursor concentrations at \( t = t_{n} \) are calculated using the precursor concentrations at \( t = t_{n-1} \) and the precursor generation rates extrapolated from the power levels at the previous two time nodes. It was observed that this method could be unstable for low velocities with which the fuel salt can move only a small fraction of a node during a time interval.

B.1.2. Method of Lines

In this method, the precursor equation in Eq. (3.11) is solved semi-analytically by converting the PDE to a system of ODEs and by solving the resulting system of ODEs using a proper ODE solver. This can be done by discretizing one of the derivatives of Eq. (3.11) and solving the system of ODEs for the remaining variable. The most important advantage of this MoL approach comes from the fact that ODE is generally simpler to solve than PDE and the implicit methods can be used for discretization. In this study, the time derivative is discretized with the backward Euler (implicit) method as
\[
\frac{\partial}{\partial t} C^{n}_{k,j}(z,t) = \frac{C^{n}_{k,j}(z) - C^{n-1}_{k,j}(z)}{\Delta t_n}, \quad \Delta t_n = t_n - t_{n-1}.
\]  

(B.13)

By inserting Eq. (B.13) into Eq. (3.11), the following equation can be obtained for each radial node \(i\):

\[
\frac{d}{dz} \left[ u^n_i(z) C^n_{k,j}(z) \right] + \left( \lambda^n_k + \frac{1}{\Delta t_n} \right) C^n_{k,j}(z) = \lambda^n \psi^n_{k,j}(z) + \frac{1}{\Delta t_n} C^{n-1}_{k,j}(z).
\]  

(B.14)

The total delayed neutron precursor source at the right hand side is related to both the delayed neutron precursor source at the current time step and the delayed neutron precursor concentration at the previous time step. The time-dependent delayed precursor equation at each time step can be written in a more convenient form to solve as

\[
\frac{d}{dz} \left[ u^n_i(z) C^n_{k,j}(z) \right] + \frac{\lambda^n_k}{u^n_i(z)} [u^n_i(z) C^n_{k,j}(z)] = Q^n_{k,j}(z),
\]  

(B.15)

where

\[
\lambda^n_k = \lambda_k + \frac{1}{\Delta t_n}, \quad Q^n_{k,j}(z) = \lambda^n \psi^n_{k,j}(z) + \frac{1}{\Delta t_n} C^{n-1}_{k,j}(z).
\]  

(B.16)

Instead of using an ODE solver, Eq. (B.15) can be solved analytically for the delayed neutron precursor concentration in terms of the inlet precursor concentration and the precursor concentration at the previous time step as

\[
u^n_i(z) C^n_{k,j}(z) = e^{-\lambda^n_k \Delta t_n} \left[ u^n_i(z_0) C^n_{k,j}(z_0) + \int_{z_0}^{z} dz' Q^n_{k,j}(z') e^{\lambda^n_k \Delta t_n} \right].
\]  

(B.17)

Then, the outgoing and average precursor concentrations of an axial node \(j\) at a time node \(n\) can be obtained as

\[
C^n_{k,j} = \frac{u^n_{l,j-1}}{u^n_{l,j}} e^{-\lambda^n_k \Delta t_n} C^n_{k,j-1} + \frac{u^n_{l,j}}{u^n_{l,j} \lambda^n_k} Q^n_{k,j} \left[ 1 - e^{-\lambda^n_k \Delta t_n} \right],
\]  

(B.18)

\[
\bar{C}^n_{k,j} = \frac{u^n_{l,j-1}}{u^n_{l,j}} \frac{1}{\lambda^n_k \Delta t_n} \left[ 1 - e^{-\lambda^n_k \Delta t_n} \right] C^n_{k,j-1} + \frac{1}{\lambda^n_k} Q^n_{k,j} \left\{ 1 - \frac{1}{\lambda^n_k \Delta t_n} \left[ 1 - e^{-\lambda^n_k \Delta t_n} \right] \right\}.
\]  

(B.19)

The node-averaged precursor concentration in Eq. (B.19) depends on the fission rate at the end of a time step. In the MoL method, the fission rates are determined iteratively by coupling Eq. (B.19) with the TFSP at the end of a time step, while in the MOC approach,
they were determined approximately with an assumed exponential flux shape with an inverse period determined by extrapolating the power levels of two previous time nodes. However, the time discretization error introduces some errors in the initial precursor concentrations, and this could lead to an instability issue.

B.2. Comparison of Transient Solutions

The accuracy of the three solution approaches, FDM, MOC, and MoL were examined by solving a null transient problem and the pump startup and coast down transient problems without thermal feedback. For the latter two transients, the fuel velocity was increased or decreased exponentially with a time constant of 1 s, and a fuel salt transient time of 4 s (2 s in the core and 2 s outside of the core) was used.

Fig. B.3 compares the power evolutions during the first 10 s obtained with the three solution methods for a null transient. It is clearly seen that the FDM is stable and produces the steady state solution accurately. However, the MOC and MoL power solutions show diverging behaviors due to error accumulation although the magnitudes are very small. The MoL power solution increases steadily while the MOC power solution decreases. It appears that these unstable behaviors are due to the approximations introduced in MOC and MoL that do not reproduce the initial steady state precursor concentrations.

The power response to the change of the fuel salt flow rate due to the pump startup and pump coast-down transients are shown in Fig. B.4 and Fig. B.5, respectively. The pump startup introduces a negative reactivity due to the increased delayed neutron fraction decaying outside of the core, and thus the power decreases with time in an oscillatory way as shown in Fig. B.4. Relative to the FDM solution, the MOC and MoL solutions show oscillatory errors. The MOC solution error varies from about -6% to ~3%, but it can be seen that the amplitude decreases with time. The MoL solution error varies from about -3% to ~3%, and its amplitude initially increases but decreases after ~20 s. The pump coast-down introduces a positive reactivity and thus the power increases exponentially as shown in Fig. B.5. It can be seen that both the MOC and MoL solution errors increase monotonically in the opposite direction as in the null transient case shown in Fig. B.3.
Fig. B.3. Power evolution in null transient without thermal feedback.

Fig. B.4. Power evolution in pump start-up transient without thermal feedback.
Fig. B.5. Power evolution in pump coast-down transient without thermal feedback.
Appendix C. Standalone MSR Thermal-Hydraulics Solver of PROTEUS

In order to account for thermal feedback effects, a single-phase, parallel-channel thermal-hydraulics model was added to PROTEUS-NODAL. In this model, the thermal-fluidic behavior of the core is described by representing the core by one-dimensional parallel channels. For a thermal MSR with solid moderator, a given total flow rate is split among the parallel flow channels to satisfy the equal pressure drop boundary conditions. Meanwhile, for a fast spectrum MSR, a uniform mass flow rate is applied at the core inlet without considering flow splitting.

C.1. Governing Equations

Ignoring the axial heat conduction and the shear forces due to velocity gradients in the fuel salt, the mass, momentum, and energy balance equations for one-dimensional flow in a vertical channel can be written as [29]

\[
\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial z} = 0, 
\]

\[
\frac{\partial (\rho u)}{\partial t} + \frac{\partial (\rho u^2)}{\partial z} = -\frac{\partial P}{\partial z} - \rho g - \left(\frac{\partial P}{\partial z}\right)_{\text{fric}}, 
\]

\[
\frac{\partial (\rho h)}{\partial t} + \frac{\partial (\rho uh)}{\partial z} = q_{\text{salt}}^{m} + \frac{q^{"P_h}}{A}. 
\]

where $\rho$, $u$, $h$, and $P$ are the density, velocity, temperature, and pressure of the fuel salt, respectively, $q_{\text{salt}}^{m}$ is the volumetric heat source produced in the fuel salt, $q^{\"}$ is the surface heat flux, $P_h$ and $A$ are the wetted perimeter and flow area of fuel salt channel, and $g$ is the gravitational acceleration. The friction pressure gradient is related to the wall shear stress and the momentum flux as

\[
\left(\frac{\partial P}{\partial z}\right)_{\text{fric}} = -\frac{\tau_w P_h}{A} = \frac{f \rho u^2}{2 D_h}, 
\]

where $\tau_w$ is the wall shear stress, $D_h$ is the hydraulic diameter, and $f$ is the friction factor for a smooth tube and all flow regimes, which is obtained from Blasius and Petukhov [30].
relations for Darcy friction factor as a function of Reynolds number for smooth pipes

\[
f = \begin{cases} 
0.316 \Re^{-0.25} & \text{Re} < 3000 \quad \text{(Blasius relation)} \\
1.0 & \text{3000} < \text{Re} < 10^6 \\
(1.82 \ln(\text{Re}) - 1.64)^2 & \text{Re} \geq 10^6 
\end{cases}, \tag{C.5}
\]

where \( \text{Re} \) is the Reynolds number defined as

\[
\text{Re} = \frac{\rho u D_h}{\mu}. \tag{C.6}
\]

The momentum and energy equations can be simplified by combining each with the continuity equation in Eq. (C.1) as

\[
\rho \frac{\partial u}{\partial t} + \rho u \frac{\partial u}{\partial z} = -\frac{\partial P}{\partial z} - \rho g \left( \frac{\partial P}{\partial z} \right)_{\text{fric}}, \tag{C.7}
\]

\[
\rho \frac{\partial h}{\partial t} + \rho u \frac{\partial h}{\partial z} = q_{\text{salt}}^m + \frac{\partial P}{\partial z} + u \left[ \frac{\partial P}{\partial z} + \left( \frac{\partial P}{\partial z} \right)_{\text{fric}} \right], \tag{C.8}
\]

The thermodynamic relation for the enthalpy change can be written as

\[
dh = \frac{dh}{dT} \left. dT + \frac{dh}{dP} \right|_{T} dP = c_p dT + \frac{1 - \beta T}{\rho} dP, \tag{C.9}
\]

Since the energy variation due to pressure changes and frictional dissipation is usually negligible compared to the heat addition and enthalpy change terms, Eq. (C.9) can be simplified as

\[
dh = \left. \frac{dh}{dT} \right|_{T} dT = c_p dT. \tag{C.10}
\]

Since the material properties of the fuel salt depend on the temperature only, it is convenient to write the energy balance equation Eq. (C.8) in terms of temperature using Eq. (C.10) as

\[
\rho c_p \frac{\partial T}{\partial t} + \rho u c_p \frac{\partial T}{\partial z} = q_{\text{salt}}^m + \frac{q^m P}{A}. \tag{C.11}
\]

For the boundary conditions of these conservation equations, the inlet flow rate, the inlet enthalpy, and the outlet pressure can be used. These boundary conditions are determined
using the given system outlet pressure and total salt flow rate. In the steady state, the time derivatives of Eq. (C.1), Eq. (C.7), and Eq. (C.11) are zero. In this case, the mass flux $\rho u$ is constant, and hence Eq. (C.7) and Eq. (C.11) can be integrated separately as:

$$T(z) = T_{in} + \frac{1}{\rho u c_p A} \int_0^z q'_{sub}(z') dz' + \frac{P}{\rho u c_p A} \int_0^z q''_{p}(z') dz',$$

(C.12)

$$P(z) = P_{in} + (\rho u)^2 \left[ \frac{1}{\rho_{in}} - \frac{1}{\rho(z)} \right] - g \int_0^z \rho(z') dz' - \frac{(\rho u)^2}{2D_h} \int_0^z f(z') dz'. $$

(C.13)

Since $\rho$ is generally a function of $T$ and $P$, Eq. (C.12) represents a non-linear equation for $P(z)$. If $\rho$ is independent of $P$ (i.e., incompressible), $\rho(z)$ can be determined with $T(z)$ obtained from Eq. (C.7), hence Eq. (C.13) is solved by simple integrations.

Neglecting the axial conduction, the heat conduction in the annual moderator ring can be written as

$$\frac{\partial}{\partial t} \left[ \rho_m c_{p,m} T_m(r, z, t) \right] = \frac{1}{r} \frac{\partial}{\partial r} \left[ k_m r \frac{\partial}{\partial r} T_m(r, z, t) \right] + q_m^m(r, z, t),$$

(C.14)

where $\rho_m$, $c_{p,m}$, $T_m$ are the solid moderator density, specific heat, and temperature, respectively, $q_m^m$ is the volumetric heat source generated in the solid moderator due to gamma heating, and $k_m$ is the heat conduction coefficient of the solid moderator. Fig. C.1 shows a schematic configuration of the fuel salt and solid moderator unit cell.

![Fig. C.1. Radial configuration of unit cell.](image)
The heat conduction equation for annular moderator is coupled with the fuel salt energy equation in Eq. (C.3) or Eq. (C.11). The moderator temperature at the moderator inner wall is coupled with the bulk fuel salt temperature through the Newton’s law for heat convection

\[
k_m \frac{\partial T_m (r, z, t)}{\partial r} \Bigg|_{r=a} = h_f (a, z, t) = q^p (z, t)
\]  (C.15)

The heat transfer coefficient \( h_f \) is calculated as

\[
h_f = \frac{k_f}{L} \text{Nu},
\]  (C.16)

where \( L \) is the characteristic length of the fuel channel, \( k_f \) is the heat conduction coefficient of the fuel salt, and Nu is the Nusselt number given by

\[
\text{Nu} = \begin{cases} 
4.36, & \text{Re} \leq 2300 \\
0.023 \text{Re}^{0.8} \text{Pr}^{0.4}, & \text{Re} > 2300 \\
0.59 \text{Ra}^{0.25}, & \text{Re} \leq 2300 \\
0.10 \text{Ra}^{1/3}, & \text{Re} > 2300 
\end{cases}
\]  (C.17)

In this expression, Re is the Reynolds number defined in Eq. (C.6), and Pr and Ra are the Prandtl and the Rayleigh numbers, respectively, defined as

\[
\text{Pr} = \frac{c_p \rho_f \mu_f}{k_f},
\]  (C.18)

\[
\text{Ra} = \text{Gr} \cdot \text{Pr} = \frac{c_p \rho_f^2 g \beta [T_m (a) - T_f (z)] L^3}{k_f \mu_f},
\]  (C.19)

where \( \rho_f \), \( \mu_f \), and \( \beta \) are the density, dynamic viscosity, and temperature expansion coefficient of fuel salt, respectively, and Gr is the Grashof number defined as

\[
\text{Gr} = \frac{\rho_f^2 g \beta [T_m (a) - T_f (z)] L^3}{\mu_f^2}.
\]  (C.20)
The other boundary condition for the heat conduction in moderator is given by the symmetry condition at the outer boundary of the moderator

$$\frac{\partial T_m(r, z, t)}{\partial r} \bigg|_{r=b} = 0. \quad \text{(C.21)}$$

The fuel salt temperatures in the lower and upper plena are assumed equal to the fuel salt inlet and outlet temperatures, respectively. This yields the following boundary conditions

$$T_f(r, 0, t) = T_{in}(t), \quad \text{(C.22)}$$

$$T_f(r, H, t) = T_{out}(t), \quad \text{(C.23)}$$

where $T_{in}$ is the fuel salt temperature at the core inlet and $T_{out}$ is the fuel salt temperature at the core outlet. The boundary conditions in Eq. (C.15), Eq. (C.21), Eq. (C.22), and Eq. (C.23) can be represented by a general mixed boundary condition

$$\alpha \vec{n} \cdot k \nabla T + \beta T = \gamma, \quad \text{(C.24)}$$

where $\vec{n}$ is the outward normal vector of the boundary surface. That is, $\alpha = 1$, $\beta = -h_f$ and $\gamma = -h_f T_f$ for Eq. (C.15), $\alpha = 1$ and $\beta = \gamma = 0$ for Eq. (C.21), $\alpha = 0$, $\beta = 1$ and $\gamma = T_{in}$ for Eq. (C.22), and $\alpha = 0$, $\beta = 1$ and $\gamma = T_{out}$ for Eq. (C.23).

### C.2. Spatial Discretization

The control volume approach is adopted in this study, where the problem domain is divided into computational cells in the axial direction. Each grid point is placed at the geometric center of each cell. Field variables such as density, pressure, and temperature are defined at these grid points, and the flow variables such as velocity component and mass flux are defined on cell faces. The main control volumes for mass and energy balances are defined by these cells as shown in Fig. C.2. The momentum control volume is displaced from the main control volume in the velocity direction, extending from grid points to grid points and encompassing the cell face upon which the velocity component is defined. The finite difference equations are derived by integrating the mass, momentum, and energy equations over the control volumes under the assumption that the values at the center of each control volume prevail over the control volume.
Fig. C.2. Main and momentum control volumes and state variables.

The mass and energy balance equations can be obtained by integrating Eq. (C.1) and Eq. (C.11), respectively, over the main control volume \( i \) \((z_{i-1} \leq z \leq z_i)\) and representing the fuel salt densities and temperatures at cell faces with the cell-center values. The momentum balance equation for the cell is obtained by integrating Eq. (C.7) over the momentum control volume \((z_{i-1} + \Delta z_i / 2 \leq z \leq z_i + \Delta z_{i+1} / 2)\). Using the up-wind differencing scheme for upward flow in a constant flow area, the spatially discretized forms of the mass, energy, and momentum conservation equations are obtained as

\[
\frac{\partial \bar{\rho}_i}{\partial t} = \frac{\bar{\rho}_{i-1} u_{i-1} - \bar{\rho}_i u_i}{\Delta z_i},
\]

\[
\frac{\partial \bar{T}_i}{\partial t} = \frac{\bar{\rho}_{i-1} u_{i-1} \bar{T}_{i-1} - \bar{\rho}_i \bar{T}_i}{\Delta z_i} + \frac{\bar{q}_i}{c_{p,i} \bar{\rho}_i A} + \frac{q_i^n P_{i,n}}{c_{p,i} \bar{\rho}_i A},
\]

\[
\frac{d}{dt}(\bar{\rho}_i u_i) = \frac{2}{\Delta z_i + \Delta z_{i+1}} \left( \frac{\bar{\rho}_{i-1}^2 u_{i-1}^2}{\bar{\rho}_i} - \frac{\bar{\rho}_i^2 u_i^2}{\bar{\rho}_{i+1}} \right) + \frac{2}{\Delta z_i + \Delta z_{i+1}} (\bar{P}_i - \bar{P}_{i+1}) + \frac{2}{\Delta z_i + \Delta z_{i+1}} g \frac{f_i \bar{\rho}_i u_i^2}{2D_h},
\]

where \( \rho_i = \rho(\bar{P}_i, \bar{h}_i) = \bar{\rho}_i \) and \( \bar{T}_i \) are the fuel salt density and temperature, respectively, at the cell center, \( \bar{q}_i \) is the cell-averaged heat source, \( u_{i-1} \) and \( u_i \) are the velocities at the inlet and
outlet faces of cell $i$, respectively, and $\Delta z_i = z_i - z_{i-1}$. Using Eq. (C.25) and Eq. (C.27), the momentum balance equation can be reduced to

$$
\frac{du_i}{dt} = \frac{\bar{\rho}_{i-1} u_{i-1}}{\bar{\rho}_i} \left( \frac{2}{\Delta z_i + \Delta z_{i+1}} \frac{\bar{\rho}_{i-1} u_{i-1}}{\bar{\rho}_i} - u_i \right) - u_i^2 \left( \frac{2}{\Delta z_i + \Delta z_{i+1}} \frac{\bar{\rho}_i}{\bar{\rho}_{i+1}} - 1 \right)
$$

$$
+ \frac{2}{\bar{\rho}_i (\Delta z_i + \Delta z_{i+1})} \left( \bar{P}_i - \bar{P}_{i+1} \right) \left( \frac{\bar{\rho} \Delta z_i + \bar{\rho}_{i+1} \Delta z_{i+1}}{\bar{\rho}_i (\Delta z_i + \Delta z_{i+1})} \right) g - \frac{f_i u_i^2}{2D_h},
$$

(C.28)

where $\bar{P}_i$ and $\bar{P}_{i+1}$ are the pressure at the center of the cell $i$ and $i+1$, respectively. For the cell at the top of the channel, Eq. (C.27) is reduced to

$$
\frac{du_i}{dt} = \frac{1}{\Delta z_i} \left( \frac{2}{\bar{\rho}_i} \frac{\bar{\rho}_{i-1} u_{i-1}}{\bar{\rho}_i} - u_i \right) - \frac{2}{\bar{\rho}_i} \frac{\bar{\rho}_i}{\bar{P}_{out}} \frac{(\bar{\rho}_i - \bar{P}_{out})}{\Delta z_i} - \frac{f_i u_i^2}{2D_h}.
$$

(C.29)

where $P_{out}$ is the outlet pressure.

For the heat conduction equation, the annular moderator is divided into computational cells, where the temperatures are defined at the cell center as shown in Fig. C.3. The finite difference equations can be derived by integrating the heat conduction equation over each cell volume under the assumption that the value at the cell center prevail over the cell volume.

![Fig. C.3. Mesh structure for heat conduction calculations.](image-url)
Consider a cell \((i, j)\) with \(z_{i+1} \leq z \leq z_i\) and \(r_{j-1} \leq r \leq r_j\). Integrating Eq. (C.14) over the cell volume, we have

\[
\frac{d}{dt} \left( \rho_{m,ij} c_{m,ij} T_{m,ij} \right) V_{m,ij} = \left[ \frac{2\pi r_j k_{m,ij}}{\Delta z_i} \frac{\partial}{\partial r} \left[ \frac{1}{\Delta z_i} \int_{z_{i+1}}^{z_i} T_m(r,z,t) \, dz \right] \right]_{r=r_j} \Delta z_i + \bar{q}_{m,ij} V_{m,ij},
\]

where \(V_{m,ij} = \pi(r_j^2 - r_{j-1}^2)\Delta z_i\), \(A_{m,j} = \pi(r_j^2 - r_{j-1}^2)\) and

\[
\bar{T}_{m,ij} = \frac{1}{V_{m,ij}} \int_{z_{i+1}}^{z_i} \int_{r_{j-1}}^{r_j} T_m(r,z,t) 2\pi r \, dr \, dz.
\]

Assuming that the temperature varies linearly from the center of the mesh cell to the midpoints of its surfaces, the derivatives at \(r = r_i\) of two adjacent cells \((i, j)\) and \((i+1, j)\) can be obtained as

\[
\frac{\partial}{\partial r} \bar{T}_{m,ij} \bigg|_{r=r_j} \approx \frac{\bar{T}_{m,i,j+1/2} - \bar{T}_{m,ij}}{\Delta r_j / 2},
\]

\[
\frac{\partial}{\partial r} \bar{T}_{m,i,j+1} \bigg|_{r=r_j} \approx \frac{\bar{T}_{m,i,j+1} - \bar{T}_{m,i,j+1/2}}{\Delta r_{j+1} / 2}.
\]

where \(\bar{T}_{i,j+1/2}\) is the surface average temperature of the interface of the cells \((i, j)\) and \((i, j+1)\). Thus the continuity conditions at the cell surface can be written as
\[ -k_{m,ij} \frac{T_{m,i,j+1/2} - T_{m,ij}}{\Delta r_j / 2} = -k_{m,ij} \frac{T_{m,i,j+1} - T_{m,i,j+1/2}}{\Delta r_{j+1} / 2}, \]  
\tag{C.34}

Eq. (C.34) can be solved for the interface temperature \( T_{i,j+1/2} \) as

\[ T_{m,i,j+1/2} = \frac{(k_{m,ij} / \Delta r_j)T_{m,ij} + (k_{m,ij+1} / \Delta r_{j+1})T_{m,i,j+1}}{k_{m,ij} / \Delta r_j + k_{m,ij+1} / \Delta r_{j+1}}. \]  
\tag{C.35}

Then the heat flux at the interface can be given as

\[ \vec{q}^n_{m,r,d}(r,t) = -k_{m,ij} \frac{\partial}{\partial r} \left[ \frac{1}{\Delta z_i} \int_{z_i}^{z} T_m(r,z,t)dz \right] = -k_{m,ij} \frac{\partial}{\partial r} T_{m,i,j}(r,t), \]  
\tag{C.36}

\[ \vec{q}^n_{m,r,d} \big|_{r=r_j} = a'_{j+1,i} (T_{m,i,j} - T_{m,i,j+1}) = -a'_{j+1,i} (T_{m,i,j+1} - T_{m,i,j}), \]  
\tag{C.37}

where

\[ a'_{j+1,i} = \frac{1}{\Delta r_j / (2k_{m,i}) + \Delta r_{j+1} / (2k_{m,ij+1})}. \]  
\tag{C.38}

When a cell surface corresponds to an external boundary, the external boundary condition in Eq. (C.24) provides the relation needed for determining the boundary coupling coefficients. Under the assumption that the temperature varies linearly from the center of the mesh cell to the midpoints of its surfaces, the temperature at a boundary can be approximated as

\[ T_b = T_y + (\vec{n} \cdot \nabla T)(\Delta_n / 2), \]  
\tag{C.39}

where \( \Delta_n = \Delta r_i \) or \( \Delta_n = \Delta z_j \). By substituting Eq. (C.39) into Eq. (C24), the normal derivative on a boundary surface can be obtained as

\[ \vec{n} \cdot \nabla T = -\frac{\beta}{\alpha k + \beta \Delta_n / 2} T_{m,ij} + \frac{\gamma}{\alpha k + \beta \Delta_n / 2}, \]  
\tag{C.40}

Thus, the heat flux at the radial boundaries can be obtained as
\[ \overline{q}_{m,r,l}^{\alpha} \bigg|_{r=r_0} = -k_{m,ij} \frac{\partial T}{\partial x} \bigg|_{x=x_0} = \begin{cases} \alpha_{j,b} T_{m,ij} - b_{j,b}, & \text{right boundary} \\ -\alpha_{j,b} T_{m,ij} + b_{j,b}, & \text{left boundary} \end{cases} \] (C.41)

where

\[ a_{j,b} = \frac{\beta}{\alpha + \beta \Delta r_j / (2k_{m,ij})}, \] (C.42)

\[ b_{j,b} = \frac{\gamma}{\alpha + \beta \Delta r_j / (2k_{m,ij})}. \] (C.43)

By inserting the surface fluxes in Eq. (C.37) and Eq. (C.41) into equation (C.30), the mesh-centered finite difference equations are obtained as

\[ \frac{d}{dt} \left( \rho_{m,ij} c_{m,p,ij} T_{m,ij} \right) = c_{ij,j+1} T_{m,i,j+1} + c_{ij,j-1} T_{m,i,j-1} - d_{ij} T_{m,ij} + \dot{q}_{m,ij}^\alpha \quad (2 \leq j \leq J - 1), \] (C.44)

\[ \frac{d}{dt} \left( \rho_{j,j} c_{j,p,ij} T_{j,ij} \right) = c_{ij,j+1} T_{j,i,j+1} - d_{ij} T_{j,ij} + \dot{q}_{j,ij}^\alpha \quad (j = 1), \] (C.45)

\[ \frac{d}{dt} \left( \rho_{j,j} c_{j,p,ij} T_{j,ij} \right) = c_{ij,j-1} T_{j,i,j-1} - d_{ij} T_{j,ij} + \dot{q}_{j,ij}^\alpha \quad (j = J). \] (C.46)

where

\[ c_{ij,j+1} = \frac{2\pi r_j h_j}{V_{m,ij}} a_{j,b} = \frac{2r_j}{r_{j+1} - r_j} a_{j,b}, \quad 1 \leq j \leq J - 1 \] (C.47)

\[ c_{ij,j-1} = \frac{2\pi r_{j-1} h_j}{V_{m,ij}} a_{j,b} = \frac{2r_{j-1}}{r_{j} - r_{j-1}} a_{j,b}, \quad 2 \leq j \leq J \] (C.48)

\[ c_{jb} = \frac{2\pi r_j h_j}{V_{m,ij}} a_{jb} = \frac{2r_j}{r_1 - r_0} h_j \frac{h_j}{1 + h_j \Delta r_j / (2k_{m,ij})}, \quad j = 1 \] (C.49)

\[ d_{ij} = \begin{cases} c_{ij,j+1} + c_{ij,j-1}, & 2 \leq j \leq J - 1 \\ c_{ij,j+1} + c_{jb}, & j = 1 \\ c_{ij,j-1}, & j = J \end{cases} \] (C.50)
C.3. Temporal Discretization

The spatially discretized equations, Eq. (C.25), Eq. (C.26), and Eq. (C.27) provide a system of nonlinear ordinary differential equations for the cell-averaged temperatures, the cell-averaged pressures, and the cell-face velocities of the fuel salt. A system of linear equations can be obtained by employing a fully explicit temporal difference scheme. However, for incompressible flow, the fully explicit scheme cannot produce an equation to update the fuel salt pressure as shown below. On the other hand, the fully implicit scheme yields a system of nonlinear equations, which requires a more complicated solution scheme. As a compromise, a semi-implicit scheme can be used as in the RELAP-5 code [31]. This semi-implicit numerical solution scheme is based on replacing the system of differential equations with a system of finite difference equations partially implicit in time. The implicit terms are formulated to be linear in the dependent variables at a new time. This results in a linear time-advancement matrix that is solved by direct inversion using a sparse matrix routine. Implicitness is selected such that the field equations can be reduced to a single difference equation per fluid control volume or mesh cell, which is generally in terms of the hydrodynamic pressure [32].

C.3.1. Fully Explicit Scheme

Using the fully explicit temporal difference scheme, the finite difference equations for the mass, energy and momentum equations Eq. (C.25), Eq. (C.26), and Eq. (C.27) can be written as

\[ \rho_i^{n+1} = \rho_i^n + (\rho_i^n u_i^n - \rho_i^n u_i^n) \frac{\Delta t}{\Delta z_i}, \]

\[ T_i^{n+1} = T_i^n + \left[ \frac{\rho_i^n}{\rho_i^n} u_i^n (T_i^n - \bar{T}_i^n) + \frac{q_i^n}{\rho_i^n c_{p,i}^n A_i} + \frac{q_i^n P_i^2 \Delta z_i}{\rho_i^n c_{p,i}^n A_i} \right] \frac{\Delta t}{\Delta z_i}, \]

\[ u_i^{n+1} = u_i^n + \left[ \frac{\rho_i^n}{\rho_i^n} u_i^n \left( \frac{2}{\Delta z_i + \Delta z_{i+1}} \frac{\rho_i^n}{\rho_i^n} u_i^n - \frac{u_i^n}{\Delta z_i} \right) - (u_i^n)^2 \left( \frac{2}{\Delta z_i + \Delta z_{i+1}} \frac{\rho_i^n}{\rho_i^n} - \frac{1}{\Delta z_i} \right) \right] \frac{\Delta t}{\Delta z_i}, \]

Since the density is generally a function of temperature and pressure, Eq. (C.51) to Eq. (C.53) represent a system of equations for the pressure, temperature, and velocity at the new time node. Once the density, enthalpy, and velocity values at the new time node are
determined, the new time pressure can be determined to yield the new time density. If the density is independent of pressure (i.e., incompressible), Eq. (C.51) and Eq. (C.52) are redundant, and there is no equation to update the pressure. Thus, this fully explicit scheme does not yield a proper system of equations.

With the fully explicit temporal difference scheme, the finite difference equations for the heat conduction equations in Eq. (C.44) through Eq. (C.46) can be obtained as

\[
\bar{\rho}_{i+1}^n = \bar{\rho}_i^n + (\bar{\rho}_{i-1}^n u_{i-1}^{n+1} - \bar{\rho}_i^n u_i^{n+1}) \frac{\Delta t_n}{\Delta z_i},
\]

\[
\bar{T}_{i+1}^n = \bar{T}_i^n + \left[ \frac{\bar{\rho}_{i-1}^n u_{i-1}^{n+1}}{\bar{\rho}_i^n} (\bar{T}_{i-1}^n - \bar{T}_i^n) + \frac{\bar{q}_i^n}{\bar{\rho}_i^n c_{p,i} A_i} + \frac{\bar{q}_{i+1}^n P_{i+1} \Delta z_i}{\bar{\rho}_i^n c_{p,i} A_i} \right] \frac{\Delta t_n}{\Delta z_i},
\]

\[
u_{i+1}^n = \nu_i^n + \left[ \frac{\bar{\rho}_i^n u_i^{n+1}}{\bar{\rho}_i^n} \left( \frac{2}{\Delta z_i + \Delta z_{i+1}} \bar{P}_{i+1}^{n+1} u_{i-1}^{n+1} - u_i^{n+1} \right) \right] \left( \nu_i^n \right)^2 \left[ \frac{2}{\Delta z_i + \Delta z_{i+1}} \bar{P}_{i+1}^{n+1} - 1 \right] \frac{\Delta t_n}{\Delta z_i}
\]

\[
u_{i+1}^n = \nu_i^n + \left[ \frac{\bar{\rho}_i^n u_i^{n+1}}{\bar{\rho}_i^n} \left( \frac{2}{\Delta z_i + \Delta z_{i+1}} \bar{P}_{i+1}^{n+1} u_{i-1}^{n+1} - u_i^{n+1} \right) \right] \left( \nu_i^n \right)^2 \left[ \frac{2}{\Delta z_i + \Delta z_{i+1}} \bar{P}_{i+1}^{n+1} - 1 \right] \frac{\Delta t_n}{\Delta z_i}
\]

\[
u_{i+1}^n = \nu_i^n + \left[ \frac{\bar{\rho}_i^n u_i^{n+1}}{\bar{\rho}_i^n} \left( \frac{2}{\Delta z_i + \Delta z_{i+1}} \bar{P}_{i+1}^{n+1} u_{i-1}^{n+1} - u_i^{n+1} \right) \right] \left( \nu_i^n \right)^2 \left[ \frac{2}{\Delta z_i + \Delta z_{i+1}} \bar{P}_{i+1}^{n+1} - 1 \right] \frac{\Delta t_n}{\Delta z_i}
\]

\[
u_{i+1}^n = \nu_i^n + \left[ \frac{\bar{\rho}_i^n u_i^{n+1}}{\bar{\rho}_i^n} \left( \frac{2}{\Delta z_i + \Delta z_{i+1}} \bar{P}_{i+1}^{n+1} u_{i-1}^{n+1} - u_i^{n+1} \right) \right] \left( \nu_i^n \right)^2 \left[ \frac{2}{\Delta z_i + \Delta z_{i+1}} \bar{P}_{i+1}^{n+1} - 1 \right] \frac{\Delta t_n}{\Delta z_i}
\]

C.3.2. Semi-Implicit Scheme

Using the semi-implicit method of RELAP-5 [31] the finite difference equations for the mass, energy, and momentum equations in Eq. (C.25), Eq. (C.26), and Eq. (C.27) can be written as

\[
\bar{\rho}_{i+1}^n = \bar{\rho}_i^n + (\bar{\rho}_{i-1}^n u_{i-1}^{n+1} - \bar{\rho}_i^n u_i^{n+1}) \frac{\Delta t_n}{\Delta z_i},
\]

\[
T_{i+1}^n = T_i^n + \left[ \frac{\bar{\rho}_{i-1}^n u_{i-1}^{n+1}}{\bar{\rho}_i^n} (T_{i-1}^n - T_i^n) + \frac{\bar{q}_i^n}{\bar{\rho}_i^n c_{p,i} A_i} + \frac{\bar{q}_{i+1}^n P_{i+1} \Delta z_i}{\bar{\rho}_i^n c_{p,i} A_i} \right] \frac{\Delta t_n}{\Delta z_i},
\]

\[
u_{i+1}^n = \nu_i^n + \left[ \frac{\bar{\rho}_i^n u_i^{n+1}}{\bar{\rho}_i^n} \left( \frac{2}{\Delta z_i + \Delta z_{i+1}} \bar{P}_{i+1}^{n+1} u_{i-1}^{n+1} - u_i^{n+1} \right) \right] \left( \nu_i^n \right)^2 \left[ \frac{2}{\Delta z_i + \Delta z_{i+1}} \bar{P}_{i+1}^{n+1} - 1 \right] \frac{\Delta t_n}{\Delta z_i}
\]

For the node at the top, Eq. (C.59) becomes

\[
u_{i+1}^n = \nu_i^n + \left[ \frac{\bar{\rho}_i^n u_i^{n+1}}{\bar{\rho}_i^n} \left( \frac{2}{\Delta z_i + \Delta z_{i+1}} \bar{P}_{i+1}^{n+1} u_{i-1}^{n+1} - u_i^{n+1} \right) \right] \left( \nu_i^n \right)^2 \left[ \frac{2}{\Delta z_i + \Delta z_{i+1}} \bar{P}_{i+1}^{n+1} - 1 \right] \frac{\Delta t_n}{\Delta z_i}
\]
The provisional advanced time density can be obtained by linearizing the density state relation about the old-time value as

$$\bar{\rho}_{i}^{n+1} = \bar{\rho}_{i}^{n} + \frac{\partial \rho}{\partial T} \left[ \bar{T}_{i}^{n+1} - \bar{T}_{i}^{n} \right] + \frac{\partial \rho}{\partial P} \left[ \bar{P}_{i}^{n+1} - \bar{P}_{i}^{n} \right]. \tag{C.61}$$

Inserting Eq. (C.61) into the mass balance equation Eq. (C.57), then

$$\frac{\partial \rho}{\partial T} \left[ \bar{T}_{i}^{n+1} - \bar{T}_{i}^{n} \right] + \frac{\partial \rho}{\partial P} \left[ \bar{P}_{i}^{n+1} - \bar{P}_{i}^{n} \right] = (\bar{P}_{i-1}^{n} u_{i-1}^{n+1} - \bar{P}_{i}^{n} u_{i}^{n+1}) \frac{\Delta t_{n}}{\Delta z_{i}}. \tag{C.62}$$

Eliminating the new time temperature using Eq. (C.58) and Eq. (C.62), the provisional pressure change can be obtained as

$$\bar{P}_{i}^{n+1} - \bar{P}_{i}^{n} = \left( \frac{\partial \rho}{\partial P} \right) \left[ \frac{1}{\bar{\rho}_{i}^{n}} \left( 1 - \frac{\partial \rho}{\partial T} \left[ \bar{T}_{i}^{n+1} - \bar{T}_{i}^{n} \right] \right) \right] u_{i-1}^{n+1} - \bar{P}_{i}^{n} u_{i}^{n+1} \frac{\Delta t_{n}}{\Delta z_{i}} \tag{C.63}$$

Substituting the new time velocities in Eq. (C.59) into Eq. (C.63), a system of linear equations for I cell-center pressures at new time can be obtained. Then, Eq. (C.59) and Eq. (C.63) can be written as

$$u_{i}^{n+1} = \alpha_{i}^{n} + \beta_{i}^{n} (\bar{P}_{i}^{n+1} - \bar{P}_{i+1}^{n+1}), \tag{C.64}$$

$$\bar{P}_{i}^{n+1} - \bar{P}_{i}^{n} = \gamma_{i}^{n} u_{i-1}^{n+1} - \eta_{i}^{n} u_{i}^{n+1} - \zeta_{i}^{n}, \tag{C.65}$$

where $\bar{P}_{i+1}^{n+1} = P_{out}^{n+1}$ and

$$\alpha_{i}^{n} = u_{i}^{n} + \left[ \frac{\bar{P}_{i}^{n} u_{i}^{n}}{\bar{\rho}_{i}^{n}} \left( \frac{2}{\Delta z_{i} + \Delta z_{i+1}} - \frac{u_{i}^{n}}{\Delta z_{i}} \right) - \left( \frac{2 \bar{P}_{i}^{n}}{\Delta z_{i} + \Delta z_{i+1}} \right) \right] \Delta t_{n}, \quad i = 1, 2, \ldots, I - 1, \tag{C.66}$$

$$\alpha_{i}^{n} = u_{i}^{n} + \left[ \frac{\bar{P}_{i-1}^{n} u_{i-1}^{n}}{\bar{\rho}_{i-1}^{n}} \left( \frac{2}{\Delta z_{i} + \Delta z_{i-1}} - \frac{u_{i-1}^{n}}{\Delta z_{i-1}} \right) - \left( \frac{2 \bar{P}_{i-1}^{n}}{\Delta z_{i+1} + \Delta z_{i-1}} \right) \right] \Delta t_{n}. \tag{C.67}$$
\[
\beta_i^n = \frac{2\Delta t_n}{\rho_i^n (\Delta z_i + \Delta z_{i+1})}, \quad i = 1, 2, \cdots, I-1,
\]

\[
\beta_i^n = \frac{2\Delta t_n}{\rho_i \Delta z_i}, \quad (C.69)
\]

\[
\xi_i^n = \rho_{i-1} \left( \frac{\partial \rho}{\partial P_i} \right) \left( 1 - \frac{1}{\rho_i} \frac{\partial \rho}{\partial T_i} \left[ (T_{i-1}^n - \bar{T}_i^n) \right] \frac{\Delta t_n}{\Delta z_i}, \quad (C.70)
\]

\[
\eta_i^n = \rho_i \left( \frac{\partial \rho}{\partial P_i} \right) \frac{\Delta t_n}{\Delta z_i}, \quad (C.71)
\]

\[
\zeta_i^n = \left( \frac{\partial \rho}{\partial P_i} \right)^{-1} \frac{\partial \rho}{\partial T_i} \left[ \frac{\bar{q}_{i}^{n} - P^{+}_{h,i}}{\rho_i^n c_{p,i} A_i \Delta z_i} + \frac{q_{i}^{n} P_{h,i}}{\rho_i^n c_{p,i} A_i} \right] \Delta t_n. \quad (C.72)
\]

Substituting the new time velocities \( u_i^{n+1} \) and \( u_{i-1}^{n+1} \) for \( i = 2, 3, \cdots, I-1 \) in Eq. (C.64) into Eq. (C.65), will result in

\[
\xi^n_i \beta^n_{i-1} P^{n+1}_{i-1} - (1 + \xi^n_i \beta^n_{i-1} + \eta^n_i \beta^n_i) \bar{P}^{n+1}_i + \eta^n_i \beta^n_i \bar{P}^{n+1}_{i+1} = \zeta^n_i - \xi^n_i \alpha^n_{i-1} + \eta^n_i \alpha^n_i - \bar{P}^n_i, \quad i = 2, 3, \cdots, I-1, \quad (C.73)
\]

Using the specified outlet pressure boundary condition, i.e., \( \bar{P}^{n+1}_{i+1} = P^{+}_{out} \), the pressure equation for \( i = I \) can be obtained as

\[
\xi^n_i \beta^n_{i-1} \bar{P}^{n+1}_{i-1} - (1 + \xi^n_i \beta^n_{i-1} + \eta^n_i \beta^n_i) \bar{P}^n_i = \zeta^n_i - \xi^n_i \alpha^n_{i-1} + \eta^n_i \alpha^n_i - \bar{P}^n_i - \eta^n_i \beta^n_i \bar{P}^{n+1}_{out}, \quad i = I, \quad (C.74)
\]

If the inlet velocity \( u_{in} \) is given, the pressure equation for \( i = 1 \) can be derived by inserting \( u^{n+1}_1 \) in Eq. (C.64) into Eq. (C.65) as

\[
-(1 + \eta^n_i \beta^n_i) \bar{P}^{n+1} + \eta^n_i \beta^n_i \bar{P}^{n+1}_{i+1} = \zeta^n_i - \xi^n_i u^{n+1}_{in} + \eta^n_i \alpha^n_i - \bar{P}^n_i, \quad i = 1. \quad (C.75)
\]

For specified inlet flow rate and temperature, Eq. (C.58) and Eq. (C.62) for \( i = 1 \) can be written as

\[
\bar{T}^{n+1}_i = \bar{T}_i^n + \left[ \frac{\rho_{in}^{n+1} u^{n+1}_{in}}{\rho_i^n} (\bar{T}_i^n - \bar{T}_i^n) + \frac{\bar{q}_i^n - P^{+}_{h,i}}{\rho_i^n c_{p,i} A_i} + \frac{q_{i}^{n} P_{h,i}}{\rho_i^n c_{p,i} A_i} \Delta z_i \right] \Delta t_i. \quad (C.76)
\]
As a result, Eq. (C.65) can be written as

\[
P_i^{n+1} - P_i^n = \tilde{\xi}_i^n \rho_i^n u_i^{n+1} - \eta_i^n u_i^{n+1} - \zeta_i^n,
\]

where

\[
\tilde{\xi}_i^n = \left( \frac{\partial \rho}{\partial P} \right)_i^{-1} \left[ 1 - \frac{1}{\rho_i^n} \frac{\partial \rho}{\partial T} \left( T_i^n - T_i^n \right) \right] \Delta t_i,
\]

\[
\eta_i^n = \left( \frac{\partial \rho}{\partial P} \right)_i^{-1} \frac{\rho_i^n}{\Delta z_i} \Delta t_i,
\]

\[
\zeta_i^n = \left( \frac{\partial \rho}{\partial P} \right)_i^{-1} \frac{\rho_i^n}{\Delta z_i} \Delta t_i.
\]

Inserting \( u_i^{n+1} \) in Eq. (C.65) into Eq. (C.79), the pressure equation for \( i = 1 \) can be derived as

\[
-(1 + \eta_i^n \beta_i^n) P_i^{n+1} + \eta_i^n \beta_i^n P_i^{n+1} = \zeta_i^n - \tilde{\xi}_i^n w_i^{n+1} - P_i^n, \quad i = 1.
\]

The coefficient matrix for the system of linear equations for the time advanced pressures given in Eq. (C.73), Eq. (C.74), and Eq. (C.75) is a tri-diagonal matrix. Thus, this system of equations can easily be solved by the forward elimination and backward substitution algorithm. Substitution of the new time pressures into Eq. (C.65) yields the new time velocities. Using the new time velocities, the new time densities and temperatures can be obtained from Eq. (C.57) and Eq. (C.58), respectively. For an incompressible flow, the density does not depend on the pressure. In this case, the provisional advanced time density in Eq. (C.61) can be linearized as

\[
\tilde{\rho}_i^{n+1} = \rho_i^n + \frac{\partial \rho}{\partial T} \left( T_i^{n+1} - T_i^n \right).
\]

Inserting Eq. (C.84) into the mass balance equation Eq. (C.58), then
\[
\frac{\partial \rho}{\partial T} \left( \bar{T}_{i}^{n+1} - \bar{T}_{i}^{n} \right) = (\bar{\rho}_{i}^{n-1} u_{i-1}^{n+1} - \bar{\rho}_{i}^{n} u_{i}^{n+1}) \frac{\Delta t}{\Delta x_{j}}.
\] (C.85)

Eliminating the new time temperature using Eq. (C.85) and Eq. (C.58), the cell-face velocities at new time can be obtained as

\[
u_{i}^{n+1} = \omega_{i}^{n} u_{i-1}^{n+1} - \chi_{i}^{n},
\] (C.86)

where

\[
\omega_{i}^{n} = \frac{\bar{p}_{i-1}^{n}}{\bar{p}_{i}^{n}} \left[ 1 - \frac{1}{\bar{\rho}_{i}^{n}} \frac{\partial \rho}{\partial T} \left( \bar{T}_{i-1}^{n} - \bar{T}_{i}^{n} \right) \right],
\] (C.87)

\[
\chi_{i}^{n} = \frac{1}{\bar{p}_{i}^{n}} \frac{\partial \rho}{\partial T} \left( \frac{\bar{q}_{i}^{n} P_{n,j}^n \Delta z_{j}}{\bar{p}_{i}^{n} c_{p,j}^{n} A_{j}} + \frac{\bar{q}_{i}^{n} P_{n,j}^n \Delta z_{j}}{\bar{p}_{i}^{n} c_{p,j}^{n} A_{j}} \right).
\] (C.88)

Substituting the new time velocities \( u_{i}^{n+1} \) and \( u_{i-1}^{n+1} \) for \( i = 2, 3, \ldots, I - 1 \) in Eq. (C.65) into Eq. (C.86), will result in

\[
\omega_{i}^{n} \beta_{i-1}^{n} \bar{P}_{i-1}^{n+1} - \left( \omega_{i}^{n} \beta_{i-1}^{n} + \beta_{i}^{n} \right) \bar{P}_{i}^{n+1} + \beta_{i}^{n} \bar{P}_{i+1}^{n+1} = \chi_{i}^{n} - \omega_{i}^{n} \alpha_{i-1}^{n+1} + \alpha_{i}^{n}, \quad i = 2, 3, \ldots, I,
\] (C.89)

For the outlet pressure boundary condition, \( \bar{P}_{i+1}^{n+1} = P_{out}^{n+1} \), the pressure equation for \( i = I \) can be obtained as

\[
\omega_{i}^{n} \beta_{i-1}^{n} \bar{P}_{i-1}^{n+1} - \left( \omega_{i}^{n} \beta_{i-1}^{n} + \beta_{i}^{n} \right) \bar{P}_{i}^{n+1} = \chi_{i}^{n} - \omega_{i}^{n} \alpha_{i-1}^{n+1} + \alpha_{i}^{n} - \beta_{i}^{n} P_{out}^{n+1}, \quad i = I,
\] (C.90)

Since the density is a function of temperature only, the inlet velocity can be determined for specified inlet flow rate and temperature. Therefore, using a known inlet velocity, the pressure equation for \( i = 1 \) can be derived by inserting \( u_{i}^{n+1} \) in Eq. (C.65) into Eq. (C.77) as

\[
-\beta_{i}^{n} \bar{P}_{i}^{n+1} + \beta_{i}^{n} \bar{P}_{i+1}^{n+1} = \chi_{i}^{n} - \omega_{i}^{n} u_{m}^{n+1} + \alpha_{i}^{n}, \quad i = 1.
\] (C.91)

The coefficient matrix for the system of linear equations for the time advanced pressures given in Eq. (C.89), Eq. (C.90), and Eq. (C.91), is a tri-diagonal matrix. Thus, this system of equations can easily be solved by the forward elimination and backward substitution algorithm. Substitution of the new time pressures into Eq. (C.59) yields the new time velocities. Using the new time velocities, the new time densities and temperatures can be obtained from Eq. (C.57) and Eq. (C.58), respectively.
Using the fully implicit temporal difference scheme, the difference equations for the heat conduction equations in Eq. (C.44) to Eq. (C.46) can be obtained as

\[
\frac{c_{ij,j-1}^{n+1} \Delta t_n}{\rho_{m,ij}^{n+1} c_{p,m,ij}^{n+1}} T_{n,j-1}^{m,i} + \left( 1 + \frac{d_{ij}^{n+1} \Delta t_n}{\rho_{m,ij}^{n+1} c_{p,m,ij}^{n+1}} \right) T_{n,j}^{m,i} - \frac{c_{ij,j+1}^{n+1} \Delta t_n}{\rho_{m,ij}^{n+1} c_{p,m,ij}^{n+1}} T_{n,j+1}^{m,i} = T_{n,j}^{m,i} + \frac{\Delta t_n}{\rho_{m,ij}^{n+1} c_{p,m,ij}^{n+1}} q_{m,ij}^{n+1},
\]

\(2 \leq j \leq J - 1),

\[
\left( 1 + \frac{d_{ij}^{n+1} \Delta t_n}{\rho_{m,ij}^{n+1} c_{p,m,ij}^{n+1}} \right) T_{n,j}^{m,i} - \frac{c_{ij,j-1}^{n+1} \Delta t_n}{\rho_{m,ij}^{n+1} c_{p,m,ij}^{n+1}} T_{n,j-1}^{m,i} = T_{n,j}^{m,i} + \frac{\Delta t_n}{\rho_{m,ij}^{n+1} c_{p,m,ij}^{n+1}} q_{m,ij}^{n+1} + c_{j+1}^{n+1} T_{n,j+1}^{ci} \), \quad (j = 1)
\]

\[
- \frac{c_{ij,j-1}^{n+1} \Delta t_n}{\rho_{m,ij}^{n+1} c_{p,m,ij}^{n+1}} T_{n,j-1}^{m,i} + \left( 1 + \frac{d_{ij}^{n+1} \Delta t_n}{\rho_{m,ij}^{n+1} c_{p,m,ij}^{n+1}} \right) T_{n,j}^{m,i} = T_{n,j}^{m,i} + \frac{\Delta t_n}{\rho_{m,ij}^{n+1} c_{p,m,ij}^{n+1}} q_{m,ij}^{n+1}, \quad (j = J).
\]

The finite difference equations in Eq. (C.92) through Eq. (C.94) form a system of nonlinear equations since the coefficients depend on the unknown temperatures through the temperature dependent material properties. This system of nonlinear equations can be solved iteratively. These coefficients can initially be estimated using an initial guess of temperature. For known coefficients, Eq. (C.92) through Eq. (C.94) form a system of linear equations, which is a tri-diagonal system. This tri-diagonal system can easily be solved by the forward elimination and backward substitution algorithm. Then, the coefficients are updated using the calculated temperatures. This iteration continues until the successive temperatures converge within a specified limit.

C.4. Steady State Solution Methods

The balance equations in Eq. (C.84), Eq. (C.58), and Eq. (C.86) couple the field and flow variables of three meshes \(i - 1, i, \) and \(i + 1,\) and hence they need to be solved simultaneously. This discretization scheme provides a stable time-dependent solution. By setting the time derivatives in Eq. (C.1), Eq. (C.2), and Eq. (C.3) to zero and by applying the finite difference discretization scheme, the steady state equations for an upward flow in a constant flow area can be derived as

\[
\bar{T}_i = \left( 1 + \frac{P_{h,i} \Delta z h_{f,i}}{\bar{c}_{p,i} \bar{r}_{i-1} u_{i-1} A_i} \right)^{-1} \left( \bar{T}_{i-1} + \frac{\bar{q}_i}{\bar{c}_{p,i} \bar{r}_{i-1} u_{i-1} A_i} + \frac{P_{h,i} \Delta z h_{f,i}}{\bar{c}_{p,i} \bar{r}_{i-1} u_{i-1} A_i} \bar{T}_i \right),
\]

\(i = (C.95)\)

\[
\rho_i = \rho(T),
\]

\(i = (C.96)\)

\[
u_i = \frac{\bar{T}_{i-1}}{\bar{\rho}_i} u_{i-1},
\]

\(i = (C.97)\)
\[
\overline{P}_i = \overline{P}_{i+1} - \left[ \left( \frac{\rho_i^2}{\rho_{i-1}} u_{i-1}^2 - \frac{\rho_i^2}{\rho_{i+1}} u_i^2 \right) - \left( \frac{\rho_i \Delta z_i + \rho_{i+1} \Delta z_{i+1}}{2} \right) g - \frac{f_i \overline{P}_i \Delta z_i}{2D_h} \right]. \tag{C.98}
\]

The steady state solution scheme is started by solving Eq. (C.95) for the temperature and then obtaining the density and velocity in each node by solving Eq. (C.96) and Eq. (C.97) respectively, until the end of the channel. The pressure in each node is solved using Eq. (C.98) starting from the end of the channel to the channel inlet, node by node.

The steady state heat condition in the moderator can be derived by setting the time derivative in Eq. (C.44) through Eq. (C.46) to zero and neglecting the axial conduction as

\[
-c_{ij,j-1}T_{i,j-1} + d_{ij}T_{i,j} - c_{ij,j+1}T_{i,j+1} = \overline{q}_{ij}^m, \quad (2 \leq j \leq J-1), \tag{C.99}
\]

\[
d_{ij}T_{i,j} - c_{ij,j+1}T_{i,j+1} = c_{ij}T_{ci} + \overline{q}_{ij}^m, \quad (j = 1), \tag{C.100}
\]

\[
-c_{ij,j-1}T_{i,j-1} + d_{ij}T_{i,j} = \overline{q}_{ij}^m, \quad (j = J). \tag{C.101}
\]

The finite difference equations in Eq. (C.99) through Eq. (C.101) form a system of nonlinear equations since the coefficients depend on the unknown temperatures through the temperature dependent material properties. This system of equations forms a tri-diagonal system that can easily be solved by the forward elimination and backward substitution algorithm.

C.5. Flow Splitting Algorithm

It is assumed that the total fuel salt flow is distributed among multiple parallel fuel channels that have common inlet and outlet. The mass flows in individual fuel channels are calculated using the equal pressure drop boundary conditions over all channels. For given mass flux \(G_i\), the pressure drop \(\Delta P_i\) across the fuel channel \(i\) is determined by solving the single channel thermal-hydraulics problem. Denoting the pressure drop as a function of the mass flux as \(\Delta P_i(G_i)\), the system of governing equations for unknown mass fluxes for \(I\) parallel fuel channels can be written as the following system of nonlinear equations

\[
\Delta P_1(G_1) = \Delta P_2(G_2) = \ldots = \Delta P_I(G_I) = \Delta P, \tag{C.102}
\]

\[
\sum_{i=1}^{I} N_i A_j G_j = W_T. \tag{C.103}
\]
where \( N_i \) is the number of coolant tubes, \( A_i \) is the flow area of channel \( i \), and \( W_i \) is the total flow rate. The system of nonlinear equations given in Eq. (C.102) and Eq. (C.103) can be solved iteratively using the Newton-Rapson method. The derivatives required in the Newton-Rapson method can be estimated by the finite difference approximation based on the single channel solutions for given mass fluxes. Using the pressure drop \( \Delta P_i^{(n)} \) calculated with the \( n \)-th iterative solution \( G_i^{(n)} \), the pressure drop for the \((n+1)\)-st solution can be approximated as

\[
\Delta P_i^{(n+1)} = \Delta P_i^{(n)} + a_i^{(n)} \left( G_i^{(n+1)} - G_i^{(n)} \right), \quad i = 1, 2, \ldots, I, \tag{C.104}
\]

where

\[
a_i^{(n)} = \left. \frac{d (\Delta P_i)}{d (G_i)} \right|_{G_i = G_i^{(n)}} = \frac{\Delta P_i^{(n)} - \Delta P_i^{(n-1)}}{G_i^{(n)} - G_i^{(n-1)}}, \tag{C.105}
\]

\[
\Delta P_i^{(n)} = \Delta P_i \left( G_i^{(n)} \right). \tag{C.106}
\]

Requiring \( \Delta P_i^{(n+1)} \) to be equal to average pressure drop \( \overline{\Delta P}^{(n+1)} \), Eq. (C.104) can be rewritten as

\[
a_i^{(n)} G_i^{(n+1)} - \overline{\Delta P}^{(n+1)} = a_i^{(n)} G_i^{(n)} - \Delta P_i^{(n)}, \quad i = 1, 2, \ldots, I. \tag{C.107}
\]

Dividing Eq. (C.107) by \( a_i^{(n)} \), multiplying by \( N_i A_i \), and summing up the resulting equations, then

\[
\sum_{i=1}^{I} N_i A_i G_i^{(n+1)} - \sum_{i=1}^{I} N_i A_i \overline{\Delta P}^{(n+1)} = \sum_{i=1}^{I} N_i A_i G_i^{(n)} - \sum_{i=1}^{I} N_i A_i \Delta P_i^{(n+1)} \frac{a_i^{(n)}}{a_i^{(n)}}, \tag{C.108}
\]

Requiring the mass fluxes \( G_i^{(n)} \) and \( G_i^{(n+1)} \) to satisfy the flow conservation equation in Eq. (C.103), the average core pressure drop of the next iteration can be determined as

\[
\overline{\Delta P}^{(n+1)} = \sum_{i=1}^{I} N_i A_i \Delta P_i^{(n)} \frac{a_i^{(n)}}{a_i^{(n)}} \left/ \sum_{i=1}^{I} N_i A_i \frac{a_i^{(n)}}{a_i^{(n)}} \right., \tag{C.109}
\]

Using this average pressure drop, the mass fluxes of the next iteration are obtained as
\[ G_i^{(n+1)} = G_i^{(n)} + \frac{1}{a_i^{(n)}} \left( \overline{\Delta P}^{(n+1)} - \Delta P_i^{(n)} \right), \quad i = 1, 2, \ldots, I. \] (C.110)

This iteration procedure continues until the mass fluxes of two subsequent iterations converge within a specified limit. Specifically, the channel flow rates are determined to satisfy the constant pressure drop boundary condition. Initial guess is made for each channel mass flux and the core-average pressure drop using the values at the previous time step. For the initial steady state, a uniform initial mass flux \( G_i^{(0)} = W_T / \sum_{i=1}^I N_i A_i \) and the core average pressure drop \( \overline{\Delta P}^{(0)} \) is used. Using the assumed mass flux \( G_i^{(0)} \), the channel pressure drop \( \Delta P_i^{(0)} \) is computed for each channel.

For a perturbed mass flux \( G_i^{(1)} = (1 + \varepsilon_G) G_i^{(0)} \), the perturbed channel pressure drop \( \Delta P_i^{(1)} \) is computed. Here \( \varepsilon_G \) is the user-specified parameter for the flow rate perturbation used to evaluate the pressure derivatives initially. Using these perturbed pressure drops, the pressure derivatives with respect to the mass flux and the core-average pressure drop are computed as

\[ a_i^{(1)} \approx a_i^{(2)} \approx \frac{\Delta P_i^{(2)} - \Delta P_i^{(1)}}{\varepsilon_G G_i^{(1)}}, \] (C.111)

\[ \overline{\Delta P}^{(2)} = \sum_{i=1}^I A_i \Delta P_i^{(1)} \sum_{i=1}^I a_i^{(1)}. \] (C.112)

The \((n+1)\)-st iteration of each channel mass flux \( G_i^{(n+1)} \) is determined to satisfy the constant pressure drop boundary conditions approximately as

\[ G_i^{(n+1)} = G_i^{(n)} + \frac{1}{a_i^{(n)}} \left( \overline{\Delta P}^{(n)} - \Delta P_i^{(n)} \right). \] (C.113)

If the summation of individual channel flow rates is equal to the core flow rate, that is, \( \sum_{i=1}^I N_i A_i G_i^{(n+1)} - W_T \mid W_T < \varepsilon_w \), then the constant pressure drop boundary condition is satisfied. The iteration for channel mass fluxes is stopped, and the single channel analysis is performed for each channel. Otherwise, the updated channel pressure drops \( \Delta P_i^{(n+1)} \) are
calculated with the channel flow rates $G_i^{(n+1)}$ and the pressure derivatives with respect to mass flux are updated as

$$a_i^{(n+1)} = \frac{\Delta P_i^{(n+1)} - \Delta P_i^{(n)}}{G_i^{(n+1)} - G_i^{(n)}}.$$  \hspace{1cm} (C.114)

and back to Eq. (C.113) until it satisfies the convergence criteria.

C.6. Primary Loop Thermal-Fluidic Model

In the reactor system, the fuel salt exiting the core flows through the primary system before returning to the core inlet. To describe the core behavior during transients more accurately, the heat exchanger in the primary system is also included in the thermal-hydraulics model of MSR. The heat exchanger has been modeled with a lumped parameter approach to simulate the heat removal by the intermediate loop, considering it as one node. As shown in Fig. C.4, the fuel salt temperatures at the core inlet and outlet are directly coupled with the fuel salt temperatures at the heat exchanger outlet and inlet, respectively. The other components of the primary loop such as primary pumps, inlet and outlet plenums are not considered in this model.

Fig. C.4. Schematic diagram of the heat removal from the primary system of MSFR.
Assuming the material temperature variations in the heat exchanger during a transient are proportional to the steady state temperatures, the heat exchanger can be characterized by the average fuel salt temperature $\bar{T}_{HX}$, of which variation is determined by

$$C_{HX} \frac{dT_{HX}}{dt} = \dot{m}_{HX} c_p \left( T_{in}^{HX} - T_{out}^{HX} \right) + \dot{Q}_{\text{decay}} - \dot{Q}_{HX}. \quad (C.115)$$

where $C_{HX} = m_{HX} c_p$ is the heat capacity of the heat exchanger, $c_p$ is the specific heat of the fuel salt at the corresponding temperature, $\dot{m}_{HX}$ is the fuel salt mass flow rate, and $m_{HX}$ is the mass of the heat exchanger. $T_{in}^{HX}$ and $T_{out}^{HX}$ are respectively the fuel salt temperatures at the inlet and outlet of the heat exchanger, $\dot{Q}_{\text{decay}}$ is the decay heat generated by the fuel salt in the heat exchanger, and $\dot{Q}_{HX}$ is the heat removal rate by the intermediate fluid in the heat exchanger, and it can be written as

$$\dot{Q}_{HX} = UA(\bar{T}_{HX} - \bar{T}_{IF}). \quad (C.116)$$

where $\bar{T}_{IF}$ is the average intermediate fluid temperature, $U$ is the overall heat transfer coefficient from the fuel salt to the intermediate fluid, and $A$ is total heat transfer area in the heat exchanger.

The overall heat transfer coefficient $U$ is evaluated as

$$\frac{1}{AU} = \left[ \frac{1}{Ah_c} + \frac{1}{A_h h_h} + R_w \right]^{-1}, \quad (C.117)$$

where $h_c$ is the heat transfer coefficient of the secondary side of the heat exchanger, $R_w$ is the thermal resistance of the heat exchanger wall, $A_h$ is the total transfer area of the hot fuel salt, and $h_h$ is the heat transfer coefficient of the fuel salt in the primary side of the heat exchanger. The primary side heat transfer coefficient is evaluated from the Gnielinski correlation for the Nusselt number [33]
\[
\text{Nu} = \frac{(f/8)(\text{Re} - 1000)\text{Pr}}{1 + 12.7(f/8)^{0.5} \left(\text{Pr}^{2/3} - 1\right)}. \tag{C.118}
\]

where \(f\) is the friction factor in Eq. (C.5). For the secondary side heat transfer coefficient, it is assumed to be constant and it is evaluated from the steady state calculations. Using the upwind scheme for spatial differencing with \(T_{in}^{HX} = T_{c}^{in}\), \(T_{out}^{HX} = T_{c}^{out}\) and \(T_{in}^{c} = T_{out}^{c}\) and the explicit scheme for time differencing, the finite difference equation for heat exchanger can be derived as

\[
\overline{T}_{HX}^n = \overline{T}_{HX}^{n-1} + \frac{m_{HX}^{n-1}c_p^{n-1}}{C_{HX}}(T_{in}^{n-1} - T_{out}^{n-1}) \Delta t_n - \frac{\Delta t_n}{C_{HX}} \left(\dot{Q}_{HX}^{n-1} - \dot{Q}_{decay}^{n-1}\right). \tag{C.119}
\]

Eq. (C.119) can be rewritten with the initial condition as

\[
\overline{T}_{HX}^n = \left[1 - \frac{m_{HX}^{n-1}c_p^{n-1}}{C_{HX}} \Delta t_n\right] \overline{T}_{HX}^{n-1} + \frac{m_{HX}^{n-1}c_p^{n-1}}{C_{HX}} T_{in}^{n-1} - \frac{\Delta t_n}{C_{HX}} \left(\dot{Q}_{HX}^{n-1} - \dot{Q}_{decay}^{n-1}\right), \tag{C.120}
\]

\[
\overline{T}_{HX}^0 = T_{out}^c - \frac{\dot{Q}_{HX}^0}{m_{HX}^0c_p^0}. \tag{C.121}
\]

C.7. Decay Heat Model

The decay heat equation for a flowing fuel is different from that of a stationary fuel. In an MSR, the decay heat can be released outside the core, especially in the heat exchanger, and it should be considered in calculating the power. A simple decay heat model has been implemented into the PROTEUS-NODAL code, where the fission products are grouped into a few decay heat precursor groups \(K_D\) and for each group, the following decay heat equation is solved

\[
\frac{\partial}{\partial t} h_k(\vec{r}, t) + \nabla \cdot \left[\vec{u}(\vec{r}, t)h_k(\vec{r}, t)\right] + \lambda_k h_k(\vec{r}, t) = f_k Q_{tot}(\vec{r}, t), \quad k = 1, 2, ..., K_D. \tag{C.122}
\]

where \(h_k\) is the product of the decay heat precursor concentration and energy release in decay heat group \(k\), \(f_k\) is the decay heat fraction in group \(k\), \(\lambda_k\) is the decay constant for group \(k\), and \(Q_{tot}\) is the total volumetric heat source produced in the core as defined in Eq. (C.127). Considering axial velocity field, then Eq. (C.122) reduces to
This equation can be solved in a similar way as the delayed neutron precursor equation analytical, semi-analytical, or numerical schemes. For the decay heat in the outer loop, there is no heat generated from fission, so the following equation is considered

\[
\frac{\partial}{\partial t} h_k(r, z, t) + \frac{\partial}{\partial z} \left[ u(z) h_k(r, z, t) \right] + \lambda_k h_k(r, z, t) = f_k Q_{\text{ex}}(r, z, t). \tag{C.123}
\]

This equation should be solved to provide a heat source for the heat exchanger. The fission, decay, and total volumetric heat sources in the core can be expressed as

\[
Q_{\text{decay}}(r, z, t) = \sum_{k=1}^{K_{\text{d}}} \lambda_k h_k(r, z, t), \tag{C.125}
\]

\[
Q_{\text{fiss}}(r, z, t) = \left[ 1 - \sum_{k=1}^{K_{\text{d}}} f_k \right] \sum_{g=1}^{G} \phi_{fg}^r(z, t) \phi_g^r(r, z, t), \tag{C.126}
\]

\[
Q_{\text{tot}}(r, z, t) = \sum_{g=1}^{G} \phi_{fg}^r(z, t) \phi_g^r(r, z, t) = Q_{\text{fiss}}(r, z, t) + Q_{\text{decay}}(r, z, t) \tag{C.127}
\]

where \( \varepsilon \) is the recoverable energy release per fission.
Appendix D. PROTEUS-Nodal Input Data for MSR Analysis

Various input cards have been introduced to PROTEUS-NODAL for MSR analyses. Table D.1 describes the added input cards to the input driver file. An input card for flowing fuel options and a card for the standalone thermal-hydraulics calculation options are provided.

Table D.1. Input Cards in Driver Input File

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Input Data</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Flowing Fuel Options</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USE_FLOWINGFUEL</td>
<td>YES</td>
<td>NO</td>
<td>Activate the flowing fuel option</td>
</tr>
<tr>
<td>TIME_OUT_CORE</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Transit time outside the core (s)</td>
</tr>
<tr>
<td><strong>MSR Thermal-Hydraulics Options</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USE_MSR_TH</td>
<td>YES</td>
<td>NO</td>
<td>Activate MSR TH calculation. Two XS files must be provided for XS interpolation</td>
</tr>
<tr>
<td>SourceFile_MSRTH</td>
<td>&lt;Max 128 Characters&gt;</td>
<td>-</td>
<td>Unix file path to a MSR TH input file</td>
</tr>
<tr>
<td>SOURCEFILE_XS_TEMP1</td>
<td>&lt;Max 128 Characters&gt;</td>
<td>-</td>
<td>Unix file path to a cross section data file at low temperature for XS interpolation</td>
</tr>
<tr>
<td>SOURCEFILE_XS_TEMP2</td>
<td>&lt;Max 128 Characters&gt;</td>
<td>-</td>
<td>Unix file path to a cross section data file at high temperature for XS interpolation</td>
</tr>
</tbody>
</table>

The key words added to the geometry input for the variational nodal P_1 solver in R-θ-Z geometry are described in Table D.2. The boundary condition for the radial boundary surface is specified with the keyword ‘BOUNDARY_CONDITION’ with input ‘R’. The boundary conditions for the azimuthal boundary surfaces are specified with the keyword ‘BOUNDARY_CONDITION’ with input ‘-T’ or ‘+T’, where ‘+’ is for the boundary of larger azimuthal angle and ‘-’ is for the boundary of smaller azimuthal angle. ‘EXTRAPOLATED’ and ‘VACUUM’ can also be given in the θ direction. If the ‘BOUNDARY_CONDITION’ keyword is not specified, the periodic boundary condition is automatically applied in the θ direction. Note that for the periodic boundary condition, the total span of the θ grid specified in the ‘TGRID’ keyword should be one of the factors of 360.
Table D.2. Keywords Added to Geometry Input for R-θ-Z Geometry

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Input Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOUNDERY_CONDITION</td>
<td>R B.C.</td>
<td>B.C. = EXTRAPOLATED or VACUUM</td>
</tr>
<tr>
<td>BOUNDERY_CONDITION</td>
<td>-T B.C.</td>
<td></td>
</tr>
<tr>
<td>BOUNDERY_CONDITION</td>
<td>+T B.C.</td>
<td></td>
</tr>
<tr>
<td>RGRID</td>
<td>( r_1 ) ( r_2 ) ( n_r )</td>
<td>( r_1 ) = inner radius in cm ( r_2 ) = outer radius in cm ( n_r ) = # of nodes between ( r_1 ) and ( r_2 )</td>
</tr>
<tr>
<td>TGRID</td>
<td>( \theta_1 ) ( \theta_2 ) ( n_\theta )</td>
<td>( \theta_1 ) = azimuthal angle at ‘-T’ boundary ( \theta_2 ) = azimuthal angle at ‘+T’ boundary ( n_\theta ) = # of nodes between ( \theta_1 ) and ( \theta_2 )</td>
</tr>
</tbody>
</table>

Table D.3 describes the added input cards to the material assignment file. This card specifies the flowing fuel velocity or the region to thermal-hydraulics channel. Table D.4 describes the input cards for the standalone thermal-hydraulics calculation. This includes the cards for general thermal-hydraulics properties of the core, the cards for cross section functionalization, the cards for heat exchanger parameters, the card for decay heat calculation model, the cards for heat conduction model in solid moderator, and the cards to activate transient scenarios.

Table D.3. Input Cards in Material Assignment File

<table>
<thead>
<tr>
<th>REGION_PROPERTY 1st</th>
<th>2nd</th>
<th>3rd</th>
<th>Input Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGION_PROPERTY</td>
<td>1st</td>
<td>Value</td>
<td>&lt;Character Value&gt;</td>
<td>Define a region to assign properties</td>
</tr>
<tr>
<td></td>
<td>2nd</td>
<td>Value</td>
<td>&lt;Character Value&gt;</td>
<td>Define the MSR property to be assigned (VELOCITY, FUELFLOW)</td>
</tr>
<tr>
<td></td>
<td>3rd</td>
<td>Value</td>
<td>&lt;Real Value&gt;</td>
<td>Assign a value to selected property (real value for VELOCITY and 0.0 or 1.0 for FUELFLOW)</td>
</tr>
<tr>
<td>Note</td>
<td>VELOCITY</td>
<td>Required for regular MSR calculations to assign a velocity value to each region (any real value can be specified) (m/s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FUELFLOW</td>
<td>Required for MSR thermal-hydraulics calculations for region to flow channel mapping (1 to be included or 0 to be excluded in the core thermal-hydraulics calculations)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table D.4. Input Cards in Standalone Thermal-Hydraulics Input File

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Input Data</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Core Parameters</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL_POWER</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Thermal power (W) for flux and power normalization. It should be equal to the value of THERMAL_POWER in the driver input file.</td>
</tr>
<tr>
<td>INLET_TEMPERATURE</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Core inlet temperature (°C)</td>
</tr>
<tr>
<td>OUTLET_PRESSURE</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Core outlet pressure (MPa)</td>
</tr>
<tr>
<td>MASS_FLOWRATE</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Inlet mass flow rate (kg/s)</td>
</tr>
<tr>
<td>INLET_VELOCITY</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Inlet salt velocity (m/s)</td>
</tr>
<tr>
<td>INITIAL_SALT_DENSITY</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Inlet fuel density (kg/m³)</td>
</tr>
<tr>
<td>INLET_FLOWAREA</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Inlet flow area (m²)</td>
</tr>
<tr>
<td>ASSEMBLY_PITCH</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Assembly pitch (cm)</td>
</tr>
<tr>
<td>ACTIVE_FUEL_HEIGHT</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Active fuel height (cm)</td>
</tr>
<tr>
<td><strong>Temperature of Cross Sections Files</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER_OF_XS_TEMPS</td>
<td>&lt;Integer Value&gt;</td>
<td>2</td>
<td>Number of temperatures for XS interpolation</td>
</tr>
<tr>
<td>TEMPERATURE_1</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Temperature for the first XS data file (°C)</td>
</tr>
<tr>
<td>TEMPERATURE_2</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Temperature for the second XS data file (°C)</td>
</tr>
<tr>
<td><strong>Heat Exchanger Parameters</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTERLOOP_HX</td>
<td>YES NO</td>
<td>NO</td>
<td>Activate the outer loop model for heat exchanger calculations</td>
</tr>
<tr>
<td>HX_SALT_VELOCITY</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Fuel salt velocity in heat exchanger (m/s)</td>
</tr>
<tr>
<td>HX_TUBE_DIAMETER</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Tube diameter of heat exchanger (m)</td>
</tr>
<tr>
<td>HX_TUBE_THICKNESS</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Tube thickness of heat exchanger (m)</td>
</tr>
<tr>
<td>HX_LENGTH</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Tube length of heat exchanger (m)</td>
</tr>
<tr>
<td>HX_NUMBER_OF_TUBES</td>
<td>&lt;Integer Value&gt;</td>
<td>-</td>
<td>Number of tubes in heat exchanger</td>
</tr>
<tr>
<td>HX_INTERMEDIATELOOP</td>
<td>YES NO</td>
<td>NO</td>
<td>Activate the intermediate loop model for heat exchanger calculations</td>
</tr>
<tr>
<td>IT_INLET_TEMP</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Inlet coolant temperature in intermediate loop (°C)</td>
</tr>
<tr>
<td><strong>Decay Heat Calculation</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECAY_HEAT</td>
<td>YES NO</td>
<td>NO</td>
<td>Activate decay heat</td>
</tr>
<tr>
<td>NO</td>
<td>calculations model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>---------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Choose Transient Scenarios</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UPOS</td>
<td>YES NO</td>
<td>NO</td>
<td>Activate unprotected pump over-speed transient</td>
</tr>
<tr>
<td>ULOF</td>
<td>YES NO</td>
<td>NO</td>
<td>Activate unprotected loss of flow transient</td>
</tr>
<tr>
<td>ULOHS</td>
<td>YES NO</td>
<td>NO</td>
<td>Activate unprotected loss of heat sink transient</td>
</tr>
<tr>
<td>UFSOC</td>
<td>YES NO</td>
<td>NO</td>
<td>Activate unprotected chilled inlet transient</td>
</tr>
<tr>
<td>Time Constant</td>
<td>&lt;Real Value&gt;</td>
<td>-</td>
<td>Transient time constant (s)</td>
</tr>
<tr>
<td>Heat Conduction in Solid Moderator</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOLID_MODERATOR</td>
<td>YES NO</td>
<td>NO</td>
<td>Activate the heat conduction in solid moderator and the flow splitting among parallel fuel channels</td>
</tr>
<tr>
<td>NATURAL_CIRCULATION</td>
<td>YES NO</td>
<td>NO</td>
<td>Use the Nusselt number correlation for natural circulation</td>
</tr>
<tr>
<td>MODERATOR_ISOTOPE</td>
<td>&lt;Character Value&gt;</td>
<td>-</td>
<td>Solid moderator isotope for thermal feedback calculation</td>
</tr>
<tr>
<td>FUEL_FRACTION_CH</td>
<td>&lt;Real Value&gt;</td>
<td>1.0</td>
<td>Fuel salt volume fraction in a unit cell</td>
</tr>
<tr>
<td>MODERATOR_FRACTION_CH</td>
<td>&lt;Real Value&gt;</td>
<td>0.0</td>
<td>Moderator volume fraction in a unit cell</td>
</tr>
<tr>
<td>NUMBER_OF_RADIAL_MESHES</td>
<td>&lt;Integer Value&gt;</td>
<td>-</td>
<td>Number of radial meshes for heat conduction calculation</td>
</tr>
<tr>
<td>MOD_GAMMA_HEAT_FACT</td>
<td>&lt;Real Value&gt;</td>
<td>0.0</td>
<td>Fraction of heat deposit in the solid moderator due to gamma heating</td>
</tr>
</tbody>
</table>
Figures D.1, D.2, and D.3 show sample input files of the driver input, assignment input, and standalone thermal-hydraulics calculation input, respectively.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>THERMAL_POWER</td>
<td>3.0000E+09</td>
</tr>
<tr>
<td>MAX_FISSION_ITER</td>
<td>500</td>
</tr>
<tr>
<td>EIGENVALUE_GUESS</td>
<td>1.0</td>
</tr>
<tr>
<td>TOLERANCE_EIGENVALUE</td>
<td>1.0e-5</td>
</tr>
<tr>
<td>TOLERANCE_FISSION</td>
<td>1.0e-5</td>
</tr>
<tr>
<td>TOLERANCE_FLUX</td>
<td>1.0e-5</td>
</tr>
<tr>
<td>SOURCEFILE_XS</td>
<td>MSFR-U233.ISOTXS</td>
</tr>
<tr>
<td>SOURCEFILE_DELAY</td>
<td>MSFR-U233.DLAYXS</td>
</tr>
<tr>
<td>SOURCEFILE_MATERIAL</td>
<td>MSFR_U233.assignment</td>
</tr>
<tr>
<td>SOURCEFILE_GRID</td>
<td>MSFR_U233.grid</td>
</tr>
<tr>
<td>USE_TRANSPORT</td>
<td>YES</td>
</tr>
<tr>
<td>USE_CMFD_ACCEL</td>
<td>YES</td>
</tr>
<tr>
<td>CMFD_SOLVER_BACKVECTORS</td>
<td>20</td>
</tr>
<tr>
<td>CMFD_UNDERRELAXATION</td>
<td>0.5</td>
</tr>
<tr>
<td>EXPORT_SOLUTION_VTK</td>
<td>YES</td>
</tr>
<tr>
<td>USE_FLOWINGFUEL</td>
<td>YES</td>
</tr>
<tr>
<td>TIME_OUT_CORE</td>
<td>2.0</td>
</tr>
<tr>
<td>USE_MSR_TH</td>
<td>YES</td>
</tr>
<tr>
<td>SourceFile_MSRTH</td>
<td>MSFR_TH_Input.TH</td>
</tr>
<tr>
<td>SOURCEFILE_XS_TEMP1</td>
<td>MSFR-U233-900K.ISOTXS</td>
</tr>
<tr>
<td>SOURCEFILE_XS_TEMP2</td>
<td>MSFR-U233-1200K.ISOTXS</td>
</tr>
</tbody>
</table>

Fig. D.1. Example of a driver input file.
Material Definition

```plaintext
MATERIAL_DEF F0101 LI7IC1 2.139E-02
MATERIAL_DEF F0101 F-9IC1 4.623E-02
(cut)
MATERIAL_DEF BLNKT F-9BLK 4.651E-02
MATERIAL_DEF BLNKT TH2BLK 6.248E-03
```

Region-Material Map

```plaintext
REGION_ALIAS IC0101 F0101
REGION_ALIAS IC0102 F0102
(cut)
REGION_ALIAS BK007 BLNKT
REGION_ALIAS BK008 BLNKT
```

Region Property

```plaintext
REGION_PROPERTY IC0101 ATOM_DENSITY 7.383E-02
REGION_PROPERTY IC0102 ATOM_DENSITY 7.383E-02
(cut)
REGION_PROPERTY BK007 ATOM_DENSITY 7.4284E-02
REGION_PROPERTY BK008 ATOM_DENSITY 7.428E-02

REGION_PROPERTY IC0101 VELOCITY 1.1200
REGION_PROPERTY IC0102 VELOCITY 1.1200
(cut)
REGION_PROPERTY BK007 VELOCITY 0.0000
REGION_PROPERTY BK008 VELOCITY 0.0000

REGION_PROPERTY IC0101 FUELFLOW 1.0
REGION_PROPERTY IC0102 FUELFLOW 1.0
(cut)
REGION_PROPERTY BK007 FUELFLOW 0.0
REGION_PROPERTY BK008 FUELFLOW 0.0
```

Fig. D.2. Example of a material assignment file.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL_POWER</td>
<td>3.00E+09</td>
</tr>
<tr>
<td>INLET_TEMPERATURE</td>
<td>650.0</td>
</tr>
<tr>
<td>OUTLET_PRESSURE</td>
<td>0.101325</td>
</tr>
<tr>
<td>MASS_FLOWRATE</td>
<td>1.8772E+04</td>
</tr>
<tr>
<td>INLET_VELOCITY</td>
<td>1.1275</td>
</tr>
<tr>
<td>INITIAL_SALT_DENSITY</td>
<td>4168.84</td>
</tr>
<tr>
<td>INLET_FLOWAREA</td>
<td>3.9938</td>
</tr>
<tr>
<td>ASSEMBLY_PITCH</td>
<td>22.5115</td>
</tr>
<tr>
<td>ACTIVE_FUEL_HEIGHT</td>
<td>225.500</td>
</tr>
<tr>
<td>NUMBER_OF_XS_TEMPS</td>
<td>2</td>
</tr>
<tr>
<td>TEMPERATURE_1</td>
<td>626.85</td>
</tr>
<tr>
<td>TEMPERATURE_2</td>
<td>926.85</td>
</tr>
<tr>
<td>SOLID_MODERATOR</td>
<td>YES</td>
</tr>
<tr>
<td>MODERATOR_ISOTOPE</td>
<td>C____7</td>
</tr>
<tr>
<td>NATURAL_CIRCULATION</td>
<td>NO</td>
</tr>
<tr>
<td>FUEL_FRACTION_CH</td>
<td>0.2229</td>
</tr>
<tr>
<td>MODERATOR_FRACTION_CH</td>
<td>0.7771</td>
</tr>
<tr>
<td>NUMBER_OF_RADIAL_MESHES</td>
<td>5</td>
</tr>
<tr>
<td>MOD_GAMMA_HEAT_FACT</td>
<td>0.05</td>
</tr>
<tr>
<td>OUTERLOOP_HX</td>
<td>YES</td>
</tr>
<tr>
<td>HX_NUMBER_OF_TUBES</td>
<td>8061</td>
</tr>
<tr>
<td>HX_SALT_VELOCITY</td>
<td>2.780</td>
</tr>
<tr>
<td>HX_TUBE_DIAMETER</td>
<td>0.004</td>
</tr>
<tr>
<td>HX_TUBE_THICKNESS</td>
<td>0.000</td>
</tr>
<tr>
<td>HX_LENGTH</td>
<td>2.000</td>
</tr>
<tr>
<td>HX_INTERMEDIATE_LOOP</td>
<td>YES</td>
</tr>
<tr>
<td>IT_INLET_TEMP</td>
<td>550.00</td>
</tr>
<tr>
<td>DECAY_HEAT</td>
<td>NO</td>
</tr>
<tr>
<td>ULOF</td>
<td>NO</td>
</tr>
<tr>
<td>UPOS</td>
<td>YES</td>
</tr>
<tr>
<td>ULOHS</td>
<td>NO</td>
</tr>
<tr>
<td>UFSOC</td>
<td>NO</td>
</tr>
<tr>
<td>TRANSIENT_TIME_CONSTANT</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Fig. D.3. Example of an assignment file for standalone thermal-hydraulics calculation.
Appendix E. Input Data for TreeFrog and TreeKangaroo Applications

The applications “TreeFrog” and “TreeKangaroo” use the MOOSE style block-structured input file format. Each block is identified with square brackets. A block can contain multiple sub-blocks. Each sub-block starts with “[./sub-block name]” and ends with “[../]”. Within each block or sub-block, the line inputs are given as pairs of parameter and value with an equal sign between them. The input data blocks are described in the following subsections, and then sample input files are presented.

E.1. Description of Input Data Blocks

E.1.1. Mesh

The Mesh block is used to generate the mesh to be consistent with the SAM mesh. Table E.1 describes the input parameters of the mesh block. The TreeKangaroo and TreeFrog meshes are generated through the MOOSE function “GeneratedMesh”. The mesh dimension is 1 if there is only 1 channel in the SAM input. If the number of channels is greater than 1, the mesh dimension is 2. The parameter “nx” is the number of axial elements in the core channels in the SAM input. The parameter “ny” is the number of core channels in the SAM input minus one.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Input Data</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Mesh]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>GeneratedMesh</td>
<td>-</td>
<td>MOOSE function for mesh generation</td>
</tr>
<tr>
<td>dim</td>
<td>1</td>
<td>-</td>
<td>1 if the SAM input has only one channel. Otherwise, 2.</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nx</td>
<td>unsigned int</td>
<td>-</td>
<td>Number of axial elements in core channels in the SAM input</td>
</tr>
<tr>
<td>ny</td>
<td>unsigned int</td>
<td>-</td>
<td>Only used when dim = 2. Number of core channels in the SAM input minus one</td>
</tr>
</tbody>
</table>

E.1.2. AuxVariables

The AuxVariables (auxiliary variable) block is used to store the power density, temperature, velocity, and density in the TreeKangaroo and TreeFrog meshes. Table E.2 describes the input parameters of the auxiliary variable block. In the MOOSE framework, auxiliary variables can be defined in Element or Nodal. However, in the TreeKangaroo and TreeFrog meshes, only the first order nodal auxiliary variables are utilized. Each auxiliary variable is defined in a sub-block and the name of the sub-block is the name of this auxiliary variable. In TreeKangaroo and TreeFrog, at least four auxiliary variables need to be defined to store the power density, temperature, velocity, and density data in each application. To transfer the data between two applications, the auxiliary variable defined in other application need to be listed as a sub-block if data is transferred from this auxiliary variable. In the sub-
block for this kind of auxiliary variable, only [./sub-block name] and [../] are specified and no detailed parameter is needed.

Table E.2. Input Description of AuxVariables Block

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Input Data</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[AuxVariables]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[./AuxVariables1]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>order</td>
<td>FIRST</td>
<td>-</td>
<td>The first order auxiliary variable is used</td>
</tr>
<tr>
<td>family</td>
<td>LAGRANGE</td>
<td>-</td>
<td>The nodal auxiliary variable is used</td>
</tr>
<tr>
<td>initial_condition</td>
<td>double</td>
<td>-</td>
<td>Initial value for this variable, only used in steady state simulations</td>
</tr>
<tr>
<td>[../]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

E.1.3. Function

The Function block is used to define functions depending on spatial position and time only. It is widely used in defining initial conditions, boundary conditions, and other quantities in MOOSE applications. Table E.3 shows two example functions: a function created by a mathematical expression and a piecewise constant function. Other available function forms can be found in the MOOSE website.

Table E.3. Input Description of Function Block

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Input Data</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Function]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[./ParsedFunction]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>ParsedFunction</td>
<td>-</td>
<td>Function created by a mathematical expression</td>
</tr>
<tr>
<td>value</td>
<td>LAGRANGE</td>
<td>-</td>
<td>User defined function depending on t, x, y, z</td>
</tr>
<tr>
<td>[../]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[./PiecewiseConstant]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>PiecewiseConstant</td>
<td>-</td>
<td>Defines data using a set of x-y data pairs</td>
</tr>
<tr>
<td>axis</td>
<td>x y z</td>
<td>-</td>
<td>The axis used (x, y, or z) if this is to be a function of position</td>
</tr>
<tr>
<td>data_file</td>
<td>FileName</td>
<td>-</td>
<td>File holding comma-separated value (csv) data for use with Piecewise</td>
</tr>
<tr>
<td>format</td>
<td>columns rows</td>
<td>-</td>
<td>Format of csv data file that is in either in columns or rows</td>
</tr>
<tr>
<td>scale_factor</td>
<td>double</td>
<td>-</td>
<td>Scale factor to be applied to the ordinate values</td>
</tr>
<tr>
<td>x</td>
<td>vector</td>
<td>-</td>
<td>The abscissa values</td>
</tr>
<tr>
<td>x_index_in_file</td>
<td>unsigned int</td>
<td>0</td>
<td>The abscissa index in the data file</td>
</tr>
<tr>
<td>xy_data</td>
<td>vector</td>
<td>-</td>
<td>All function data, supplied in abscissa and ordinate pairs</td>
</tr>
<tr>
<td>xy_in_file_only</td>
<td>bool</td>
<td>True</td>
<td>If the data file contains only abscissa and ordinate data</td>
</tr>
</tbody>
</table>
The Problem block is designed to hold the numerical systems that are ultimately solved. Tables E.4 and E.5 show the problem blocks of TreeKangaroo and TreeFrog, respectively. Since there is no equation system solved in TreeKangaroo, the “solve” option is set to false. In TreeFrog, there are two types of problems supported: “ProteusSP3Problem” and “ProteusP1Problem”. They are all derived from ExternalProblem.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Input Data</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Problem] type</td>
<td>FEProblem</td>
<td>-</td>
<td>MOOSE Problem type in TreeKangaroo</td>
</tr>
<tr>
<td>solve</td>
<td>false</td>
<td>-</td>
<td>Whether or not actually solve the nonlinear system. In TreeKangaroo, it is always set to false.</td>
</tr>
<tr>
<td>restart_file_base</td>
<td>FileNameNoExtension</td>
<td>-</td>
<td>File base name used for restart (e.g. / or /LATEST to grab the latest file available). Used when restart transient calculation from steady state solution</td>
</tr>
<tr>
<td>skip_additional_restart_data</td>
<td>bool</td>
<td>False</td>
<td>True to skip additional data in equation system for restart. It needs to be true for starting a transient calculation with a steady state solution.</td>
</tr>
<tr>
<td>Keyword</td>
<td>Input Data</td>
<td>Default value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>[Problem]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>ProteusSP3Problem</td>
<td>-</td>
<td>Use PROTEUS-NODAL SP3 solver or P1 solver</td>
</tr>
<tr>
<td>problem_type</td>
<td>Steady</td>
<td>Steady</td>
<td>Specify a steady state or transient problem</td>
</tr>
<tr>
<td>sync_to_mesh_power</td>
<td>string</td>
<td>-</td>
<td>The auxiliary variable name to hold the power density data in TreeFrog</td>
</tr>
<tr>
<td>sync_from_mesh_temperature</td>
<td>string</td>
<td>-</td>
<td>The auxiliary variable name to hold the temperature data from SAM in TreeFrog</td>
</tr>
<tr>
<td>sync_from_mesh_velocity</td>
<td>string</td>
<td>-</td>
<td>The auxiliary variable name to hold the velocity data from SAM in TreeFrog</td>
</tr>
<tr>
<td>sync_from_mesh_density</td>
<td>string</td>
<td>-</td>
<td>The auxiliary variable name to hold the density data from SAM in TreeFrog</td>
</tr>
<tr>
<td>total_node_number</td>
<td>unsigned int</td>
<td>-</td>
<td>The number of assemblies in PROTEUS-NODAL</td>
</tr>
<tr>
<td>total_plane_number</td>
<td>unsigned int</td>
<td>-</td>
<td>The number of axial planes in PROTEUS-NODAL</td>
</tr>
<tr>
<td>flow_plane_down</td>
<td>unsigned int</td>
<td>-</td>
<td>The first flow plane number in PROTEUS-NODAL</td>
</tr>
<tr>
<td>flow_plane_up</td>
<td>unsigned int</td>
<td>-</td>
<td>The last flow plane number in PROTEUS-NODAL</td>
</tr>
<tr>
<td>channel_total_number</td>
<td>unsigned int</td>
<td>-</td>
<td>Total number of core channels in the SAM input</td>
</tr>
<tr>
<td>channel_number</td>
<td>vector</td>
<td>-</td>
<td>The corresponding channel number in SAM of each PROTEUS-NODAL assembly. If 0 is set for an assembly, this assembly is not included in the SAM model (not flowing assembly)</td>
</tr>
<tr>
<td>geometry_type</td>
<td>RZ</td>
<td>-</td>
<td>Only use when problem type is ProteusP1Problem. It provides whether the geometry in PROTEUS-NODAL input is RZ or Hexagonal</td>
</tr>
<tr>
<td>r_grid</td>
<td>vector</td>
<td>-</td>
<td>Only use when geometry_type is RZ. The first value is the inner radius of first radial node. Other value is the outer radius of each radial node</td>
</tr>
</tbody>
</table>


E.1.5. Executioner

The Executioner block specifies how the simulation will be executed. It includes the commands to control the solver behavior and time stepping. In TreeKangaroo and TreeFrog, three kinds of executioner are utilized: “Steady”, “SteadyWithPicardCheck”, and “Transient”. Tables E.6, E.7, and E.8 show the input descriptions for these three executioner blocks. There are many options available in these executioners, but only the options used in TreeKangaroo and TreeFrog are listed. The complete list of available options of these executioners can be found in the MOOSE website. A steady state calculation is usually performed with the Steady Executioner in TreeFrog and with the SteadyWithPicardCheck Executioner in TreeKangaroo. In the Transient executioner, the TimeSteppers system is usually used as a sub-block in Executioner block to control how to move through time (i.e., to select the time step size). FuctionDT is usually used in TreeKangaroo and TreeFrog.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Input Data</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Executioner]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>Steady</td>
<td>-</td>
<td>Steady Executioner used in TreeFrog</td>
</tr>
<tr>
<td>[]</td>
<td>Table E.6. Input Description for Steady Executioner Block</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Input Data</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Executioner]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>SteadyWithPicardCheck</td>
<td>-</td>
<td>Use Picard iteration in Steady calculation</td>
</tr>
<tr>
<td>disable_picard _residual_norm _check</td>
<td>True</td>
<td>-</td>
<td>Disable the default Picard residual norm evaluation of MOOSE</td>
</tr>
<tr>
<td>picard_max_its</td>
<td>unsigned int</td>
<td>1</td>
<td>Specifies the maximum number of Picard iterations</td>
</tr>
<tr>
<td>pp_name</td>
<td>PostprocessorName</td>
<td>-</td>
<td>Postprocessor for custom Picard convergence check</td>
</tr>
<tr>
<td>pp_step_tol</td>
<td>double</td>
<td>-</td>
<td>The relative difference of the postprocessor values between two iterations to shoot for during Picard iterations</td>
</tr>
<tr>
<td>[]</td>
<td>Table E.7. Input Description for SteadyWithPicardCheck Executioner Block</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Input Data</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Executioner]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>Transient</td>
<td>-</td>
<td>Executor for time varying simulations</td>
</tr>
<tr>
<td>start_time</td>
<td>double</td>
<td>0</td>
<td>The start time of the simulation</td>
</tr>
<tr>
<td>[]</td>
<td>Table E.8. Input Description for Transient Executioner Block</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
end_time double 1E+30 The end time of the simulation
dt double 1 The time step size between solves

[./TimeStepper]

<table>
<thead>
<tr>
<th>type</th>
<th>FunctionDT</th>
<th>-</th>
<th>Timestepper whose steps vary over time according to a user-defined function</th>
</tr>
</thead>
<tbody>
<tr>
<td>function</td>
<td>FunctionName</td>
<td>-</td>
<td>The name of the time-dependent function that prescribes the time step size</td>
</tr>
</tbody>
</table>

E.1.6. MultiApps

The MultiApps (multi-application) block is used in a master application to specify the parameters for sub-applications. The parameters for each sub-application are included in a sub-block of MultiApps block. Table E.9 shows the input description for MultiApps block. Two types of MultiApps can be used in TreeKangaroo: “FullSolveMultiApp” for steady state calculations and “TransientMultiApp” for transient calculations. FullSolveMultiApp performs a complete simulation during each execution, so it is used in the steady state calculations. TransientMultiApp is designed to perform simulations with sub-applications that progress in time with the master application, which is used in the transient calculations.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Input Data</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[MultiApps]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[./sub-app]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>FullSolveMultiApp TransientMultiApp</td>
<td>-</td>
<td>FullSolveMultiApp for steady state calculations and TransientMultiApp for transient calculations</td>
</tr>
<tr>
<td>app_type</td>
<td>treefrogApp SamApp</td>
<td>-</td>
<td>The type of application to build (applications not registered can be loaded with dynamic libraries). Master application type will be used if not provided.</td>
</tr>
<tr>
<td>input_files</td>
<td>FileName</td>
<td>-</td>
<td>The input file for each application</td>
</tr>
<tr>
<td>library_path</td>
<td>string</td>
<td>-</td>
<td>Path to search for dynamic libraries</td>
</tr>
<tr>
<td>execute_on</td>
<td>NONE, INITIAL, LINEAR, NONLINEAR, TIMESTEP_END, TIMESTEP_BEGIN, FINAL, CUSTOM</td>
<td>TIMESTEP_BEGIN</td>
<td>Indicate when this subapp should be executed</td>
</tr>
<tr>
<td>sub_cycling</td>
<td>bool</td>
<td>False</td>
<td>Set to true to allow this MultiApp to take smaller time steps than the master</td>
</tr>
</tbody>
</table>


E.1.7. Transfers

Transfer objects in MOOSE are designed to move data to and from the sub-applications. Table E.10 presents the input description for transfer block. The data transfer between TreeKangaroo and TreeFrog is performed through MultiAppCopyTransfer. The data transfer between TreeKangaroo and SAM is carried out by MultiAppVPPTTransferFromSam and MultiAppPowerTransferToSam.

Table E.10. Input Description for Transfers Block

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Input Data</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Transfers]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>./MultiAppCopyTransfer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>MultiAppCopyTransfer</td>
<td>-</td>
<td>The meshes in the master and sub application must be identical</td>
</tr>
<tr>
<td>direction</td>
<td>to_multiapp from_multiapp</td>
<td>-</td>
<td>Whether this transfer will be 'to' or 'from' a MultiApp. Bidirectional when both from_multiapp and to_multiapp are provided</td>
</tr>
<tr>
<td>multi_app</td>
<td>MultiAppName</td>
<td>-</td>
<td>The name of the MultiApp to use</td>
</tr>
<tr>
<td>Source variable</td>
<td>vector</td>
<td>-</td>
<td>The variable to transfer from</td>
</tr>
<tr>
<td>variable</td>
<td>vector</td>
<td>-</td>
<td>The auxiliary variable to store the transferred values in</td>
</tr>
<tr>
<td>execute_on</td>
<td>NONE, INITIAL, LINEAR, NONLINEAR, TIMESTEP_END, TIMESTEP_BEGIN, FINAL, CUSTOM</td>
<td>Same as MultiApps</td>
<td>Indicate when this transfer function should be executed</td>
</tr>
</tbody>
</table>

[./]

./MultiAppPowerTransferToSam

<p>| type | MultiAppPowerTransferToSam | - | Transfer power density from TreeKangaroo to SAM |
| direction | to_multiapp from_multiapp | - | Whether this Transfer will be 'to' or 'from' a MultiApp, or bidirectional, by providing both FROM_MULTIAPP and TO_MULTIAPP |
| multi_app | SAM MultiAppName | - | The name of the SAM MultiApp specified in MultiApps block |</p>
<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proteus_power</td>
<td>The name of the Aux variable to transfer the power density value from <code>proteus_power</code> to <code>sam_power</code> variable name.</td>
</tr>
<tr>
<td>sam_power</td>
<td>The name of the Aux variable to transfer the power density value to <code>sam_power</code> variable name.</td>
</tr>
<tr>
<td>channel_index</td>
<td>The channel index in SAM. The channels are ordered from center to outer, i.e. the index for center channel is 1.</td>
</tr>
<tr>
<td>channel_total_number</td>
<td>Total core channel number in SAM input.</td>
</tr>
<tr>
<td>elem_begin</td>
<td>The id for first element in SAM mesh of this channel.</td>
</tr>
<tr>
<td>elem_end</td>
<td>The id for last element in SAM mesh of this channel.</td>
</tr>
<tr>
<td>execute_on</td>
<td>Same as MultiApps. Indicate when this transfer function should be executed.</td>
</tr>
<tr>
<td>type</td>
<td>Transfer temperature, velocity, density from SAM.</td>
</tr>
<tr>
<td>direction</td>
<td>Whether this Transfer will be 'to' or 'from' a MultiApp, or bidirectional, by providing both FROM_MULTIAPP and TO_MULTIAPP.</td>
</tr>
<tr>
<td>multi_app</td>
<td>The name of the SAM MultiApp specified in MultiApps block.</td>
</tr>
<tr>
<td>vector_postprocessor</td>
<td>The name of the VectorPostprocessor in SAM input to transfer the data from VectorPostprocessorName variable name.</td>
</tr>
<tr>
<td>variable_vpp</td>
<td>The name of variables in SAM that the VectorPostprocessor operates on.</td>
</tr>
<tr>
<td>variable</td>
<td>The name of aux variable to store the transferred data. The sequence of the variables.</td>
</tr>
</tbody>
</table>
aux variables in this input is the same as the sequence in variable sam

<table>
<thead>
<tr>
<th>keyword</th>
<th>input data</th>
<th>default value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel_index</td>
<td>unsigned int</td>
<td>-</td>
<td>The channel index in SAM. The channels are ordered from center to outer, i.e. the index for center channel is 1</td>
</tr>
<tr>
<td>channel_total_number</td>
<td>unsigned int</td>
<td>-</td>
<td>Total core channel number in SAM input</td>
</tr>
</tbody>
</table>

E.1.8. Outputs

The Output block is designed to set different output types. Table E.11 shows the input description of output block. The output types used in TreeKangaroo and TreeFrog are Console, Checkpoint, and CSV. The Console type is to output to screen. The Checkpoint is the MOOSE internal format used for restart and recovery. The CSV is to write the postprocessor and scalar variables to a CSV file. Additional output types supported in the MOOSE framework can be found in the MOOSE website. To activate a certain output type, MOOSE allows two ways. One is the “short-cut” syntax in the output block shown in Table E.11. In this case, the output type is activated by setting the output short-cut to True. The other is the “sub-block” syntax shown in Table E.12. Within each sub-block, there are options to control how to output data.

Table E.11. Input Description for Outputs Block with Short-cut Syntax

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Input Data</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Outputs]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>console</td>
<td>bool</td>
<td>-</td>
<td>Short-cut syntax for Console output</td>
</tr>
<tr>
<td>checkpoint</td>
<td>bool</td>
<td>-</td>
<td>Short-cut syntax for Checkpoint output</td>
</tr>
<tr>
<td>csv</td>
<td>bool</td>
<td>-</td>
<td>Short-cut syntax for CSV output</td>
</tr>
</tbody>
</table>

Table E.12. Input Description for Outputs Block with Sub-block Syntax

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Input Data</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Outputs]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[.//]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>Console</td>
<td>-</td>
<td>Write to the screen and optionally a file</td>
</tr>
<tr>
<td>file_base</td>
<td>string</td>
<td>-</td>
<td>Desired solution output name without extension. If not provided, MOOSE sets the output file name of master app to the input file name of master app and sets the output file name of subapp to the input file name of</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>output_file</td>
<td>bool</td>
<td>False, Output to the file</td>
<td></td>
</tr>
<tr>
<td>output_screen</td>
<td>bool</td>
<td>True, Output to the screen</td>
<td></td>
</tr>
<tr>
<td>file_base</td>
<td>string</td>
<td>Desired solution output name without extension. If not provided, MOOSE sets the output file name of master app to the input file name of master app and sets the output file name of subapp to the input file name of master app plus the subapp name.</td>
<td></td>
</tr>
<tr>
<td>execute_on</td>
<td>string</td>
<td>NONE, INITIAL, LINEAR, NONLINEAR, TIMESTEP_END, TIMESTEP_BEGIN, FINAL, CUSTOM</td>
<td></td>
</tr>
<tr>
<td>start_step</td>
<td>int</td>
<td>Time step at which this output object begins to operate</td>
<td></td>
</tr>
<tr>
<td>end_step</td>
<td>int</td>
<td>Time step at which this output object stops operating</td>
<td></td>
</tr>
<tr>
<td>time_column</td>
<td>bool</td>
<td>True, Whether or not the 'time' column should be written for Postprocessor CSV files</td>
<td></td>
</tr>
</tbody>
</table>

### CSV Output

`type` | CSV | `-` | Output for postprocessors, vector postprocessors, and scalar variables using CSV
`file_base` | string | `-` | Desired solution output name without extension. If not provided, MOOSE sets the output file name of master app to the input file name of master app and sets the output file name of subapp to the input file name of master app plus the subapp name.
`execute_on` | NONE, INITIAL, LINEAR, NONLINEAR, TIMESTEP_END, TIMESTEP_BEGIN, FINAL, CUSTOM | INITIAL, TIMESTEP_END | Indicate when this output should be executed

### Checkpoint Output

`type` | Checkpoint | `-` | Output for MOOSE recovery checkpoint files
`file_base` | string | `-` | Desired solution output name without extension. If not provided, MOOSE sets the output file name of master app to the input file name of master app and sets the output file name of subapp to the input file name of master app plus the subapp name.
`execute_on` | NONE, INITIAL, LINEAR, NONLINEAR, TIMESTEP_END, TIMESTEP_BEGIN, FINAL, CUSTOM | INITIAL, TIMESTEP_END | Indicate when this output should be executed

`start_step` | int | `-` | Time step at which this output object begins to operate
`end_step` | int | `-` | Time step at which this output object stops operating
E.2. Sample Inputs

Fig. E.1 presents a sample input for TreeKangaroo, and Fig. E.2 presents a sample input for TreeFrog.

```plaintext
[Mesh]
type = GeneratedMesh
dim = 2
ny = 3
nx = 20
[]

[AuxVariables]
[./proteus_power_density]
  order = FIRST
  family = LAGRANGE
  initial_condition = 925
[../]

[./sam_temperature]
  order = FIRST
  family = LAGRANGE
  initial_condition = 925
[../]

[./sam_velocity]
  order = FIRST
  family = LAGRANGE
  initial_condition = 1.15
[../]

[./sam_density]
  order = FIRST
  family = LAGRANGE
  initial_condition = 4166
[../]

[./sam_from_proteus_power]
[../]

[./sam_temperature_sub]
[../]

[./sam_velocity_sub]
```
[../]
[./sam_density_sub][../]

[./proteus_power_density_sub][../]
[]

[Problem]
  solve = false
  type = FEProblem
[]

[Postprocessors]
  [.pnorm]
    type = NodalL2Norm
    variable = proteus_power_density
    execute_on = 'initial timestep_end'
[../]
[]

[Executioner]
  type = SteadyWithPicardCheck
  disable_picard_residual_norm_check = true
  picard_max_its = 10
  pp_name = pnorm
  pp_step_tol = 1e-4
[]

[MultiApps]
  [.treefrog]
    type = FullSolveMultiApp
    app_type = treefrogApp
    input_files = 'treefrog.i'
    library_path = ' /treefrog/lib'
[../]

  [.sam]
    type = FullSolveMultiApp
    app_type = SamApp
    input_files = sam.i'
    library_path = ' /SAM/lib'
    execute_on = TIMESTEP_END
[../]
[]

[Transfers]
  [.from_proteus]
    type = MultiAppCopyTransfer
    direction = from_multiapp
    multi_app = treefrog
    source_variable = "proteus_power_density_sub"
    variable = "proteus_power_density"
[../]
Multiphysics Coupling of PROTEUS-NODAL and SAM for Molten Salt Reactor Simulation

February 28, 2020


type = MultiAppCopyTransfer
  direction = to_multiapp
  multi_app = treefrog
  source_variable = 'sam_temperature'
  variable = 'sam_temperature_sub'


type = MultiAppCopyTransfer
  direction = to_multiapp
  multi_app = treefrog
  source_variable = 'sam_velocity'
  variable = 'sam_velocity_sub'


type = MultiAppCopyTransfer
  direction = to_multiapp
  multi_app = treefrog
  source_variable = 'sam_density'
  variable = 'sam_density_sub'


type = MultiAppVPPTransferFromSam
  direction = from_multiapp
  vector_postprocessor = fluid_all_1
  variable_vpp = 'temperature velocity rho'
  channel_index = 1
  channel_total_number = 4
  multi_app = sam
  variable = 'sam_temperature sam_velocity sam_density'


type = MultiAppVPPTransferFromSam
  direction = from_multiapp
  vector_postprocessor = fluid_all_2
  variable_vpp = 'temperature velocity rho'
  channel_index = 2
  channel_total_number = 4
  multi_app = sam
  variable = 'sam_temperature sam_velocity sam_density'


type = MultiAppVPPTransferFromSam
  direction = from_multiapp
  vector_postprocessor = fluid_all_3
  variable_vpp = 'temperature velocity rho'
  channel_index = 3
  channel_total_number = 4
  multi_app = sam
  variable = 'sam_temperature sam_velocity sam_density'
```plaintext
[../

[./from_sam_4]
  type = MultiAppVPPTtransferFromSam
  direction = from_multiapp
  vector_postprocessor = fluid_all_4
  variable_vpp = 'temperature velocity rho'
  channel_index = 4
  channel_total_number = 4
  multi_app = sam
  variable = 'sam_temperature sam_velocity sam_density'
[../

[./to_sam_1]
  type = MultiAppPowerTransferToSam
  direction = to_multiapp
  proteus_power = proteus_power_density
  sam_power = sam_from_proteus_power
  channel_index = 1
  channel_total_number = 4
  elem_begin = 0
  elem_end = 19
  multi_app = sam
  [../

[./to_sam_2]
  type = MultiAppPowerTransferToSam
  direction = to_multiapp
  proteus_power = proteus_power_density
  sam_power = sam_from_proteus_power
  channel_index = 2
  channel_total_number = 4
  elem_begin = 20
  elem_end = 39
  multi_app = sam
[../

[./to_sam_3]
  type = MultiAppPowerTransferToSam
  direction = to_multiapp
  proteus_power = proteus_power_density
  sam_power = sam_from_proteus_power
  channel_index = 3
  channel_total_number = 4
  elem_begin = 40
  elem_end = 59
  multi_app = sam
[../

[./to_sam_4]
  type = MultiAppPowerTransferToSam
  direction = to_multiapp
  proteus_power = proteus_power_density
  sam_power = sam_from_proteus_power
  channel_index = 4
  channel_total_number = 4
  elem_begin = 60
```
Fig. E.1. Sample input for TreeKangaroo.

```plaintext
elem_end = 79
multi_app = sam

[Outputs]
csv = true
checkpoint = true

[Mesh]
type = GeneratedMesh
dim = 2
ny = 3
nx = 20

[AuxVariables]
[./proteus_power_density_sub]
  order = FIRST
  family = LAGRANGE

[./sam_temperature_sub]
  order = FIRST
  family = LAGRANGE
  initial_condition = 925

[./sam_velocity_sub]
  order = FIRST
  family = LAGRANGE
  initial_condition = 1.15

[./sam_density_sub]
  order = FIRST
  family = LAGRANGE
  initial_condition = 4166

[./proteus_power_density]

[Problem]
type = ProteusSP3Problem
sync_to_mesh_power = proteus_power_density_sub
sync_from_mesh_temperature = sam_temperature_sub
sync_from_mesh_velocity = sam_velocity_sub
sync_from_mesh_density = sam_density_sub
total_node_number = 73
total_plane_number = 24
flow_plane_down = 3
```
Fig. E.2. Sample input for TreeFrog.