

Nuclear Reactor Multiphysics via Bond Graph Formalism

by

Eugeny Sosnovsky

B.S., Mechanical Engineering and Physics
Worcester Polytechnic Institute, 2008

S.M., Nuclear Science and Engineering
Massachusetts Institute of Technology, 2010

Submitted to the Department of Nuclear Science and Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Science in Nuclear Science and Engineering
at the

Massachusetts Institute of Technology
June 2014

© 2014 Massachusetts Institute of Technology
All rights reserved.

Author:

Department of Nuclear Science and Engineering
May 16, 2014

Certified by:

Benoit Forget, Ph.D.
Associate Professor of Nuclear Science and Engineering
Thesis Supervisor

Certified by:

Edward E. Pilat, Ph.D.
Research Scientist
Thesis Reader

Certified by:

Kord S. Smith, Ph.D.
KEPCO Professor of the Practice of Nuclear Science and Engineering
Thesis Reader

Accepted by:

Mujid S. Kazimi, Ph.D.
TEPCO Professor of Nuclear Engineering
Chair, Department Committee on Graduate Students

Nuclear Reactor Multiphysics via Bond Graph Formalism

by

Eugeny Sosnovsky

Submitted to the Department of Nuclear Science and Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Science in Nuclear Science and Engineering

June 2014

Abstract

This work proposes a simple and effective approach to modeling nuclear reactor multiphysics problems using bond graphs. Conventional multiphysics simulation paradigms normally use operator splitting, which treats the individual physics separately and exchanges the information at every time step. This approach has limited accuracy, and so recently, there has been an increased interest in fully coupled physics simulation. The bond graph formalism has recently been suggested as a potential paradigm for reactor multiphysics simulation; this work develops the tools necessary to utilize bond graphs for practical transient reactor analysis.

The bond graph formalism was first introduced to solve the multiphysics problem in electromechanical systems. Over the years, it has been used in many fields including nuclear engineering, but with limited scope due to its perceived impracticality in large systems. Bond graph formalism works by first representing a discretized multiphysics system using a group of graph elements, connected with bonds; the bonds transport conserved quantities, and the elements impose the relations between them. The representation can be automatically converted into a state derivative vector, which can be integrated in time.

In an earlier work, the bond graph formalism was first applied to neutron diffusion, and coupled to diffusive heat transfer in a 1D slab reactor. In this work, methods are developed to represent, using bond graphs, 2D and 3D multigroup neutron diffusion with precursors, nonlinear point kinetics, and basic nearly-incompressible 1D flow for fully coupled reactor simulation. High-performance, matrix-based bond graph processing methods were developed to support the simulation of medium- and large-scale problems.

A pressurized water reactor point kinetics, single-channel rod ejection benchmark problem was used to verify the nonlinear point kinetics representation. 2D and 3D boiling water reactor control blade drop problems were also successfully simulated with the matrix-based bond graph processing code. The code demonstrated 3rd-order convergence in time, a very desirable property of fully coupled time integrators.

Thesis Supervisor: Benoit Forget, Ph.D.

Title: Associate Professor of Nuclear Science and Engineering

Thesis Reader: Edward E. Pilat, Ph.D.

Title: Research Scientist

Thesis Reader: Kord S. Smith, Ph.D.

Title: KEPCO Professor of the Practice of Nuclear Science and Engineering

Acknowledgements

My greatest thanks and appreciation, first and foremost, go to my advisor, Professor Benoit Forget. His support, help, and patience over the course of this project cannot be overstated. I would not have been able to undertake, much less successfully complete, this type of autonomous, highly uncertain but exciting project, under any other advisor, and for this I am endlessly grateful.

I also wish to thank Professor Kord Smith for recommending the LRA BWR control blade drop problem to me as well as his specific critiques and analyses of my results. My other reader, Dr. Edward Pilat, has also been a phenomenal resource; it was very exciting to be able to speak to someone who learned from the inventor of the formalism that I, 50 years later, extended. Reactor spatial kinetics is a very old and important problem, and I am very thankful to both of my readers for their genuinely excited, delicate corrections and guidance over the course of this work.

Three out of four years of this work were funded by a Nuclear Energy University Program (NEUP) fellowship, which is gratefully acknowledged.

A “Thank you!” goes to my parents, Ray and Olga Hayes, whose physical, financial and mental support helped me immeasurably during my graduate studies, and earlier. More importantly, thank you for taking care of Poirot, my pug. To Poirot specifically — well done, pug!

Lastly, I would like to deeply thank my wife, Yana, for the love and support she has given me over the many years we have been together, and for the future as well.

Contents

Abstract	3
Contents	7
List of Figures	9
List of Tables	11
1 Introduction	15
1.1 Background on Nuclear Reactor Multiphysics	15
1.2 Background on Bond Graph Formalism	17
1.3 Objectives	17
1.4 A Note to the Reader	18
2 Background	19
2.1 Nuclear Reactor Physics	23
2.1.1 Neutron Transport	23
2.1.2 Neutron Diffusion	58
2.1.3 Neutron Point Kinetics	67
2.1.4 Thermal Hydraulics	71
2.1.5 Thermal Hydraulics-Neutron Transport Coupling	91
2.2 Existing Multiphysics Simulation Methods	98
2.2.1 Operator Splitting and Full Coupling	98
2.2.2 Physics-specific Methods	100
2.2.3 General Methods	101
2.3 Bond Graph Formalism	103
2.3.1 Bond Graph Representation Fundamentals	103
2.3.2 Bond Graph Process	110
2.3.3 Bond Graph Processing Codes	112
2.4 Existing Bond Graph-Based Reactor Simulation Methods	113
2.4.1 Linearized Point Kinetics Models	113
2.4.2 Coupled Neutron and Thermal Diffusion Models	114
3 Bond Graph Representation of Nuclear Reactor Physics	115
3.1 Nonlinear Point Kinetics Models	116
3.2 Systems-level Thermal Hydraulics	127
3.3 Multidimensional, Multigroup Neutron Diffusion	135

3.4	Spherical Harmonics and Discrete Ordinates Methods	142
4	Automated Bond Graph Processing	149
4.1	Implementation Methods	149
4.1.1	Separate Bond Graph Processing Code	150
4.1.2	Bond Graph Processing via Object-Oriented Libraries	150
4.2	Processing Algorithms	151
4.2.1	Symbolic Sorting	152
4.2.2	Matrix-Based Sorting	153
4.3	Use of Bond Graphs for Steady State Problems	159
4.3.1	Problems with Forcing Functions	159
4.3.2	Eigenvalue Problems	159
4.4	Time Integration	160
5	Neutron Point Kinetics with Feedback Problem	163
5.1	Benchmark Problem Definition	163
5.2	Steady State Search	168
5.3	Results	169
5.3.1	Transient Results	169
5.3.2	Convergence Results	172
5.4	Summary	174
6	Two-dimensional BWR Control Blade Drop Problem	177
6.1	Benchmark Problem Definition	178
6.2	Coarse Mesh Model	182
6.2.1	Steady State Search	182
6.2.2	Transient Results	184
6.2.3	Convergence Results	186
6.3	Fine Mesh Model	188
6.3.1	Steady State Search	188
6.3.2	Transient Results	191
6.3.3	Convergence Results	191
6.4	Code Performance	192
6.5	Summary	193
7	Three-dimensional BWR Control Blade Drop Problem	195
7.1	Benchmark Problem Definition	195
7.2	Steady State Search	197
7.3	Results	198
7.4	Summary	200
8	Conclusions and Recommendations for Future Work	203
8.1	Viability of Bond Graphs for Reactor Multiphysics Analysis	203
8.2	Viability of Large-scale Automated Bond Graph Processing	204
8.3	Recommendations for Future Work	204
	References	205

List of Figures

2.1	Energy Domain Discretization	34
2.2	Fuel Pin Geometries	41
2.3	2D BASALA Geometry	42
2.4	2D BWR Full Core Geometry Specification	43
2.5	Time Integration Approaches	99
2.6	Conventional and Bond Graph Code Development Processes	111
3.1	Power PKE Bond Graph Representation	119
3.2	Precursor PKEs Bond Graph Representation	121
3.3	Reactivity with Linear Separable Feedback Bond Graph Representation	122
3.4	Reactivity with Nonlinear Separable Feedback Bond Graph Representation	123
3.5	Reactivity with Coupled Feedback Bond Graph Representation	124
3.6	1D Advection Bond Graph Representation	132
3.7	Fuel Cylindrical HDE Bond Graph Representation	133
3.8	Fuel, Gap and Clad Cylindrical HDE Bond Graph Representation	133
3.9	Multigroup NDE Node Bond Graph Representation	137
3.10	Neutron Diffusion Internodal Resistance Bond Graph Representation	140
3.11	Neutron Diffusion Dirichlet BC Bond Graph Representation	141
3.12	Spherical Harmonics General Local Reactions Bond Graph Representation	143
3.13	Spherical Harmonics Fission Neutron Reactions Bond Graph Representation	144
3.14	Spherical Harmonics Single Streaming Term Bond Graph Representation	145
3.15	Discrete Ordinates Streaming Term Bond Graph Representation	146
5.1	PWR PKE Benchmark Problem Material Thermal Properties	166
5.2	PWR PKE Initial Fuel Temperature Spatial Convergence Plot	168
5.3	PWR PKE Benchmark Problem Reactivity Transients	170
5.4	PWR PKE Benchmark Problem Reactor Power Transients	170
5.5	PWR PKE Benchmark Problem Hottest Axial Segment Temperature Transients	171
5.6	PWR PKE MATLAB ode15s Time Step	171
5.7	PWR PKE Benchmark Problem Temporal Convergence Plots	173
6.1	2D LRA BWR Benchmark Problem Specification	180
6.2	2D LRA BWR Coarse Mesh Initial Power Density	184
6.3	2D LRA BWR Coarse Mesh Average Core Power Density Transients	185
6.4	2D LRA BWR Coarse Mesh Average Core Temperature Transients	185
6.5	2D LRA BWR Coarse Mesh Temporal Convergence Plots	186

LIST OF FIGURES

6.6	2D LRA BWR Steady State Spatial Convergence Plots	190
6.7	2D LRA BWR Fine Mesh Initial Power Density	190
6.8	2D LRA BWR Fine Mesh Transients	191
6.9	2D LRA BWR Fine Mesh Peak Average Core Power Temporal Convergence Plot . . .	192
7.1	3D LRA BWR Benchmark Problem Axial Specifications	196
7.2	3D LRA BWR Initial Power Density	198
7.3	3D LRA BWR Transients	199
7.4	3D LRA BWR Temporal Convergence Plots	199

List of Tables

2.1	Full Bond Causality-Direction Configurations	105
2.2	Bond and State Variables in Different Physical Domains	106
2.3	Bond Graph Elements	108
5.1	PWR PKE Benchmark Problem Axial Parameters	164
5.2	PWR PKE Benchmark Problem Precursor Family Properties	164
5.3	PWR PKE Benchmark Problem Additional Parameters	165
6.1	LRA BWR Benchmark Problem Initial Region Properties	180
6.2	LRA BWR Benchmark Problem Additional Reactor-wide Parameters	181
6.3	BGSolver v1.03 and v2.0 Performance on the 2D LRA BWR Benchmark	193

To my grandfather

Chapter 1

Introduction

The important physics for reactor simulation include neutron transport and thermal hydraulics which are inherently strongly coupled. Because of this, nuclear reactor modeling is a multiphysics problem, which is normally treated through operator splitting: integrating each physics separately, and exchanging the state information at every time step.

The operator splitting approach has limitations, and so there has recently been an increased interest in developing fully coupled codes aimed at modeling coupled physics as a single problem, as opposed to multiple connected problems. The primary incentives for doing so are: (a) it is easier to obtain higher order convergence in time for fully coupled transient codes, and (b) full coupling reduces the number of iterations required for convergence of steady state codes. One existing approach to full coupling is the “bond graph formalism” [1].

The bond graph formalism is a technique for modeling engineering systems as combinations of connected elements. Bond graphs were originally introduced for mechatronics, but over time grew from a comprehensive methodology to model mechatronic systems into a complete research field, concerned with modeling mechanical, electrical, magnetic, hydraulic, thermal, and even optical and financial systems. Bond graphs have been used for modeling various field problems, such as thermal diffusion, but have never, until recently, been applied to neutron transport. Bond graph formalism works by first representing a discretized multiphysics system using a group of graph elements, connected with bonds; the bonds transport conserved quantities, and the elements impose the relations between them. The representation can be automatically converted into a state derivative vector, which can be integrated in time.

In my S.M. thesis, I investigated the possibility of the use of bond graphs for reactor analysis. Here, this work is continued.

1.1 Background on Nuclear Reactor Multiphysics

Chapter 2 presents the necessary background on nuclear reactor multiphysics in great detail; here, it is briefly summarized to understand the work’s objectives. References [2, 3] alone provide sufficient nuclear reactor analysis background for a general understanding of this work.

The important physics in transient safety analysis are those with time scales from milliseconds to several hours, but not longer. Assuming a stationary geometry (other than control assembly movement), the physics that have such time scales are time-dependent neutron transport, delayed neutron production from fission products, thermal hydraulics (the combination of fluid mechanics,

heat generation and transfer) and, possibly, Zircaloy oxidation. Zircaloy oxidation is frequently treated as an external source of thermal energy, or is incorporated into the thermal hydraulic model, which limits the analysis to two important classes of physics: spatial neutron kinetics (including delayed neutrons from fission products) and thermal hydraulics. This text focuses only on these two sets of physics in reactor transients, but it must be understood that in some cases, any of the above physics may be important in a short-term accident scenario.

Fundamentally, both thermal hydraulics and neutron transport are described by combinations of spatially continuous partial differential and integro-differential scalar and vector equations. Such systems are not analytically solvable, except in primitive special cases, and so must be solved numerically using appropriate discretizations. It should be noted that the thermal hydraulic equations are also “transport equations”: they describe the transport and conservation of fluid mass, momentum and energy, similarly to how the neutron transport equation describes the transport and conservation of neutrons.

Neutron transport models of core subregions or full cores vary in scale, geometry type, dimensionality and purpose. True neutron transport models solve the neutron transport equation directly, without simplifying physical assumptions; such solutions are very costly, and so are rarely affordable for full core analysis. Neutron diffusion models solve a significantly simplified, Fick’s law-based version of the neutron transport equation; many such models require geometric homogenization, and so are rarely applied for any geometry other than a full core. Neutron point kinetics models are also sometimes used; in these models, only the total number of neutrons in the core is modeled [4].

Similarly, thermal hydraulic models of reactor subregions or of the entire vessel also vary in scale, geometry type, dimensionality and purpose. A true geometry of a subregion or a full core or vessel can be modeled using computational fluid dynamics methods (CFD); similarly to true neutron transport, such models are very costly, and so are normally built for subregions of the reactor. Other approaches to reactor thermal hydraulic models rely on 1-dimensional approximations of either individual core subchannels, or entire lumped subvolumes of the vessel. The subchannels are discretized axially (along the direction of the bulk fluid flow), and may be radially connected, or independent. The lumped volume approach (also known as “systems-level”) is often used when the reactor vessel is modeled as part of the plant balance problem [3].

Most reactor transient analysis utilizes the full core neutron diffusion approach coupled with either systems-level or subchannel-level approaches. This work concentrates on neutron diffusion and systems-level approach; subchannel-level models are briefly addressed.

The thermohydraulic state of the reactor affects its neutron transport properties, and the fission and radioactive decay rate density profiles affect the heat generation rates. This makes reactor analysis, even if only the two physics discussed here are modeled, a multiphysics problem. To solve a multiphysics problem, besides correctly modeling the individual physics in it, the solution method must also adequately account for the coupling between the physics. There exist multiple approaches to multiphysics coupling; they broadly fall into the categories of “*operator splitting*” (OS) and “*full coupling*” (FC):

- **Operator splitting** constitutes the solution of the individual physics separately and exchanging the solutions at every time step (for a transient problem), or at every iteration (for an iterative steady state search). It can be viewed as solving several small problems.
- **Full coupling** constitutes the treatment of different physics as a single complex physics. It can be viewed as solving a single large problem.

One existing approach to full coupling is the “*bond graph formalism*”.

1.2 Background on Bond Graph Formalism

The bond graph formalism is a technique for modeling engineering systems as combinations of connected elements. The basic idea of modeling a system with bond graphs is to represent it using a system of graph elements connected with directed edges, called “bonds.” The bonds convey conserved quantities, and the elements store them and adjust their transfer rates. A bond graph processing algorithm, which can be automated, is applied to a bond graph system, resulting the formulation of the state derivative vector, which can then be integrated to obtain full information about the system’s dynamics.

Bond graph formalism is discussed in detail in chapter 2; the interested reader is recommended to consult it for a more complete understanding of the formalism, and its system representation techniques.

In my Master’s thesis, I investigated the possibility of representing nuclear reactors with bond graphs, and identified the limitations that at the time were preventing accurate reactor transient analysis with bond graphs. In this work, these limitations are addressed.

1.3 Objectives

Prior to my S.M. thesis research, the state of the use of automated bond graph processing for reactor analysis was as follows:

Neutron physics: Representation methods were developed for a linearized neutron point kinetics model; i.e., the neutron population cannot vary appreciably from the nominal. No neutron transport or neutron diffusion representation methods existed.

Thermal hydraulic physics: Basic lumped volume systems code-like bond graph representation techniques existed, developed primarily for internal combustion engine and chemical reactor modeling.

Automated bond graph processing: Basic lumped volume systems code-like bond graph representation techniques existed, developed primarily for internal combustion engine and chemical reactor modeling.

In my Master’s thesis, methods were developed for representing one-group 1D slab neutron diffusion using bond graphs, and a MATLAB-based bond graph processing code, reliant on MATLAB symbolic engine, was developed. These tools were successfully tested. This research acted as a proof of concept of the use of bond graphs for reactor analysis, but the resulting code and methods were limited in the scale of the problems they could address. For this reason, the following objectives were posed for this Sc.D. thesis:

Objective 1. To develop the bond graph formalism sufficiently to be able to model realistic reactor multiphysics transient problems. To do this, the following steps remained:

Sub-objective 1a. To expand the MATLAB Symbolic Engine-based bond graph processing code to support the multiport resistive element, necessary for multigroup and precursor representation.

Sub-objective 1b. To develop bond graph representation techniques for multidimensional, multigroup neutron diffusion with precursor families.

Sub-objective 1c. To identify, implement and test a fully numeric (not reliant on symbolic solutions) bond graph processing algorithm, to replace the corresponding steps in the bond graph processing code.

Objective 2. To analyze a full core multiphysics model using a prototype bond graph processing code.

Objective 3. To confirm or deny whether full coupling is computationally efficient for realistic reactor problems.

These objectives were successfully achieved.

1.4 A Note to the Reader

The intended audience for this text is: (a) nuclear engineers, and (b) bond graph specialists. Because this is a diverse, nearly mutually exclusive group of people, a very detailed background chapter 2 is provided. Chapter 2 provides a thorough background on both nuclear reactor multiphysics, and on the bond graph formalism.

Because nuclear engineers are expected to be familiar with the physics modeled through bond graphs in this work, they are recommended to skip section 2.1 of the text. It was primarily intended as a general reference on the modeled physics. Section 2.3 details the background on the bond graph formalism, and may be skipped by readers familiar with bond graphs without loss of continuity.

The subsequent chapters present the new contributions of this work. Chapters 3 and 4 detail the techniques developed for representing various nuclear reactor physics using bond graphs, and algorithms and codes developed as part of this work to process such representations, respectively. Three benchmark problems are addressed: a small point kinetics pressurized water reactor core benchmark (chapter 5), and a 2D and 3D boiling water reactor spatial kinetics control blade drop problem (chapters 6 and 7, respectively). The conclusions of the study are given in chapter 8.

Chapter 2

Background

Accurate models of nuclear reactor transients must represent multiple coupled physics. Depending on the type of transient, the fidelity of the model and the time scale of interest, the physics involved may include:

1. **Neutron transport.** The fundamental purpose of a nuclear fission reactor is to maintain, by means of a sustained nuclear chain reaction, a controllable density of neutrons in a specific region of the reactor. This neutron density can then be used for: (a) experimental and medical purposes, (b) testing reactor theory, (c) isotope transmutation, (d) generating power for (normally, naval) propulsion, and (e) generating electrical power (Ref. [5, Chapter 1]).

“Neutron transport” here refers to the scientific theory that describes the dynamics of neutron motion, neutron-initiated nuclear reaction rates, including neutron production and removal, and the numerical and computational methods for applying it. The application of neutron transport to nuclear reactor analysis is also known as “reactor physics.” Neutron transport is generally the fastest set of physics in a reactor, with some transport phenomena occurring over microseconds or faster.

References [6], [2, 5] and [4] are recommended for neutron transport fundamentals, discretization theory and modern neutron transport code practices, respectively.

2. **Isotope transmutation and fuel depletion.** In a nuclear reactor, nuclides change from one to another via three processes: (a) neutron capture, (b) fission, and (c) radioactive decay, which may be induced by a particle capture, or may occur independently. Most isotope transmutations, including fuel depletion, occur over long periods of operation (months), and so are outside the time scales of most transient analysis. However, delayed neutrons produced by decay of fission products, and the heat generated by minor actinides’ and fission product radioactive decays, can affect faster transients, which have time scales from several seconds to several hours.

Fuel depletion and long-term isotope transmutations are usually treated quasi-statically, and are analyzed using dedicated steady state neutron transport codes. References [4, 5] are recommended as summaries of the conventional underlying methods. Similarly, delayed neutron production is accounted for by neutron kinetics codes, discussed in detail by Stacey [2]. Decay heat generation is frequently decoupled from neutron analysis, and is instead treated as an explicit heat source for thermal hydraulic calculations. References [7, 8] discuss the modern

decay heat calculation methods for light water reactors; Refs. [9–11] provide a more general background on decay heat generation for all reactor types.

- 3. One- and two-phase fluid mechanics.** Fission reactions produce a significant amount of heat, which requires the nuclear reactors to have adequate cooling. The most common cooling systems work by using flowing or stationary fluids to remove heat from the fuel and transfer it to a heat sink. In some reactors, partial or full boiling also occurs at normal operation or in accident scenarios, which makes the flow in the system two-phase. Neutron transport is usually the fastest physics in a reactor, but depending on the transient, fluid mechanics (including single-phase) can be comparably fast, with representative time scales on the order of milliseconds. Additionally, when the dynamics of the full plant are analyzed, the fluid mechanics and heat transfer inside the reactor are the physics most commonly treated.

References [12, 13] provide a formal mathematical treatment of fluid mechanics; Refs. [3, 14] give a more practical, industry-specific look at how one- and two-phase fluid mechanics are treated in reactor and plant analysis.

- 4. Heat generation and transfer in fluids and in solids.** Neutron-induced fission, as well as radioactive decay, are responsible for the heat generation in a reactor that must be compensated for by the cooling system. This generated thermal energy is conductively transferred through the solid structures and fuel, and is convectively removed by the coolant, which may fully or partially boil in the process.

Except for adiabatic heat generation (a conservative assumption sometimes made in reactor analysis), heat generation and transfer are treated together with fluid mechanics, and have comparable time scales. References [3, 14] treat the two physics together, both in single-phase and two-phase systems.

- 5. Solid mechanics of the fuel and the structures.** While an operating nuclear reactor typically contains few moving parts (other than the control assemblies, the in-vessel pumps and the steam separators), the vessel, internal structures and the fuel all undergo significant mechanical stresses both during transients and at nominal steady state. These stresses are a result of thermal expansion, flow-induced vibration, long-term material restructuring due to irradiation, possible external sources (e.g., an earthquake) and the pressure applied by the gases produced as fission products (“fission gases”). Additionally, depending on the coolant and cladding material, the flowing liquid may also be abrasive.

In the event of mechanical failure, significant deformation can occur very rapidly (milliseconds, or faster). However, by definition, such geometry change constitutes a mechanical failure, and the majority of reactor analysis therefore assumes that all structural materials hold, and no geometry variation occurs. This allows the analysts to decouple the solid mechanics of the reactor with the other physics of interest, and to analyze them separately.

Reference [15] is recommended as an overview of the structural materials used in various reactor types, and discusses the underlying criteria in choosing the appropriate materials. The basic theory behind fuel element cladding, boiling water reactor (BWR) assembly cans, and the reactor pressure vessel mechanics, all of which can be viewed as pressure vessels, is summarized in Ref. [16]. Reference [17] provides a very thorough background on the finite element method normally used for numerical analysis of structural mechanics. Reference [18] discusses the fluid-structure interaction, which is an additional significant source of vibrational stresses.

6. **Materials degradation under irradiation.** The microstructure of materials exposed to radiation can be affected by the isotope transmutation induced by neutron captures and by neutron scattering, which displaces the atoms from their lattice. This gradual change in microstructure can cause material creep, solid phase change, or simply change the material's structural and thermal properties. This is a slow phenomenon, and so, while it is very important, it is not usually part of the transient analysis; instead, material properties are evaluated based on the known irradiation history.

References [19] and [20] discuss the effects of neutron exposure on steel and Zircaloy properties, respectively; these are the most common structural materials in nuclear reactors.

7. **Materials chemistry.** Lastly, while an ideal nuclear reactor environment would be chemically inert, the (often, radiation-assisted) chemical interaction between the coolant and the reactor materials must be carefully considered. At steady state, this chemical interaction leads to relatively gradual (months to years) corrosion and to “CRUD” (Chalk River Unidentified Deposit, or Corrosion-Related Unidentified Deposit) formation on the clad surface. In an accident scenario, additional chemical reactions can occur: the water coolant may rapidly oxidize the Zircaloy (zirconium-based alloy used for fuel cladding in many reactors) clad; this is an exothermal reaction, which produces hydrogen gas. Hydrogen gas is highly combustible in air, and so its production in an accident is likely to damage the reactor.

Because of the time scales involved, materials chemistry other than this rapid oxidation, similarly to materials degradation under irradiation, is not considered in transient analysis.

References [21] and [22] introduce nuclear corrosion and discuss CRUD deposition, respectively.

The list of recommended references above is detailed; however, Refs. [2, 3] alone provide sufficient nuclear reactor analysis background for a general understanding of this work.

The important physics in transient safety analysis are those with time scales from milliseconds to several hours, but not longer. As discussed above, assuming a stationary geometry (other than control assembly movement), the physics that have such time scales are time-dependent neutron transport, delayed neutron production from fission products, thermal hydraulics (the combination of fluid mechanics, heat generation and transfer) and, possibly, Zircaloy oxidation. Zircaloy oxidation is frequently treated as an external source of thermal energy, or is incorporated into the thermal hydraulic model, which limits the analysis to two important classes of physics: spatial neutron kinetics (including delayed neutrons from fission products) and thermal hydraulics. This text focuses only on these two sets of physics in reactor transients, but it must be understood that in some cases, any of the above physics may be important in a short-term accident scenario.

Fundamentally, both thermal hydraulics and neutron transport are described by combinations of spatially continuous partial differential and integro-differential scalar and vector equations. Such systems are not analytically solvable, except in primitive special cases, and so must be solved numerically using appropriate discretizations. It should be noted that the thermal hydraulic equations are also “transport equations”: they describe the transport and conservation of fluid mass, momentum and energy, similarly to how the neutron transport equation describes the transport and conservation of neutrons.

Neutron transport models of core subregions or full cores vary in scale, geometry type, dimensionality and purpose. True neutron transport models solve the neutron transport equation directly, without simplifying physical assumptions; such solutions are very costly, and so are rarely

affordable for full core analysis. Neutron diffusion models solve a significantly simplified, Fick's law-based version of the neutron transport equation; many such models require geometric homogenization, and so are rarely applied for any geometry other than a full core. Neutron point kinetics models are also sometimes used; in these models, only the total number of neutrons in the core is modeled [4].

Similarly, thermal hydraulic models of reactor subregions or of the entire vessel also vary in scale, geometry type, dimensionality and purpose. A true geometry of a subregion or a full core or vessel can be modeled using computational fluid dynamics methods (CFD); similarly to true neutron transport, such models are very costly, and so are normally built for subregions of the reactor. Other approaches to reactor thermal hydraulic models rely on 1-dimensional approximations of either individual core subchannels, or entire lumped subvolumes of the vessel. The subchannels are discretized axially (along the direction of the bulk flow), and may be radially connected, or independent. The lumped volume approach (also known as "systems-level") is often used when the reactor vessel is modeled as part of the plant balance problem [3].

Most reactor transient analysis utilizes the full core neutron diffusion approach coupled with either systems-level or subchannel-level approaches. This work concentrates on neutron diffusion and systems-level approach; subchannel-level models are briefly addressed.

The thermohydraulic state of the reactor affects its neutron transport properties, and the fission and radioactive decay rate density profiles affect the heat generation rates. This makes reactor analysis, even if only the two physics discussed here are modeled, a multiphysics problem. To solve a multiphysics problem, besides correctly modeling the individual physics in it, the solution method must also adequately account for the coupling between the physics. There exist multiple approaches to multiphysics coupling; they broadly fall into the categories of "*operator splitting*" (OS) and "*full coupling*" (FC):

- **Operator splitting** constitutes the solution of the individual physics separately and exchanging the solutions at every time step (for a transient problem), or at every iteration (for an iterative steady state search). It can be viewed as solving several small problems.
- **Full coupling** constitutes the treatment of different physics as a single complex physics. It can be viewed as solving a single large problem.

The operator splitting approach has limitations, and so there are reasons to develop fully coupled codes aimed at modeling coupled physics as a single problem, as opposed to multiple connected problems. The primary incentives for doing so are: (a) it is easier to obtain higher order convergence in time for fully coupled transient codes, and (b) full coupling reduces the number of iterations required for convergence of steady state codes [23]. One existing approach to full coupling is the "*bond graph formalism*".

The bond graph formalism is a technique for modeling engineering systems as combinations of connected elements. Bond graphs were originally introduced for mechatronics, but over time grew from a comprehensive methodology to model mechatronic systems into a complete research field, concerned with modeling mechanical, electrical, magnetic, hydraulic, thermal, and even optical and financial systems [1]. The primary objective of this work is to develop and analyze ways of application of the bond graph formalism to nuclear reactor multiphysics.

In this chapter, section 2.1 discusses, in detail, the physics of interest in transient nuclear reactor analysis. Next, section 2.2 summarizes existing reactor multiphysics simulation methods that are not based on bond graphs, and provides examples of practical commercial codes that use

these methods. Sections 2.3 and 2.4 provide the necessary background on bond graph formalism and existing bond graph applications to nuclear reactor analysis, respectively.

2.1 Nuclear Reactor Physics

This section presents the most general versions of the equations that describe the individual physics of interest, as well as their appropriate simplifications. The general versions are given here for completeness; only the simplified versions are usually used in reactor analysis. The effects of coupling on these equations are discussed separately. The section also contains summaries of the most commonly used discretization methods for the nuclear reactor physics equations.

2.1.1 Neutron Transport

This subsection details: 1) the general linear neutron transport equation (NTE), 2) the delayed neutron precursor equation (DNPE), 3) the steady state neutron transport eigenvalue problem, 4) the multigroup energy discretization methods, 5) the three important neutron transport problem types, 6) the spherical harmonics angular discretization methods, and 7) the discrete ordinates angular discretization methods.

2.1.1.1 Neutron Transport Equation

The basic assumptions made in neutron transport analysis of nuclear reactors are [6, Chapter 4]:

1. Neutrons can be treated as point particles with no wave-like quantum mechanical effects, which is valid on the macroscopic scale considered.
2. The neutron density is high enough for the deterministic approach to be valid. Neutron noise is present, but this assumption is generally valid in nuclear reactors, both at power and at cold shutdown.
3. There are no neutron-to-neutron interactions. This is valid because the neutron-to-neutron interaction potential, due to neutrons being neutral particles, is very weak, and atom density is orders of magnitude greater than the neutron density.
4. Neutron to nucleus collisions are well-defined 2-body events which occur instantaneously. This is an experimentally validated fact.
5. Between collisions, neutrons stream with constant velocity. This is valid because neutrons are neutral elementary particles, which only undergo weak nuclear, strong nuclear and gravitational interactions, and therefore are only measurably slowed down through collisions with nuclei.
6. Neutrons born through prompt fission or through the decay of delayed neutron precursors (i.e., certain fission products) are born isotropically in lab coordinate system (LCS). This assumption matches well with the available nuclear data [24].
7. Delayed neutron precursors are assumed to all be fission products. In reality, some neutron precursors may be also produced by neutron capture, but, because delayed neutron precursor

families are themselves an approximation, neglecting the capture-produced precursors does not increase uncertainty.

8. Photoneutron production is either not accounted for, or is incorporated in the external source. These neutrons are created through (γ, n) reactions of high energy γ particles (normally produced through slow decay of fission products) and certain nuclei (e.g., ^2H and ^9Be); these isotopes are only typically present in appreciable amounts in heavy water reactors, where their effect on kinetics does become important. They may also be considered in subcritical measurements and approaches to criticality. In such cases, high energy γ particle transport problem must be solved, to evaluate the rate of high energy γ incidence on the photoneutron-producing nuclei; again, here (γ, n) reactions are not treated explicitly.
9. Scattered neutrons' angular distributions are symmetric about the incident direction of travel. This is true for all scatterers except certain nonrandomly oriented (e.g., ferromagnetic) materials, which may bias the scattering; such materials are not present in a reactor.
10. The effects of coupling, for now, are omitted; they are added in subsection 2.1.5.

Under these assumptions, the neutron transport partial integro-differential equation (PIDE) becomes a linear, variable coefficient PIDE, given by (adapted from Refs. [2, 24]):

$$\begin{aligned}
 \frac{\partial}{\partial t} n(t, \vec{r}, E, \hat{\Omega}) &= -\nabla \cdot \hat{\Omega} \psi(t, \vec{r}, E, \hat{\Omega}) - \Sigma_t(t, \vec{r}, E) \psi(t, \vec{r}, E, \hat{\Omega}) + \\
 &+ \sum_{j=1}^J \left[\int_0^\infty dE' \iint_{4\pi} d\Omega' \Sigma_s^j(t, \vec{r}, E') P_s^j(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi(t, \vec{r}, E', \hat{\Omega}') \right] + \\
 &+ \frac{1}{4\pi} \sum_{j_f=1}^{J_f} \left[\int_0^\infty dE' \chi_p^{j_f}(E', E) \nu_p^{j_f}(E') \Sigma_f^{j_f}(t, \vec{r}, E') \iint_{4\pi} d\Omega' \psi(t, \vec{r}, E', \hat{\Omega}') \right] + \\
 &+ \sum_{j_y=1}^{J_y} \left[\sum_{y=1}^Y \sum_{\text{all } R_y} y \int_0^\infty dE' \iint_{4\pi} d\Omega' \left(\Sigma_{n,yn}^{j_y, R_y}(t, \vec{r}, E') \times \right. \right. \\
 &\quad \left. \left. \times P_{n,yn}^{j_y, R_y}(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi(t, \vec{r}, E', \hat{\Omega}') \right) \right] + \\
 &+ \frac{1}{4\pi} \sum_{j_f=1}^{J_f} \left[\sum_{m=1}^{M^{j_f}} \chi_{d,m}^{j_f}(E) \lambda_m^{j_f} c_m^{j_f}(t, \vec{r}) \right] + s_{ex}(t, \vec{r}, E, \hat{\Omega}).
 \end{aligned} \tag{2.1}$$

The following nomenclature is used:

t	= Time; an independent variable. Units: s.
\vec{r}	= Position vector in geometric space; a vector of 3 independent variables. Units: m.
E	= Neutron energy; an independent variable. Units: eV.
$\hat{\Omega}$	= Neutron direction unit vector; a 3-dimensional vector of 2 independent variables. It is defined in Eq. (2.2) below. Dimensionless.

- $n(t, \vec{r}, E, \hat{\Omega})$ = Energy-dependent angular neutron density. This is a density in a 6-dimensional phase space: in geometric space, in neutron energy and in neutron direction (i.e., in solid angle). This quantity can be related to $\psi(t, \vec{r}, E, \hat{\Omega})$, the unknown of Eq. (2.1), through Eq. (2.5) below. Units: neutrons/cm³ eV sr.
- $\psi(t, \vec{r}, E, \hat{\Omega})$ = Energy-dependent angular neutron flux density. It is defined in Eq. (2.5) below. It is the unknown in Eq. (2.1). Units: neutrons/cm² s eV sr.
- $\Sigma_t(t, \vec{r}, E)$ = Total macroscopic cross section. It is a derived, known quantity, defined by Eq. (2.6) below. Units: cm⁻¹.
- $\iint_{4\pi} d\Omega$ = Double integral over all target directions. It is defined in Eq. (2.3) below.
- $\Sigma_s^j(t, \vec{r}, E)$ = Scattering macroscopic cross section of nuclide j . It is a derived, known quantity, defined by Eq. (2.7) below. Units: cm⁻¹.
- $P_s^j(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega})$ = Double differential scattering probability distribution of nuclide j . This is a derived, known quantity, which is a sum of appropriate elastic, discrete inelastic and continuous inelastic energy-angle distributions, all of which are tabulated, known nuclide properties. It normalizes to 1. This is a double differential probability distribution in E and $\hat{\Omega}$, symmetric about the incident direction of travel $\hat{\Omega}'$ by Assumption 9, defined by Eq. (2.9) below. Units: 1/eV sr.
- j = Scattering nuclide index. Here a “nuclide” may be an isotope or a molecule.
- J = Number of scattering nuclides present in the reactor. This is a given, known quantity.
- $\Sigma_f^{j_f}(t, \vec{r}, E)$ = Fission macroscopic cross section of nuclide j_f . It is a derived, known quantity, defined by Eq. (2.7) below. Units: cm⁻¹.
- $\chi_p^{j_f}(E', E)$ = Prompt fission yield spectrum for fissionable nuclide j_f ; E' is the energy of the neutron that causes the fission and E is the energy of the generated prompt neutron. This is a tabulated, known nuclide property; it is a probability density in E which normalizes to 1. Units: 1/eV.
- $\nu_p^{j_f}(E)$ = Average number of prompt neutrons born when a nuclide j_f is fissioned by a neutron with energy E . This is a tabulated, known nuclide property. Dimensionless.
- j_f = Fissionable nuclide index. Here a nuclide may only be a fissionable isotope.
- J_f = Number of fissionable nuclides present in the reactor. This is a given, known quantity.
- $\Sigma_{n,yn}^{j_y,R_y}(t, \vec{r}, E)$ = (n, yn) reaction subtype R_y macroscopic cross section of nuclide j_y . y refers to the number of secondary neutrons produced in the reaction, and R_y refers to the reaction subtype; both quantities are discussed below. $\Sigma_{n,yn}^{j_y,R_y}(t, \vec{r}, E)$ is a derived, known quantity, defined by Eq. (2.8) below. Units: cm⁻¹.

- $P_{n,yn}^{j_y,R_y}(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega})$
 = Double differential secondary neutron yield probability distribution for (n, yn) reaction subtype R_y of nuclide j_y . Like $P_s^j(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega})$, this is a double differential probability distribution in E and $\hat{\Omega}$, symmetric about the incident direction of travel $\hat{\Omega}'$, which normalizes to 1. y here refers to the number of secondary neutrons produced in the reaction, and R_y refers to the reaction subtype; both quantities are discussed below. This is a tabulated, known nuclide property, defined by Eq. (2.10) below. Units: 1/eV sr.
- y
 = Number of secondary neutrons produced in an (n, yn) reaction, along with, possibly, other secondary particles. Note, that y may be 1: while an (n, n) reaction is a scattering, and not an (n, yn) reaction, a reaction that produces a single neutron along with other particles, like $(n, n\alpha)$, is treated as an $(n, 1n)$ reaction subtype.
- R_y
 = (n, yn) reaction subtype. For a given y , the subtypes differ by the non-neutron secondary particles produced: e.g., with $y = 2$, R_y may be the simple $(n, 2n)$, the more complicated $(n, 2n\alpha)$ reaction, or any other non-fission reaction with 2 secondary neutrons.
- j_y
 = Nuclide capable of undergoing an (n, yn) reaction index.
- J_y
 = Number of nuclides capable of undergoing (n, yn) reactions. This is a given, known quantity.
- Y
 = Maximum number of secondary neutrons that may be produced in (n, yn) reactions. This is a property of the nuclear database used; in ENDF/B-VII.1, $Y = 8$ [24].
- $\mathcal{C}_m^{j_f}(t, \vec{r})$
 = Density of delayed neutron precursors of family m produced by fissions of nuclide j_f . This is an unknown, discussed below. Units: precursors/cm³.
- $\chi_{d,m}^{j_f}(E)$
 = Delayed neutron yield spectrum for precursor family m produced by fissions of nuclide j_f . This is a tabulated, known nuclide property; like $\chi_p^{j_f}(E', E)$, it is a probability density in E which normalizes to 1. Units: 1/eV.
- $\lambda_m^{j_f}$
 = Decay constant of delayed neutron precursor family m produced by fissions of nuclide j_f . This is a tabulated, known nuclide property. Units: s⁻¹.
- m
 = Precursor family index.
- M^{j_f}
 = Number of precursor families produced by fissions of nuclide j_f . This is a tabulated, known nuclide property.
- $s_{ex}(t, \vec{r}, E, \hat{\Omega})$
 = Energy-dependent external angular neutron source density function. This is a given, known quantity; it is frequently negligible for reactors at power. Units: neutrons/cm³ s eV sr.

The following equations define the direction unit vector (as a Cartesian 3-dimensional vector) and the double integral over all directions:

$$\begin{aligned}\hat{\Omega} &= \cos(\theta)\hat{x} + \sin(\theta)\cos(\varphi)\hat{y} + \sin(\theta)\sin(\varphi)\hat{z} = \\ &= \mu\hat{x} + \eta\hat{y} + \xi\hat{z},\end{aligned}\tag{2.2}$$

$$\iint_{4\pi} d\Omega = \int_0^{2\pi} d\varphi \int_0^\pi d\theta \sin(\theta).\tag{2.3}$$

Here, the following nomenclature was used:

- θ = Polar (or colatitude) angle of the direction unit vector with respect to the x -axis. It varies from 0 to π rad.
- φ = Azimuthal angle of the direction unit vector on the yz -plane. It varies from 0 to 2π rad.
- μ, η, ξ = Cartesian direction cosines, defined by Eqs. (2.4). They each vary from -1 to 1 .
- $\hat{x}, \hat{y}, \hat{z}$ = Cartesian base unit vectors.

Direction cosines in Eq. (2.2) are given by:

$$\mu = \cos(\theta),\tag{2.4a}$$

$$\eta = \sin(\theta)\cos(\varphi),\tag{2.4b}$$

$$\xi = \sin(\theta)\sin(\varphi).\tag{2.4c}$$

The following equations define the derived quantities used in Eq. (2.1):

$$\psi(t, \vec{r}, E, \hat{\Omega}) = V_n(E) n(t, \vec{r}, E, \hat{\Omega}) = \sqrt{\frac{2E}{m_n}} n(t, \vec{r}, E, \hat{\Omega}),\tag{2.5}$$

$$\Sigma_R(t, \vec{r}, E) = \sum_{j=1}^J \Sigma_R^j(t, \vec{r}, E),\tag{2.6}$$

$$\Sigma_R^j(t, \vec{r}, E) = N^j(t, \vec{r}) \sigma_R^j(E),\tag{2.7}$$

$$\Sigma_{n,yn}^{jy,Ry}(t, \vec{r}, E) = N^{jy}(t, \vec{r}) \sigma_{n,yn}^{jy,Ry}(E),\tag{2.8}$$

$$P_s^j(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega}) = \frac{1}{2\pi} P_s^j(E' \rightarrow E, \mu_s),\tag{2.9}$$

$$P_{n,yn}^{jy,Ry}(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega}) = \frac{1}{2\pi} P_{n,yn}^{jy,Ry}(E' \rightarrow E, \mu_s),\tag{2.10}$$

$$\mu_s = \hat{\Omega}' \cdot \hat{\Omega}.\tag{2.11}$$

Here, the following nomenclature was used:

- $V_n(E)$ = Speed of a neutron with energy E . This quantity is often called “neutron velocity.”
Units: cm/s.
- m_n = Mass of a neutron, 939.565379(21) MeV/ c^2 [25].
- $\Sigma_R(t, \vec{r}, E)$ = Reaction type R macroscopic cross section. Units: cm^{-1} .

- $\Sigma_R^j(t, \vec{r}, E)$ = Reaction type R macroscopic cross section of nuclide j . Units: cm^{-1} .
- $N^j(t, \vec{r})$ = Number density of nuclide j . This is a given quantity which is specified as part of the reactor geometry and material/fluid composition. The time dependence here can be used to model the movement of reactor components, such as the control assemblies. Units: nuclides/ cm^3 .
- $\sigma_R^j(E)$ = Reaction type R microscopic cross section of nuclide j . This is a tabulated, known nuclide property. Units: $\text{b} = 1 \times 10^{-24} \text{ cm}^2$.
- $\sigma_{n,yn}^{jy,Ry}(E)$ = (n, yn) reaction subtype R_y microscopic cross section of nuclide j_y . This is a tabulated, known nuclide property. Units: $\text{b} = 1 \times 10^{-24} \text{ cm}^2$.
- $P_s^j(E' \rightarrow E, \mu_s)$ = Double differential scattering probability distribution in target energy E and scattering angle cosine μ_s of nuclide j . This is a sum of appropriate elastic, discrete inelastic and continuous inelastic energy-scattering angle cosine distributions, all of which are tabulated, known properties. Scattered neutron angular (and energy-angle) probability distributions' angular dependencies are tabulated as functions of μ_s in nuclear databases; μ_s may be in lab (LCS) or center of mass (CMCS) coordinate system. If it is in CMCS, it must first be converted to LCS. A scattering angle cosine ranges from -1 to 1 , with the scattering probability cone for a given μ_s being symmetric about the incident direction of travel $\hat{\Omega}'$, which is where the $1/2\pi$ factor comes from. $P_s^j(E' \rightarrow E, \mu_s)$ normalizes to 1. Units: $1/\text{eV}$ unit-cosine.
- $P_{n,yn}^{jy,Ry}(E' \rightarrow E, \mu_s)$ = Double differential secondary neutron yield probability distribution in target energy E and incident-to-secondary direction angle cosine μ_s for (n, yn) reaction subtype R_y of nuclide j_y . Like $P_s^j(E' \rightarrow E, \mu_s)$, this quantity is a tabulated (as function of μ_s), known property, with the secondary neutron's direction probability cone for a given μ_s symmetric about $\hat{\Omega}'$, which is where the $1/2\pi$ factor comes from. It normalizes to 1. Units: $1/\text{eV}$ unit-cosine.
- μ_s = In scattering reactions, μ_s is the scattering angle cosine. In (n, yn) reactions, this is the cosine of the angle between the incident and secondary neutrons' direction vectors. It ranges from -1 to 1 . Dimensionless.

An energy-dependent scalar flux density $\phi(t, \vec{r}, E)$ is implicitly present in Eq. (2.1). It, and the scalar flux $\phi(t, \vec{r})$ are given by:

$$\phi(t, \vec{r}, E) = \iint_{4\pi} d\Omega \psi(t, \vec{r}, E, \hat{\Omega}), \quad (2.12a)$$

$$\phi(t, \vec{r}) = \int_0^\infty dE \iint_{4\pi} d\Omega \psi(t, \vec{r}, E, \hat{\Omega}) = \int_0^\infty dE \phi(t, \vec{r}, E). \quad (2.12b)$$

Equation (2.1) is an energy-dependent, angular reaction rate density (RRD) balance. Four types of RRDs of reaction type R can be postulated:

$w_R(t, \vec{r}, E, \hat{\Omega})$ = Energy-dependent, angular type R RRD. Units: reactions/cm³ s eV sr.

$w_R(t, \vec{r}, \hat{\Omega})$ = Angular type R RRD. Units: reactions/cm³ s sr.

$W_R(t, \vec{r}, E)$ = Energy-dependent type R RRD. Units: reactions/cm³ s eV.

$W_R(t, \vec{r})$ = Type R RRD. Units: reactions/cm³ s.

They are given by Eqs. (2.13):

$$w_R(t, \vec{r}, E, \hat{\Omega}) = \Sigma_R(t, \vec{r}, E) \psi(t, \vec{r}, E, \hat{\Omega}), \quad (2.13a)$$

$$w_R(t, \vec{r}, \hat{\Omega}) = \int_0^\infty dE \Sigma_R(t, \vec{r}, E) d\Omega \psi(t, \vec{r}, E, \hat{\Omega}), \quad (2.13b)$$

$$W_R(t, \vec{r}, E) = \Sigma_R(t, \vec{r}, E) \iint_{4\pi} d\Omega \psi(t, \vec{r}, E, \hat{\Omega}) = \Sigma_R(t, \vec{r}, E) \phi(t, \vec{r}, E), \quad (2.13c)$$

$$W_R(t, \vec{r}) = \int_0^\infty dE \Sigma_R(t, \vec{r}, E) \iint_{4\pi} d\Omega \psi(t, \vec{r}, E, \hat{\Omega}) = \int_0^\infty dE \Sigma_R(t, \vec{r}, E) \phi(t, \vec{r}, E). \quad (2.13d)$$

All of the quantities in Eqs. (2.1)–(2.13) are either the unknowns, independent variables, known given (as part of the problem statement) quantities, known tabulated nuclide properties or quantities derived from the others using Eqs. (2.5)–(2.13). The known, tabulated nuclide properties are all provided in nuclear databases, such as ENDF/B-VII.1 [24]. It should be noted that, while all nuclides are capable of radiative capture and scattering reactions, not all nuclides are capable of, e.g., fission or $(n, n\alpha)$ reactions. Fundamentally, a nuclide j being incapable of undergoing a reaction of type R simply means $\sigma_R^j(E) = 0$, however, it would be inefficient to include this data in the reaction rate calculations: many additional zero terms would have to be evaluated prior to being summed. For this reason, when considering individual nuclides' contributions to RRDs, only the nuclides able to undergo the reaction are accounted for.

This completes the summary of the linear NTE. Next, the delayed neutron precursor concentration term $c_m^j(t, \vec{r})$ is discussed.

2.1.1.2 Delayed Neutron Precursor Equation

Some of the fission product nuclei are capable of decaying by neutron emission. Such nuclei are called “delayed neutron precursors,” and play an important role in neutron kinetics. There are many such nuclei, they have different degrees of stability, and therefore all have different, by many orders of magnitude, decay constants. Because it would be prohibitively costly and unnecessary to keep track of the spatial profiles of every precursor nuclide type, the delayed neutron precursors are commonly grouped in “precursor families.” All nuclides in a precursor family share the decay constant and the delayed neutron yield spectrum.

Strictly, the decay constants of precursor families created by fissions of different fissionable nuclides differ slightly. For this reason, in ENDF/B-VII.1, this difference is accounted for: each

fissionable isotope's 6 precursor families have different decay constants and delayed neutron yield spectra [24]. Other nuclear databases, such as JEFF-3.1, instead use the same precursor families for all fissionable isotopes. JEFF-3.1 uses 8 precursor families [26]. In this subsection, the more general case of isotope-specific precursor families is treated. In practice, if a database like ENDF/B-VII.1 is used, and more than one fissionable isotope is present (e.g., ^{235}U and ^{238}U), it is generally recommended to assume a common set of precursor families, and to use the precursor family characteristics from the principal fissioning isotope with nuclide-specific delayed neutron precursor fractions.

To model the delayed neutron precursor dynamics, the following assumptions are normally made [6, Chapter 6]:

1. NTE assumption 7 (p. 23), which states that delayed neutron precursors are assumed to all be fission products.
2. The fissionable materials being modeled are treated as stationary: delayed neutron precursor nuclei cannot move. This is generally true for all reactors with solid stationary fuel (excluding meltdown and atomic diffusion), but not true for reactors with moving fuel, such as molten fuel salt reactors.

Under these assumptions, the linear delayed neutron precursor equation (DNPE) becomes (adapted from Ref. [4, Chapter 3]):

$$\begin{aligned} \frac{\partial}{\partial t} c_m^{j_f}(t, \vec{r}) &= \int_0^\infty dE' \nu_{d,m}^{j_f}(E') \Sigma_f^{j_f}(t, \vec{r}, E') \iint_{4\pi} d\Omega' \psi(t, \vec{r}, E', \hat{\Omega}') - \\ &\quad - \lambda_m^{j_f} c_m^{j_f}(t, \vec{r}) \quad \forall m \in [1, \dots, M^{j_f}], j_f \in [1, \dots, J_f]. \end{aligned} \quad (2.14)$$

The following nomenclature was used here:

$\nu_{d,m}^{j_f}(E)$ = Average number of delayed neutron precursors created in precursor family m when a nuclide j_f is fissioned by a neutron with energy E . This is a tabulated, known nuclide property. Dimensionless.

It is often convenient to express the prompt neutron and precursor generation rates in terms of delayed neutron fractions (DNFs) and total numbers of secondary neutrons:

$$\nu^{j_f}(E) = \nu_p^{j_f}(E) + \sum_{m=1}^{M^{j_f}} \nu_{d,m}^{j_f}(E), \quad (2.15)$$

$$\beta_m^{j_f}(E) = \frac{\nu_{d,m}^{j_f}(E)}{\nu^{j_f}(E)}, \quad (2.16)$$

$$\beta^{j_f}(E) = \sum_{m=1}^{M^{j_f}} \beta_m^{j_f}(E), \quad (2.17)$$

in which:

$\nu^{jf}(E)$ = Average total number of secondary neutrons (prompt and delayed) produced by a fission of nuclide j_f by a neutron with energy E . Dimensionless.

$\beta_m^{jf}(E)$ = DNF of precursor family m produced by fissions of nuclide j_f . Dimensionless.

$\beta^{jf}(E)$ = Total DNF of nuclide j_f . Dimensionless.

To use the DNFs in Eqs. (2.1) and (2.14), we can simply substitute the numbers of prompt and delayed secondary neutrons:

$$\nu_p^{jf}(E) = (1 - \beta^{jf}(E)) \nu^{jf}(E), \quad (2.18a)$$

$$\nu_{d,m}^{jf}(E) = \beta_m^{jf}(E) \nu^{jf}(E). \quad (2.18b)$$

This completes the summary of the linear DNPE. Together, under the given assumptions, Eqs. (2.1) and (2.14) form the most general model for the spatial nuclear reactor neutron kinetics. Next, the steady state simplification of the linear NTE is discussed.

2.1.1.3 Steady State Neutron Transport Eigenproblem

Equation (2.1) is time-dependent. When a reactor is at steady state, with a nonzero external source $s_{ex}(\vec{r}, E, \hat{\Omega})$, the problem becomes a linear, inhomogeneous, 6-dimensional boundary value problem, which is solved by discretizing the terms and solving the resulting algebraic linear system. A nonzero external source here means that it is nonzero in at least part of the phase 6-space. The delayed neutron precursors can be simplified out, because the varying decay rates of precursor families are not important in a steady reactor; the delayed neutrons' influence on the fission yield spectrum can be modeled without explicitly calculating the precursor densities. With these modifications, the steady state neutron transport equation with a nonzero external source becomes (adapted from Ref. [4, Chapter 3]):

$$\begin{aligned} & \nabla \cdot \hat{\Omega} \psi(\vec{r}, E, \hat{\Omega}) + \Sigma_t(\vec{r}, E) \psi(\vec{r}, E, \hat{\Omega}) - \\ & - \sum_{j=1}^J \left[\int_0^\infty dE' \iint_{4\pi} d\Omega' \Sigma_s^j(\vec{r}, E') P_s^j(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi(\vec{r}, E', \hat{\Omega}') \right] - \\ & - \sum_{j_y=1}^{J_y} \left[\sum_{y=1}^Y \sum_{\text{all } R_y} y \int_0^\infty dE' \iint_{4\pi} d\Omega' \left(\Sigma_{n,y_n}^{j_y, R_y}(\vec{r}, E') \times \right. \right. \\ & \quad \left. \left. \times P_{n,y_n}^{j_y, R_y}(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi(\vec{r}, E', \hat{\Omega}') \right) \right] = \\ & = \frac{1}{4\pi} \sum_{j_f=1}^{J_f} \left[\int_0^\infty dE' \chi_{ss}^{j_f}(E', E) \nu^{j_f}(E') \Sigma_f^{j_f}(\vec{r}, E') \iint_{4\pi} d\Omega' \psi(\vec{r}, E', \hat{\Omega}') \right] + s_{ex}(\vec{r}, E, \hat{\Omega}). \end{aligned} \quad (2.19)$$

Here, the steady state fission yield spectrum $\chi_{ss}^{jf}(E', E)$ is given by:

$$\chi_{ss}^{jf}(E', E) = \left(1 - \beta^{jf}(E')\right) \chi_p^{jf}(E', E) + \sum_{m=1}^{M^{jf}} \left[\beta_m^{jf}(E') \chi_{d,m}^{jf}(E) \right], \quad (2.20)$$

in which:

$\chi_{ss}^{jf}(E', E)$ = Steady state fission yield spectrum; E' is the energy of the neutron that causes the fission and E is the energy of the generated neutron. Units: 1/eV.

If the external source is zero, which is effectively the case for a reactor at power, the problem becomes homogeneous. For this problem to have a nontrivial solution, the determinant of the operator must be zero, which is not generally true. To make it zero, an eigenvalue that would modify one of the terms has to be introduced. Several types of eigenvalues may be used for the linear NTE (they divide different terms in Eq. (2.19)), but in reactor analysis, the multiplication constant is the most common type [5, Chapter 3]. The linear steady state neutron transport eigenproblem is given by (adapted from Ref. [4, Chapter 3]):

$$\begin{aligned} & \nabla \cdot \hat{\Omega} \psi(\vec{r}, E, \hat{\Omega}) + \Sigma_t(\vec{r}, E) \psi(\vec{r}, E, \hat{\Omega}) - \\ & - \sum_{j=1}^J \left[\int_0^\infty dE' \iint_{4\pi} d\Omega' \Sigma_s^j(\vec{r}, E') P_s^j(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi(\vec{r}, E', \hat{\Omega}') \right] - \\ & - \sum_{j_y=1}^{J_y} \left[\sum_{y=1}^Y \sum_{\text{all } R_y} y \int_0^\infty dE' \iint_{4\pi} d\Omega' \left(\Sigma_{n,y}^{j_y, R_y}(\vec{r}, E') \times \right. \right. \\ & \quad \left. \left. \times P_{n,y}^{j_y, R_y}(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi(\vec{r}, E', \hat{\Omega}') \right) \right] = \\ & = \frac{1}{k_{eff}} \frac{1}{4\pi} \sum_{j_f=1}^{J_f} \left[\int_0^\infty dE' \chi_{ss}^{j_f}(E', E) \nu^{j_f}(E') \Sigma_f^{j_f}(\vec{r}, E') \iint_{4\pi} d\Omega' \psi(\vec{r}, E', \hat{\Omega}') \right], \end{aligned} \quad (2.21)$$

in which:

k_{eff} = Reactor eigenvalue, also known as reactor multiplication constant. A reactor with $k_{eff} = 1$ is critical; a reactor with $k_{eff} > 1$ or $k_{eff} < 1$ is supercritical and subcritical, respectively. Dimensionless.

The solution of Eq. (2.21) is known as ‘‘eigenvalue search.’’ Equation (2.21) can be rewritten as follows:

$$\hat{\mathcal{M}} \psi(\vec{r}, E, \hat{\Omega}) = \frac{1}{k_{eff}} \hat{\mathcal{F}} \psi(\vec{r}, E, \hat{\Omega}), \quad (2.22)$$

in which:

$\hat{\mathcal{M}}$ = Net removal operator. It accounts for streaming, all collisions, inscattering and (n, yn) production.

$\hat{\mathcal{F}}$ = Fission operator. It accounts for neutron generation due to fission.

Equation (2.22) is known as a “generalized eigenproblem,” where k_{eff} is the eigenvalue and $\psi(\vec{r}, E, \hat{\Omega})$ the eigenfunction. The discretized version of $\psi(\vec{r}, E, \hat{\Omega})$ is the discretized problem eigenvector.

Unlike Eqs. (2.1) or (2.19), Eq. (2.22) is not a closed system: any multiple of the solution eigenfunction is also a solution. To close the system, total reactor power is usually specified.

Besides finding the steady state of a system, the eigenproblem can also be used to identify the “critical state” of the reactor, which, due to the thermohydraulic state, may not be a true steady state. A critical flux eigenfunction, together with a specified thermohydraulic state is often the initial condition for a transient, and, in order for the transient to develop correctly, the initial eigenvalue must stay in the transient NTE and DNPEs, which become:

$$\begin{aligned}
 \frac{\partial}{\partial t} n(t, \vec{r}, E, \hat{\Omega}) &= -\nabla \cdot \hat{\Omega} \psi(t, \vec{r}, E, \hat{\Omega}) - \Sigma_t(t, \vec{r}, E) \psi(t, \vec{r}, E, \hat{\Omega}) + \\
 &+ \sum_{j=1}^J \left[\int_0^\infty dE' \iint_{4\pi} d\Omega' \Sigma_s^j(t, \vec{r}, E') P_s^j(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi(t, \vec{r}, E', \hat{\Omega}') \right] + \\
 &+ \frac{1}{k_{eff}^0} \frac{1}{4\pi} \sum_{j_f=1}^{J_f} \left[\int_0^\infty dE' \chi_p^{j_f}(E', E) \nu_p^{j_f}(E') \Sigma_f^{j_f}(t, \vec{r}, E') \iint_{4\pi} d\Omega' \psi(t, \vec{r}, E', \hat{\Omega}') \right] + \\
 &+ \sum_{j_y=1}^{J_y} \left[\sum_{y=1}^Y \sum_{\text{all } R_y} y \int_0^\infty dE' \iint_{4\pi} d\Omega' \left(\Sigma_{n,yn}^{j_y, R_y}(t, \vec{r}, E') \times \right. \right. \\
 &\quad \left. \left. \times P_{n,yn}^{j_y, R_y}(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi(t, \vec{r}, E', \hat{\Omega}') \right) \right] + \\
 &+ \frac{1}{4\pi} \sum_{j_f=1}^{J_f} \left[\sum_{m=1}^{M^{j_f}} \chi_{d,m}^{j_f}(E) \lambda_m^{j_f} c_m^{j_f}(t, \vec{r}) \right] + s_{ex}(t, \vec{r}, E, \hat{\Omega}),
 \end{aligned} \tag{2.23}$$

and

$$\begin{aligned}
 \frac{\partial}{\partial t} c_m^{j_f}(t, \vec{r}) &= \frac{1}{k_{eff}^0} \int_0^\infty dE' \nu_{d,m}^{j_f}(E') \Sigma_f^{j_f}(t, \vec{r}, E') \iint_{4\pi} d\Omega' \psi(t, \vec{r}, E', \hat{\Omega}') - \\
 &- \lambda_m^{j_f} c_m^{j_f}(t, \vec{r}) \quad \forall m \in [1, \dots, M^{j_f}], j_f \in [1, \dots, J_f],
 \end{aligned} \tag{2.24}$$

where:

k_{eff}^0 = Reactor eigenvalue that makes the initial condition a critical reactor. It, effectively, divides the prompt neutron and delayed neutron precursor production terms in Eqs. (2.23) and (2.24), respectively, and stays in them throughout the transient. It is omitted in the discretized neutron physics equations below, but may divide the corresponding terms if the equations are used to model transients with critical initial conditions. An external source may be present in such problems; it must be omitted for the criticality search, otherwise the problem is a fixed source, and not an eigenvalue problem, and an eigenvalue cannot be defined. Dimensionless.

This concludes the summary of the steady state versions of the linear NTE. To be solved, both the transient and steady state versions of the NTE (and, for transient problems, also the DNPE) have to be discretized. The most common energy discretization technique is the multigroup energy discretization method, summarized in the following subsection.

2.1.1.4 Multigroup Energy Discretization Methods

Below, energy discretization of the transient NTE and DNPE is discussed. Steady state NTE can be discretized using exactly the same methods.

To discretize Eqs. (2.1) and (2.14) in energy, the most common approach is multigroup energy condensation. Fundamentally, multigroup methods work by assuming a maximum reachable neutron energy E_0 and separating the energy domain into G energy groups, as shown in Figure 2.1.

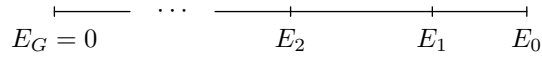


Figure 2.1: Energy Domain Discretization

Once the energy domain is discretized into groups, the energy condensation procedure aims to arrive at “group properties.” Group neutron densities, fluxes and external sources are expressed as follows:

$$n_g(t, \vec{r}, \hat{\Omega}) = \int_{E_g}^{E_{g-1}} dE n(t, \vec{r}, E, \hat{\Omega}), \quad (2.25a)$$

$$\psi_g(t, \vec{r}, \hat{\Omega}) = \int_{E_g}^{E_{g-1}} dE \psi(t, \vec{r}, E, \hat{\Omega}), \quad (2.25b)$$

$$\phi_g(t, \vec{r}) = \int_{E_g}^{E_{g-1}} dE \phi(t, \vec{r}, E), \quad (2.25c)$$

$$s_{exg}(t, \vec{r}, \hat{\Omega}) = \int_{E_g}^{E_{g-1}} dE s_{ex}(t, \vec{r}, E, \hat{\Omega}), \quad (2.25d)$$

in which:

E_g = Energy bound between groups g and $g + 1$. See Figure 2.1. Units: eV.

$n_g(t, \vec{r}, \hat{\Omega})$ = Group g angular neutron density. Units: neutrons/cm³ sr.

$\psi_g(t, \vec{r}, \hat{\Omega})$ = Group g angular neutron flux. Units: neutrons/cm² sr.

$\phi_g(t, \vec{r})$ = Group g scalar neutron flux. Units: neutrons/cm² s.

$s_{exg}(t, \vec{r}, \hat{\Omega})$ = Group g angular external source function. Units: neutrons/cm³ s sr.

Group properties are flux-weighted integrals of energy-dependent properties. Four types of group properties can be outlined: (a) group cross sections, (b) group-to-group cross sections, (c) group velocities, and (d) yield fractions. Group cross sections are intragroup averages of corresponding cross sections, weighted by either the energy-dependent angular or scalar flux densities. If a nonseparable angular weighting flux $\psi(t, \vec{r}, E, \hat{\Omega})$ is used, it introduces an angular dependence into the group cross section, even if the corresponding energy-dependent cross section was not direction-dependent. It may also introduce flux-based time and position dependence. This is a potential issue for some angular discretization methods (e.g., the spherical harmonics method in subsection 2.1.1.6), and so a region-specific separable weighting flux is often assumed:

$$\psi(t, \vec{r}, E, \hat{\Omega}) \cong \psi_g(t, \vec{r}, \hat{\Omega}) \Psi_g^r(E) \quad \text{with } \vec{r} \in \mathcal{D}_r, \quad (2.26)$$

in which:

$\Psi_g^r(E)$ = Group g neutron energy spectrum in region r . Units: eV⁻¹.

\mathcal{D}_r = Spatial domain of the geometric region r , in which a constant neutron energy spectrum is assumed.

$\Psi_g^r(E)$ is typically computed as a spatial and angular average of the steady angular flux (obtained by solving the transport eigenproblem) in the region of interest. With this method, its shape in energy is identical to that of the spatially averaged scalar flux, by Eq. (2.12). This does not have to be the case: in principle, any weighting spectrum may be used in group property construction, although the choice of the spectrum will affect the quality of the group properties produced.

Group cross sections are defined by Eqs. (2.27):

$$\begin{aligned} w_{Rg}(t, \vec{r}, \hat{\Omega}) &= \Sigma_{Rg}(t, \vec{r}, \hat{\Omega}) \psi_g(t, \vec{r}, \hat{\Omega}) = \\ &= \int_{E_g}^{E_{g-1}} dE \Sigma_R(t, \vec{r}, E) \psi(t, \vec{r}, E, \hat{\Omega}) \quad \text{with nonseparable flux,} \end{aligned} \quad (2.27a)$$

$$\begin{aligned} w_{Rg}(t, \vec{r}, \hat{\Omega}) &= \Sigma_{Rg}(t, \vec{r}) \psi_g(t, \vec{r}, \hat{\Omega}) = \\ &= \int_{E_g}^{E_{g-1}} dE \Sigma_R(t, \vec{r}, E) \psi(t, \vec{r}, E, \hat{\Omega}) = \\ &= \left[\int_{E_g}^{E_{g-1}} dE \Sigma_R(t, \vec{r}, E) \Psi_g^r(E) \right] \psi_g(t, \vec{r}, \hat{\Omega}) \quad \text{with } \vec{r} \in \mathcal{D}_r \text{ and separable flux,} \end{aligned} \quad (2.27b)$$

$$\begin{aligned}
 W_{Rg}(t, \vec{r}) &= \Sigma_{Rg}(t, \vec{r}) \phi_g(t, \vec{r}) = \\
 &= \int_{E_g}^{E_{g-1}} dE \Sigma_R(t, \vec{r}, E) \phi(t, \vec{r}, E) \quad \text{with scalar flux.}
 \end{aligned} \tag{2.27c}$$

Here the following nomenclature was used:

$w_{Rg}(t, \vec{r}, \hat{\Omega})$ = Angular type R group g RRD. Units: reactions/cm³ s sr.

$\Sigma_{Rg}(t, \vec{r}, \hat{\Omega})$ = Reaction type R group g macroscopic cross section, weighted by the nonseparable angular flux. Together with group g angular flux, this quantity characterizes the type R angular reaction rate density in group g . Note the angular dependence, which arises due to the nonseparable angular flux weighting. In Eq. (2.1), the total reaction rate can be quantified using the total macroscopic group cross section; the energy-dependent cross section here is $\Sigma_t(t, \vec{r}, E)$. Units: cm⁻¹.

$W_{Rg}(t, \vec{r})$ = Type R group g RRD. Units: reactions/cm³ s.

$\Sigma_{Rg}(t, \vec{r})$ = Reaction type R group g macroscopic cross section, weighted by either the separable angular, or the scalar flux. Together with group g separable angular, or the scalar flux, this quantity characterizes the type R angular or scalar reaction rate density in group g . In Eq. (2.1), the total reaction rate can be quantified using the total macroscopic group cross section (weighted by the separable angular flux); the energy-dependent cross section here is $\Sigma_t(t, \vec{r}, E)$. In Eq. (2.14), the delayed neutron precursor generation rate can be quantified using the ‘‘precursor production macroscopic group cross section’’; a ‘‘precursor production cross section’’ is the product $\nu_{d,m}^{jf}(E) \Sigma_f^{jf}(t, \vec{r}, E)$. Units: cm⁻¹.

Group-to-group cross sections are given by Eqs. (2.28):

$$\begin{aligned}
 \Sigma_{Rgg'}(t, \vec{r}, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi_{g'}(t, \vec{r}, \hat{\Omega}') &= \\
 &= \int_{E_g}^{E_{g-1}} dE \int_{E_{g'}}^{E_{g'-1}} dE' \Sigma_R(t, \vec{r}, E') P_R(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi(t, \vec{r}, E', \hat{\Omega}'),
 \end{aligned} \tag{2.28a}$$

$$\begin{aligned}
 \Sigma_{Rgg'}(t, \vec{r}) \phi_{g'}(t, \vec{r}) &= \\
 &= \int_{E_g}^{E_{g-1}} dE \int_{E_{g'}}^{E_{g'-1}} dE' \Sigma_R(t, \vec{r}, E') P_R(E' \rightarrow E) \phi(t, \vec{r}, E'),
 \end{aligned} \tag{2.28b}$$

in which:

$$\Sigma_{Rgg'}(t, \vec{r}, \hat{\Omega}' \rightarrow \hat{\Omega})$$

= Reaction type R group g' to g (i.e., $g' \rightarrow g$) angular differential macroscopic cross section. This quantity characterizes the rate of neutron appearance in target group g as a function of type R angular reaction rate density in source group g' . In Eq. (2.1), the scattering and (n, yn) contributions to neutron appearance rates at target energies can be quantified using the group-to-group scattering and group-to-group (n, yn) cross sections, respectively. The energy-dependent cross sections $\Sigma_R(t, \vec{r}, E)$ for these two reaction types are $\Sigma_s^j(t, \vec{r}, E)$ (scattering) and $\Sigma_{n,yn}^{jy,Ry}(t, \vec{r}, E)$ (n, yn), respectively. Units: cm^{-1}/sr .

$$P_R(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega})$$

= Reaction type R double differential probability distribution. For scattering and (n, yn) , these are the $P_s^j(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega})$ and $P_{n,yn}^{jy,Ry}(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega})$ probability distributions, respectively. Units: $1/\text{eV sr}$.

$\Sigma_{Rgg'}(t, \vec{r})$ = Reaction type R group g' to g (i.e., $g' \rightarrow g$) macroscopic cross section. This quantity characterizes the rate of neutron appearance in target group g as a function of type R reaction rate density in source group g' . In Eq. (2.1), the fission contributions to prompt neutron appearance rates at target energies can be quantified using the group-to-group prompt fission production cross section. The energy-dependent cross section $\Sigma_R(t, \vec{r}, E)$ for this reaction type is $\nu_p^{jf}(E) \Sigma_f^{jf}(t, \vec{r}, E)$ (prompt neutron fission production). Units: cm^{-1} .

$P_R(E' \rightarrow E)$ = Reaction type R single differential probability distribution. For prompt neutron fission production, this is the prompt fission yield spectrum $\chi_p^{jf}(E', E)$. Units: $1/\text{eV}$.

Group velocities, like group cross sections, can gain flux-based direction and position dependence when weighted by a nonseparable angular flux, which may adversely affect certain angular discretization methods. For this reason, separable flux (Eq. (2.26)) can be assumed for their calculation. They are defined by Eqs. (2.29):

$$\psi_g(t, \vec{r}, \hat{\Omega}) = V_{ng}(t, \vec{r}, \hat{\Omega}) n_g(t, \vec{r}, \hat{\Omega}) \quad \text{with nonseparable flux,} \quad (2.29a)$$

$$\psi_g(t, \vec{r}, \hat{\Omega}) = V_{ng}^r n_g(t, \vec{r}, \hat{\Omega}) \quad \text{with } \vec{r} \in \mathcal{D}_r \text{ and separable flux,} \quad (2.29b)$$

in which:

$V_{ng}(t, \vec{r}, \hat{\Omega})$ = Group g neutron velocity, weighted by nonseparable angular flux. Units: cm/s .

V_{ng}^r = Group g region r neutron velocity, weighted by separable angular flux. Units: cm/s .

Yield fractions are defined by Eq. (2.30):

$$\chi_{Rg} = \int_{E_g}^{E_{g-1}} dE \chi_{Rg}(E), \quad (2.30)$$

in which:

$\chi_{Rg}(E)$ = Reaction type R energy-dependent yield spectrum. Units: 1/eV.

χ_{Rg} = Reaction type R group g yield fraction. In Eq. (2.1), the delayed neutron generation rate at target energies can be quantified using the delayed neutron group yield fraction. The energy-dependent yield spectrum here is $\chi_{d,m}^{jf}(E)$. Dimensionless.

Once the group properties are constructed using the more general nonseparable weighting fluxes, the multigroup linear neutron transport equation becomes (adapted from Ref. [4, Chapter 3]):

$$\begin{aligned} \frac{\partial}{\partial t} n_g(t, \vec{r}, \hat{\Omega}) &= -\nabla \cdot \hat{\Omega} \psi_g(t, \vec{r}, \hat{\Omega}) - \Sigma_{tg}(t, \vec{r}, \hat{\Omega}) \psi_g(t, \vec{r}, \hat{\Omega}) + \\ &+ \sum_{j=1}^J \sum_{g'=1}^G \left[\iint_{4\pi} d\Omega' \Sigma_{sgg'}^j(t, \vec{r}, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi_{g'}(t, \vec{r}, \hat{\Omega}') \right] + \\ &+ \frac{1}{4\pi} \sum_{j_f=1}^{J_f} \sum_{g'=1}^G \left[\chi_{\nu_p \Sigma_{fgg'}^{j_f}}(t, \vec{r}) \iint_{4\pi} d\Omega' \psi_{g'}(t, \vec{r}, \hat{\Omega}') \right] + \\ &+ \sum_{j_y=1}^{J_y} \sum_{y=1}^Y \sum_{\text{all } R_y} \sum_{g'=1}^G \left[y \iint_{4\pi} d\Omega' \Sigma_{n,yngg'}^{j_y, R_y}(t, \vec{r}, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi_{g'}(t, \vec{r}, \hat{\Omega}') \right] + \\ &+ \frac{1}{4\pi} \sum_{j_f=1}^{J_f} \left[\sum_{m=1}^{M^{j_f}} \chi_{dg,m}^{j_f} \lambda_m^{j_f} c_m^{j_f}(t, \vec{r}) \right] + s_{exg}(t, \vec{r}, \hat{\Omega}). \end{aligned} \quad (2.31)$$

The following notation was used here:

$\Sigma_{tg}(t, \vec{r}, \hat{\Omega})$ = Group g total macroscopic cross section, weighted by nonseparable angular flux. Units: cm^{-1} .

g' = Source energy group index.

G = Number of energy groups.

$\Sigma_{sgg'}^j(t, \vec{r}, \hat{\Omega}' \rightarrow \hat{\Omega})$
= Group g' to g angular differential macroscopic scattering cross section for nuclide j . Units: cm^{-1} .

$\chi_{\nu_p \Sigma_{fgg'}^{j_f}}(t, \vec{r})$ = Group g' to g macroscopic prompt fission production cross section for nuclide j_f . Units: cm^{-1} .

- $\Sigma_{n,yn}^{j_y, R_y}(t, \vec{r}, \hat{\Omega}' \rightarrow \hat{\Omega})$
 = Group g' to g angular differential macroscopic (n, yn) reaction subtype R_y cross section for nuclide j_y . Units: cm^{-1} .
- $\chi_{dg,m}^{j_f}$
 = Group g delayed neutron yield fraction for precursor family m produced by fissions of nuclide j_f . Dimensionless.

As stated previously, the direction dependence of the total cross section can be a problem for angular discretization, and group quantities weighted with separable fluxes are often used. With these simpler quantities, Eq. (2.32) becomes:

$$\begin{aligned}
 \frac{\partial}{\partial t} n_g(t, \vec{r}, \hat{\Omega}) &= -\nabla \cdot \hat{\Omega} \psi_g(t, \vec{r}, \hat{\Omega}) - \Sigma_{tg}(t, \vec{r}) \psi_g(t, \vec{r}, \hat{\Omega}) + \\
 &+ \sum_{j=1}^J \sum_{g'=1}^G \left[\iint_{4\pi} d\Omega' \Sigma_{sgg'}^j(t, \vec{r}, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi_{g'}(t, \vec{r}, \hat{\Omega}') \right] + \\
 &+ \frac{1}{4\pi} \sum_{j_f=1}^{J_f} \sum_{g'=1}^G \left[\chi_{dg,m}^{j_f} \nu_p \Sigma_{fgg'}^{j_f}(t, \vec{r}) \iint_{4\pi} d\Omega' \psi_{g'}(t, \vec{r}, \hat{\Omega}') \right] + \\
 &+ \sum_{j_y=1}^{J_y} \sum_{y=1}^Y \sum_{\text{all } R_y} \sum_{g'=1}^G \left[y \iint_{4\pi} d\Omega' \Sigma_{n,yn}^{j_y, R_y}(t, \vec{r}, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi_{g'}(t, \vec{r}, \hat{\Omega}') \right] + \\
 &+ \frac{1}{4\pi} \sum_{j_f=1}^{J_f} \left[\sum_{m=1}^{M^{j_f}} \chi_{dg,m}^{j_f} \lambda_m^{j_f} c_m^{j_f}(t, \vec{r}) \right] + s_{exg}(t, \vec{r}, \hat{\Omega}),
 \end{aligned} \tag{2.32}$$

in which:

$$\Sigma_{tg}(t, \vec{r}) = \text{Group } g \text{ macroscopic total cross section, weighted by separable angular flux. Units: } \text{cm}^{-1}.$$

In Eq. (2.31), the group cross sections' time, angular and position dependence is based both on those of the weighting fluxes, as well as those of the corresponding number densities. In Eq. (2.32), their time, angular and position dependencies are only due to those of the corresponding number densities and probability distributions.

With the above definitions of the group quantities, the multigroup linear delayed neutron precursor equation becomes (adapted from Ref. [4, Chapter 3]):

$$\begin{aligned}
 \frac{\partial}{\partial t} c_m^{j_f}(t, \vec{r}) &= \sum_{g'=1}^G \left[\nu_{d,m} \Sigma_{fg'}^{j_f}(t, \vec{r}) \iint_{4\pi} d\Omega' \psi_{g'}(t, \vec{r}, \hat{\Omega}') \right] - \\
 &- \lambda_m^{j_f} c_m^{j_f}(t, \vec{r}) \quad \forall m \in [1, \dots, M^{j_f}], j_f \in [1, \dots, J_f],
 \end{aligned} \tag{2.33}$$

in which:

$\nu_{d,m}\Sigma_{fg'}^{jf}(t, \vec{r}) =$ Group g' delayed neutron precursor family m production cross section for nuclide j_f , weighted by scalar flux. Units: cm^{-1} .

Group fluxes cannot be used to evaluate energy-dependent reaction rate densities (Eqs. (2.13a) and (2.13c)), but can be used to evaluate group angular and scalar reaction rate densities through Eqs. (2.27). This is sufficient information for all types of reactor analysis.

G angular group fluxes are the unknowns in the multigroup linear NTE system. To solve for them, it is necessary to build all of the group properties. The yield fractions can be built directly from nuclear data, but the group velocities, group and group-to-group cross sections require appropriate weighting fluxes. The most conventional way to do so is to assume a given in-group flux, and use it in Eqs. (2.27) and (2.28) to define the group properties. This assumed in-group flux is almost always constant in time, but may be position- and direction-dependent (therefore, nonseparable). Depending on the model geometry, effective region-specific homogenized weighting fluxes (i.e., separable fluxes) may instead be used. These assumed fluxes are necessarily only approximations (true fluxes are not known), but as G increases, the potential error due to inaccurate assumed fluxes reduces to zero.

Nuclear data processing codes like NJOY 2012 build group properties for fine group structures by assuming a fixed energy spectrum specified by a number of parameters [27, 28]. These output cross sections are microscopic; the group cross sections and velocities are therefore constants, and the group-to-group cross sections are angle-dependent in a discretized form (angular discretization is discussed in subsections 2.1.1.6 and 2.1.1.7 below). Being microscopic, they are also nuclide-specific. Neutron transport codes use these fine group nuclide-specific cross sections to build position- and direction-dependent cross sections with coarser group structures. This process is discussed in more detail in subsection 2.1.1.5; ultimately, once sufficiently accurate group cross sections are available, their source is irrelevant for the transport solver.

Group properties are nuclide-specific prior to angular discretization, at which point they can be combined.

This completes the summary of multigroup energy discretization. Two of the three important neutron transport problems exist primarily for group property generation. All three are discussed in the following subsection.

2.1.1.5 Three Important Neutron Transport Problems

Three types of neutron transport geometries are usually addressed in nuclear reactor analysis:

1. An individual fuel pin cell, either 0- (homogenized), 1- or 2-dimensional.
2. A 2-dimensional fuel pin lattice, either of an individual assembly, or a group of assemblies.
3. A 0-dimensional (constant spatial shape of the solution, only the amplitude of the solution varies), 1-, 2- or 3-dimensional full core.

These geometries are discussed separately below.

Fuel pin cell geometry is usually the first step of reactor analysis. A cross-section of a single representative fuel pin is analyzed, with the purpose of computing an approximate weighting flux to be used for building group properties for subsequent analysis. For a square lattice, the cell

is a square with a side that is the length of the fuel pitch, centered at the pin center. For a hexagonal lattice, the cell is, similarly, an equilateral triangle with a side that is the length of the fuel pitch, centered at the pin center. Both of these geometries are 2-dimensional, and so for simplification, some codes use an equivalent 1-dimensional infinite cylinder geometry; this is known as the Wigner-Seitz approximation. In the equivalent 1D geometry, the outer radius of the coolant flow area is determined such that the coolant to fuel area ratio is conserved. The 2D and 1D square pin cell geometries are shown in Figure 2.2.

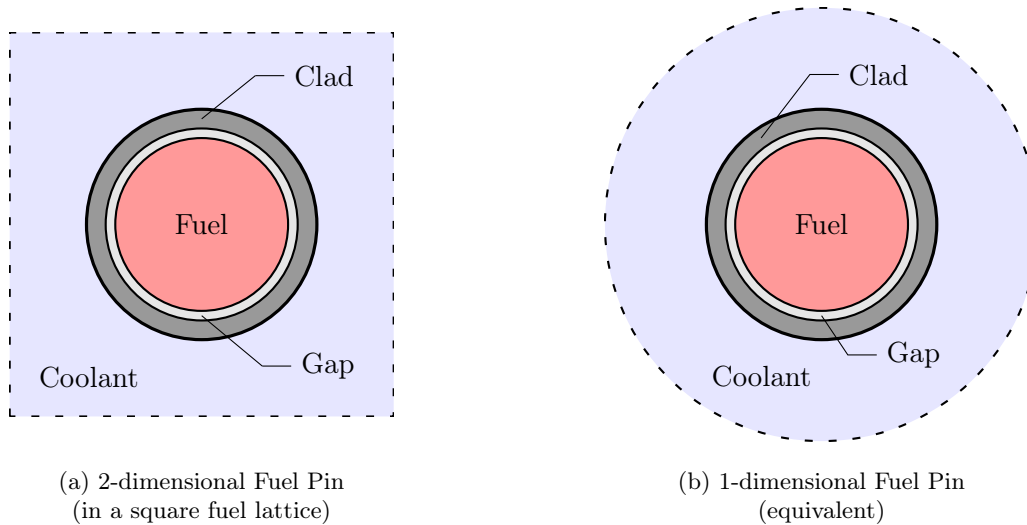


Figure 2.2: Fuel Pin Geometries

Because the fuel pin cell problem is only meant for building cross sections for a subsequent larger calculation, it does not have to be an accurate representation of pin-level reaction rates. It is inadequate for transient calculations, and so is only solved as a steady state eigenproblem. Reflective or periodic boundary conditions (BCs) are normally used; the Wigner-Seitz approximation uses white BCs, which are not recommended [29]. Some group property generation codes, like NJOY 2012 [27], do not include the spatial component at all, and instead homogenize the pin cell to solve a 0-dimensional eigenproblem.

When the cross-sectional geometry is accounted for, it is important to recognize that neutrons traveling between two points may not be traveling parallel to the cross-sectional plane, and instead may be traveling at an incline — that is, partly transversely. To account for this polar angle variation in the direction of travel, the polar angle θ is typically discretized using an axial quadrature. The choice of quadrature varies depending on the angular discretization chosen. Examples include the Tabuchi-Yamamoto quadrature for the Method of Characteristics (MOC) [30] and the Level Symmetric quadrature for the Discrete Ordinates method [31, 32].

In a reactor with multiple sufficiently different fuel types, it may be useful to solve several fuel pin eigenproblems to use the resulting weighting fluxes for different parts of the reactor. In a pressurized water reactor (PWR), the fuel pin cell problem may also be used for rough, but fast, depletion calculations. After the group properties are constructed for subsequent calculation, the fuel pin lattice may be solved.

Fuel pin lattice is the next step in reactor analysis. Like the pin cell problem, it is a 2-dimensional cross-section perpendicular to the pins' axes, but here a lattice of pins is looked at. The lattice problem may be an individual fuel assembly, or a group of them, up to a full reactor cross section. Non-fuel elements, such as the control assemblies, reflector and structural elements, which are absent from the pin cell problem, may be present in the lattice problem. A sample 2D lattice, the BASALA experiment geometry (used for studying mixed oxide recycling in boiling water reactors [33]) is shown in Figure 2.3. The smaller cells here contain the absorbers within the control blades; the larger cells contain the fuel.

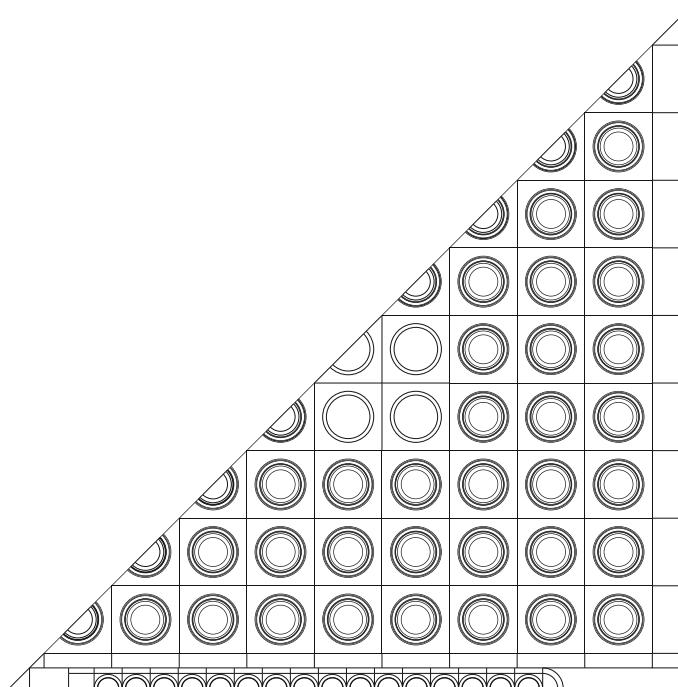


Figure 2.3: 2D BASALA Geometry
(from Ref. [33])

A lattice calculation is, like the pin cell, almost always steady state. It has several purposes: (a) to build group properties for the full core calculation, (b) to compute other potentially useful properties for the full core calculation, and (c) to compute fuel depletion rates throughout fuel lifetime. A lattice calculation is most useful for reactors with short neutron mean free paths, such as light water reactors (LWRs). In other reactor types, it is often skipped, in favor of a full core analysis with more energy groups.

Lattice codes further condense the properties in energy, but also spatially homogenize them. Spatial homogenization consists of dividing the problem into regions (usually one per assembly) and building a single set of group properties for this region, such that the reaction rates in the region are conserved, and individual pin power may be reconstructed from the homogenized solution. The homogenization procedure depends on the full core method to be used.

Like the pin cell problem, a 2D lattice requires the use of polar angle quadratures. Additionally, while axial leakage (the rate of neutrons leaving the geometry by moving transversely to the cross-sectional cell of interest) is neglected in the fuel pin cell problem, in a lattice calculation, axial

leakage must be accounted for. This is typically done through a special transverse leakage term, which modifies the streaming term $(\nabla \cdot \hat{\Omega}\psi(\vec{r}, E, \hat{\Omega}))$ in the steady state NTE.

The boundary conditions used in a lattice calculation depend on the problem geometry. A single assembly may use reflective, periodic or albedo (a combination of vacuum and reflective) BCs; a large cross-section of the reactor will typically use reflective BCs across (radial) planes of symmetry, and vacuum BCs at the outer edge of the reactor.

Multiple sections of the core are usually analyzed using a lattice code. In the active fuel region, there is typically little variation in fresh fuel number densities; however, as the fuel depletes, the higher power regions deplete faster, and so axial variation in fuel number densities results.

CASMO-4E [34] and DRAGON Version4 [35] are two examples of lattice codes, used primarily for LWR cores and for general applications, respectively.

After a set of lattice calculations builds the full core group properties, the full reactor core may be analyzed.

Full core geometry is analyzed using significantly different methods from the pin cell and the lattice geometries. The full core problem may be steady state or transient, and is usually solved using homogenized, few group properties obtained from the lattice calculations. Typically, instead of solving the multigroup NTE (Eq. (2.32)), the multigroup neutron diffusion equation (NDE) is solved. The delayed neutron precursor equations are unaffected. The neutron diffusion equation is discussed in subsection 2.1.2.

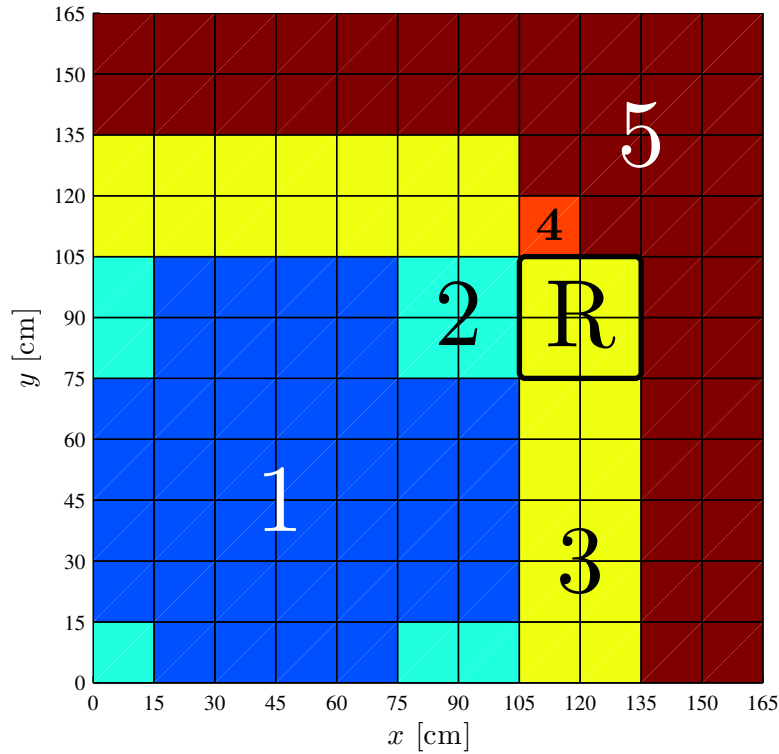


Figure 2.4: 2D BWR Full Core Geometry Specification
(from Ref. [36, Problem 14])

The full core problem may be 1-, 2- or 3-dimensional; 1D and 2D approaches are primarily used to capture core transient behavior, while the 3D geometry does this in addition to providing the full power profile. The difference between a 2D lattice problem and a full 2D core problem is in the physics used: a lattice problem would use a transport model (fine group, no diffusion approximation, steady state), while a full core problem would usually use a few group, potentially transient neutron diffusion model.

A sample 2D BWR full core geometry specification (as a quarter core with reflective BCs) is shown in Figure 2.4. Here the numbers 1–5 refer to node type (they vary by enrichment and control blade height; 5 is light water), R indicates the region from which a control blade drops, initiating the transient of interest. The geometry is spatially homogenized: each $15\text{ cm} \times 15\text{ cm}$ assembly is treated as a node with uniform properties. This problem is analyzed in detail in chapters 6 and 7.

Like 2D pin cell and lattice models, a 2D full core model must also account for transverse leakage; here an axial buckling term, which acts similarly to absorption, is often used. All full core models almost always use vacuum boundary conditions on the core boundary, and reflective boundary conditions at planes of symmetry.

As stated above, a 3D full core model yields the full power profile of the reactor. In many applications, the variation of the shape of this profile in a transient is unimportant, and so may be assumed constant — only the power profile amplitude varies. Such model is known as the “neutron point kinetics” model — a 0-dimensional full core neutron model. The point kinetics are discussed in subsection 2.1.3 below.

While coupling to thermal hydraulics (not discussed so far) is important for all 3 of the problem types, full core models, being the only ones analyzed in transients, are most affected by it. As stated earlier, full core neutron diffusion analyses are the focus of this work.

Examples of full core diffusion codes include SIMULATE-5, used primarily for LWRs [37], and DIF3D, used for general applications [38].

This concludes the summary of the three important neutron transport problems. Multigroup energy discretization has already been discussed; neutron diffusion is a specific form of angular discretization, and is discussed in subsection 2.1.2 below. The pin cell and lattice problems require other angular discretization methods, two of which are discussed in the following subsections.

2.1.1.6 Spherical Harmonics Angular Discretization Methods

Below, the multigroup transient NTE and DNPE are discretized using spherical harmonics. The spherical harmonics angular discretization method can only be used if the group quantities were weighted by separable fluxes, because it does not work if the group velocity or total macroscopic group cross section are direction-dependent. Therefore, Eqs. (2.32) and (2.33) are discretized here.

Spherical harmonics (also known as P_n) discretization works by expanding the angular quantities using either the spherical harmonics functions, or their real components. Several different normalizations exist for spherical harmonics. Below, the discretization based on real spherical harmonics components (RSHCs) is used, similar to the one given by Hébert [39]. Stacey gives the more conventional complex spherical harmonics-based discretization [2, Chapter 9].

RSHCs are based on Legendre polynomials. The first few Legendre polynomials are defined by Eqs. (2.34):

$$P_0(\mu) = 1, \quad (2.34a)$$

$$P_1(\mu) = \mu, \quad (2.34b)$$

$$P_2(\mu) = \frac{1}{2}(3\mu^2 - 1), \quad (2.34c)$$

and the subsequent Legendre polynomials are defined by the following recursion relation:

$$P_{l+1}(\mu) = \frac{2l+1}{l+1}\mu P_l(\mu) - \frac{l}{l+1}P_{l-1}(\mu) \quad \text{with } l \geq 1, \quad (2.35)$$

in which:

$P_l(\mu)$ = Legendre polynomial of order l .

The associated Legendre polynomials are defined by Eq. (2.36):

$$P_l^n(\mu) = (1 - \mu^2)^{n/2} \frac{d^n}{d\mu^n} P_l(\mu) \quad \text{with } l \geq n \geq 0, \quad (2.36)$$

in which:

$P_l^n(\mu)$ = Associated Legendre polynomial of order l and degree n .

Special trigonometric functions are used in this form of RSHCs, defined by:

$$\mathcal{T}_n(\varphi) = \begin{cases} \cos(n\varphi) & \text{if } n \geq 0, \\ \sin(|n|\varphi) & \text{if } n < 0, \end{cases} \quad (2.37)$$

in which:

$\mathcal{T}_n(\varphi)$ = Special trigonometric function of degree n .

Lastly, the Kronecker delta is used:

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases} \quad (2.38)$$

where

δ_{ij} = Kronecker delta symbol.

The argument of the RSHC function is the direction unit vector $\hat{\Omega}$ (defined in Eq. (2.2)). Here, it is expressed in terms of 2 degrees of freedom: the colatitude angle cosine μ and the azimuthal angle φ :

$$\hat{\Omega} = \mu \hat{x} + \sqrt{1 - \mu^2} \cos(\varphi) \hat{y} + \sqrt{1 - \mu^2} \sin(\varphi) \hat{z}. \quad (2.39)$$

Using the above quantities, the RSHC function is given by:

$$R_l^n(\hat{\Omega}) = \sqrt{(2 - \delta_{n0}) \frac{(l - |n|)!}{(l + |n|)!}} P_l^{|n|}(\mu) \mathcal{T}_n(\varphi) \quad \text{with } l \geq 0, -l \leq n \leq l, \quad (2.40)$$

in which:

$$R_l^n(\hat{\Omega}) = \text{Real spherical harmonics component of order } l \text{ and degree } n.$$

The regular and associated Legendre polynomials, special associated trigonometric functions, and RSHCs obey the following orthogonality relations:

$$\int_0^{2\pi} d\varphi \mathcal{T}_n(\varphi) \mathcal{T}_{n'}(\varphi) = \begin{cases} 0 & \text{if } n \neq n', \\ \pi & \text{if } n = n' \neq 0, \\ 2\pi & \text{if } n = n' = 0. \end{cases} \quad (2.41)$$

$$\int_{-1}^1 d\mu P_l(\mu) P_{l'}(\mu) = \frac{2}{2l+1} \delta_{ll'}, \quad (2.42)$$

$$\int_{-1}^1 d\mu P_l^n(\mu) P_l^{n'}(\mu) = \frac{2(l+n)!}{(2l+1)(l-n)!} \delta_{nn'}, \quad (2.43)$$

$$\iint_{4\pi} d\Omega R_l^n(\hat{\Omega}) R_{l'}^{n'}(\hat{\Omega}) = \frac{4\pi}{2l+1} \delta_{ll'} \delta_{nn'}. \quad (2.44)$$

The addition theorem for real spherical harmonics is given by:

$$P_l(\mu) (\hat{\Omega} \cdot \hat{\Omega}') = \sum_{n=-l}^l R_l^n(\hat{\Omega}) R_l^n(\hat{\Omega}') \quad (2.45)$$

Cosine-dependent and angular quantities can be expanded in terms of Legendre polynomials and RSHCs, using Eqs. (2.46) and (2.47), respectively:

$$f(\mu) \cong \sum_{l=0}^L \frac{2l+1}{2} f^l P_l(\mu), \quad (2.46)$$

$$f(\hat{\Omega}) \cong \sum_{l=0}^L \frac{2l+1}{4\pi} \sum_{n=-l}^l f^{ln} R_l^n(\hat{\Omega}). \quad (2.47)$$

Here the individual Legendre and real spherical harmonics moments are given by Eqs. (2.48) and (2.49), respectively:

$$f^l = \int_{-1}^1 d\mu f(\mu) P_l(\mu), \quad (2.48)$$

$$f^{ln} = \iint_{4\pi} d\Omega f(\hat{\Omega}) R_l^n(\hat{\Omega}). \quad (2.49)$$

The following nomenclature was used:

$f(\mu)$ = A cosine-dependent quantity.

L = Order of expansion. As L is increased, the expansion becomes a more accurate approximation.

f^l = l^{th} Legendre moment of $f(\mu)$.

$f(\hat{\Omega})$ = A direction-dependent quantity.

f^{ln} = Order l , degree n real spherical harmonics moment of $f(\hat{\Omega})$.

The angular group neutron density, flux and external source can be expanded using Eq. (2.47). The angular differential scattering and (n, yn) group cross sections are formally densities in target direction, but, by Eqs. (2.9), (2.10) and (2.28a), are actually only functions of the scattering angle cosine. The two cross sections can therefore be expanded using Eq. (2.46).

The spherical harmonics discretization of the multigroup linear NTE is arrived at by performing the following steps on each term in Eq. (2.32):

1. Expand the angular group neutron density, flux and external source using Eq. (2.47).
2. Expand the differential scattering and (n, yn) cross sections using Eq. (2.46).
3. Multiply the equation by $R_l^n(\hat{\Omega})$.
4. Integrate the resulting equation over all directions.
5. Simplify the resulting expressions using the recursion and orthogonality relations and the addition theorem for real spherical harmonics (Eqs. (2.35), (2.41) and (2.45)).

Performing these steps yields the spherical harmonics discretization of the multigroup linear NTE (adapted from Ref. [39], with the streaming term's coefficients from Eqs.(2.55)):

$$\begin{aligned}
 \frac{\partial}{\partial t} n_g^{ln}(t, \vec{r}) = & -\mathcal{I}_l^n \frac{\partial}{\partial x} \psi_g^{l+1,n}(t, \vec{r}) - \mathcal{J}_l^n \frac{\partial}{\partial x} \psi_g^{l-1,n}(t, \vec{r}) - \mathcal{A}_l^n \frac{\partial}{\partial y} \psi_g^{l-1,n-1}(t, \vec{r}) - \\
 & -\mathcal{B}_l^n \frac{\partial}{\partial y} \psi_g^{l+1,n-1}(t, \vec{r}) - \mathcal{C}_l^n \frac{\partial}{\partial y} \psi_g^{l-1,n+1}(t, \vec{r}) - \mathcal{D}_l^n \frac{\partial}{\partial y} \psi_g^{l+1,n+1}(t, \vec{r}) - \\
 & -\mathcal{E}_l^n \frac{\partial}{\partial z} \psi_g^{l-1,-n+1}(t, \vec{r}) - \mathcal{F}_l^n \frac{\partial}{\partial z} \psi_g^{l+1,-n+1}(t, \vec{r}) - \mathcal{G}_l^n \frac{\partial}{\partial z} \psi_g^{l-1,-n-1}(t, \vec{r}) - \\
 & -\mathcal{H}_l^n \frac{\partial}{\partial z} \psi_g^{l+1,-n-1}(t, \vec{r}) - \Sigma_{tg}(t, \vec{r}) \psi_g^{ln}(t, \vec{r}) + \\
 & + \sum_{j=1}^J \sum_{g'=1}^G \left[\Sigma_{sgg'}^{j,l}(t, \vec{r}) \psi_{g'}^{ln}(t, \vec{r}) \right] + \delta_{l0} \sum_{j_f=1}^{J_f} \sum_{g'=1}^G \left[\chi \nu_p \Sigma_{f g g'}^{j_f}(t, \vec{r}) \psi_{g'}^{00}(t, \vec{r}) \right] + \\
 & + \sum_{j_y=1}^{J_y} \sum_{y=1}^Y \sum_{\text{all } R_y} \sum_{g'=1}^G \left[y \Sigma_{n, y n g g'}^{j_y, R_y, l}(t, \vec{r}) \psi_{g'}^{ln}(t, \vec{r}) \right] + \\
 & + \delta_{l0} \sum_{j_f=1}^{J_f} \left[\sum_{m=1}^{M^{j_f}} \chi_{dg,m}^{j_f} \lambda_m^{j_f} c_m^{j_f}(t, \vec{r}) \right] + s_{exg}^{ln}(t, \vec{r}).
 \end{aligned} \tag{2.50}$$

Real spherical harmonics moments are only nonzero with $-l \leq n \leq l$, otherwise the corresponding terms in Eq. (2.50) cancel out. The following notation was used here:

- $n_g^{ln}(t, \vec{r})$ = Group g neutron density spherical harmonics moment of order l and degree n . It is derived from $n_g(t, \vec{r}, \hat{\Omega})$ using Eq. (2.49), and can be related to the group g flux spherical harmonics moment $\psi_g^{ln}(t, \vec{r})$ using Eq. (2.51). Units: neutrons/cm³.
- $\psi_g^{ln}(t, \vec{r})$ = Group g flux spherical harmonics moment of order l and degree n . It is derived from $\psi_g(t, \vec{r}, \hat{\Omega})$ using Eq. (2.49), and is the unknown in Eq. (2.50). Units: neutrons/cm² s.
- $\mathcal{A}_l^n \dots \mathcal{J}_l^n$ = Streaming term's coefficients, given by Eqs. (2.55). Dimensionless.
- $\Sigma_{sgg'}^{j,l}(t, \vec{r})$ = l^{th} Legendre moment of group g' to g macroscopic scattering cross section for nuclide j . It is given by Eq. (2.52). Units: cm⁻¹.
- $\Sigma_{n,yngg'}^{j_y,R_y,l}(t, \vec{r})$ = l^{th} Legendre moment of group g' to g macroscopic (n, yn) reaction subtype R_y cross section for nuclide j_y . It is given by Eq. (2.53). Units: cm⁻¹.
- $s_{exg}^{ln}(t, \vec{r})$ = Group g external source spherical harmonics moment of order l and degree n . It is given by Eq. (2.54). Unit: neutrons/cm³ s.

The following equations define the derived quantities in Eq. (2.50):

$$\psi_g^{ln}(t, \vec{r}) = V_{ng} n_g^{ln}(t, \vec{r}), \quad (2.51)$$

$$\Sigma_{sgg'}^{j,l}(t, \vec{r}) = 2\pi \int_{-1}^1 d\mu_s \Sigma_{sgg'}^j(t, \vec{r}, \hat{\Omega}' \rightarrow \hat{\Omega}) P_l(\mu_s), \quad (2.52)$$

$$\Sigma_{n,yngg'}^{j_y,R_y,l}(t, \vec{r}) = 2\pi \int_{-1}^1 d\mu_s \Sigma_{n,yngg'}^{j_y,R_y}(t, \vec{r}, \hat{\Omega}' \rightarrow \hat{\Omega}) P_l(\mu_s), \quad (2.53)$$

$$s_{exg}^{ln}(t, \vec{r}) = \iint_{4\pi} d\Omega s_{exg}(t, \vec{r}, \hat{\Omega}) R_l^n(\hat{\Omega}). \quad (2.54)$$

Note, that to evaluate Eqs. (2.52) and (2.53), the definition of the scattering angle cosine (Eq. (2.11)) must be used.

In a 3-dimensional geometry, the streaming term $\nabla \cdot \hat{\Omega} \psi_g(t, \vec{r}, \hat{\Omega})$ is discretized as a sum of 10 spatial derivatives of coupled moments, multiplied by constant coefficients. Reference [39] contains the spherical harmonics discretization of a 2-dimensional geometry (yz -plane), but its $\mathcal{E}_l^n - \mathcal{H}_l^n$ coefficients have an incorrect sign in front of them. It also does not contain the \mathcal{I}_l^n and \mathcal{J}_l^n coefficients, as they are only present in the 3-dimensional geometry. The complete spherical harmonics discretization of the 3-dimensional streaming term uses the coefficients given in Eqs. (2.55).

$$\mathcal{A}_l^n = \frac{\text{sgn}(n)}{2(2l+1)} \sqrt{(1-\delta_{n0})(1+\delta_{n1})(l+n)(l+n-1)}, \quad (2.55a)$$

$$\mathcal{B}_l^n = -\frac{\text{sgn}(n)}{2(2l+1)} \sqrt{(1-\delta_{n0})(1+\delta_{n1})(l-n+1)(l-n+2)}, \quad (2.55b)$$

$$\mathcal{C}_l^n = -\frac{\text{sgn}(n)}{2(2l+1)}\sqrt{(1-\delta_{n,-1})(1+\delta_{n0})(l-n)(l-n-1)}, \quad (2.55c)$$

$$\mathcal{D}_l^n = \frac{\text{sgn}(n)}{2(2l+1)}\sqrt{(1-\delta_{n,-1})(1+\delta_{n0})(l+n+1)(l+n+2)}, \quad (2.55d)$$

$$\mathcal{E}_l^n = -\frac{\text{sgn}(n)}{2(2l+1)}\sqrt{(1-\delta_{n0})(1-\delta_{n1})(l+n)(l+n-1)}, \quad (2.55e)$$

$$\mathcal{F}_l^n = \frac{\text{sgn}(n)}{2(2l+1)}\sqrt{(1-\delta_{n0})(1-\delta_{n1})(l-n+1)(l-n+2)}, \quad (2.55f)$$

$$\mathcal{G}_l^n = -\frac{\text{sgn}(n)}{2(2l+1)}\sqrt{(1+\delta_{n0})(1+\delta_{n,-1})(l-n)(l-n-1)}, \quad (2.55g)$$

$$\mathcal{H}_l^n = \frac{\text{sgn}(n)}{2(2l+1)}\sqrt{(1+\delta_{n0})(1+\delta_{n,-1})(l+n+1)(l+n+2)}, \quad (2.55h)$$

$$\mathcal{I}_l^n = \frac{1}{2l+1}\sqrt{(l+n+1)(l-n+1)}, \quad (2.55i)$$

$$\mathcal{J}_l^n = \frac{1}{2l+1}\sqrt{(l+n)(l-n)}. \quad (2.55j)$$

A modified signum function $\text{sgn}(n)$ is used in Eqs. (2.55), defined by:

$$\text{sgn}(n) = \begin{cases} 1 & \text{if } n \geq 0, \\ -1 & \text{if } n < 0, \end{cases} \quad (2.56)$$

in which:

$\text{sgn}(n) = 1$ if $n \geq 0$. The more conventional signum function is 0 at $n = 0$, while this version is 1 at $n = 0$.

Prior to angular discretization, it was not possible to combine the individual constituents of the group-to-group cross sections into a single quantity. However, after their group-to-group Legendre moments are built, the individual group-to-group Legendre moments are only time- and position-dependent. By Eqs. (2.7) and (2.8), and the region-specific separable flux assumption in subsection 2.1.1.4, we can see that this dependence is only due to the time and position dependence of the number density, which, spatially, is piecewise-constant. Except during fluid mixing or solute (e.g., boric acid in PWR coolant) concentration changes, the number densities of individual nuclide types in a region will vary proportionally to each other in time; such variation may be due to fluid density change or control assembly movement. This energy and angle discretization, together with the transient shape similarity of the number densities, give rise to a “combined scattering” group-to-group cross section, defined by Eq. (2.57):

$$\Sigma_{s+gg'}^l(t, \vec{r}) = \sum_{j=1}^J \Sigma_{sgg'}^{j,l}(t, \vec{r}) + \sum_{j_y=1}^{J_y} \sum_{y=1}^Y \sum_{\text{all } R_y} y \Sigma_{n,ymgg'}^{j_y,R_y,l}(t, \vec{r}), \quad (2.57)$$

in which:

$\Sigma_{s+gg'}^l(t, \vec{r}) = l^{\text{th}}$ Legendre moment of group g' to g combined scattering macroscopic cross section. This cross section accounts for the l^{th} Legendre moment of all non-fission reactions with secondary neutrons, including scattering and all (n, yn) . The multiplication with $y > 1$ is also accounted for. Units: cm^{-1} .

In the benchmark problems with the group properties and group-to-group Legendre moments already constructed, a separate (n, yn) cross section is rarely given; instead, the “scattering” cross section given is normally the combined scattering macroscopic cross section. In most materials, the two quantities are almost the same, as the scattering cross section is usually significantly higher than (n, yn) . Many lattice codes also expect the scattering matrices (the $G \times G$ set of all group-to-group scattering cross sections for one or more l) to account for (n, yn) , and so this is the form in which group property generation codes output the scattering matrices.

The group-to-group cross sections’ Legendre moments for the solutes of interest (e.g., boron concentration) must therefore be accounted for separately in transients, as their number densities may change differently from the other isotopes. At steady state, solutes of interest may be incorporated into Eq. (2.57).

Note, that while the fission production group-to-group cross sections could be combined similarly to the non-fission secondary neutron cross sections, individual isotopes’ fission rates must be evaluated separately to properly compute the individual delayed neutron precursor generation rates, and so they are kept separate below. In steady state problems, or in problems with the same precursor families for all fissionable isotopes, the fission production group-to-group cross sections are combined.

With the combined scattering cross section, Eq. (2.50) simplifies:

$$\begin{aligned}
 \frac{\partial}{\partial t} n_g^{ln}(t, \vec{r}) = & -\mathcal{I}_l^n \frac{\partial}{\partial x} \psi_g^{l+1,n}(t, \vec{r}) - \mathcal{J}_l^n \frac{\partial}{\partial x} \psi_g^{l-1,n}(t, \vec{r}) - \mathcal{A}_l^n \frac{\partial}{\partial y} \psi_g^{l-1,n-1}(t, \vec{r}) - \\
 & -\mathcal{B}_l^n \frac{\partial}{\partial y} \psi_g^{l+1,n-1}(t, \vec{r}) - \mathcal{C}_l^n \frac{\partial}{\partial y} \psi_g^{l-1,n+1}(t, \vec{r}) - \mathcal{D}_l^n \frac{\partial}{\partial y} \psi_g^{l+1,n+1}(t, \vec{r}) - \\
 & -\mathcal{E}_l^n \frac{\partial}{\partial z} \psi_g^{l-1,-n+1}(t, \vec{r}) - \mathcal{F}_l^n \frac{\partial}{\partial z} \psi_g^{l+1,-n+1}(t, \vec{r}) - \mathcal{G}_l^n \frac{\partial}{\partial z} \psi_g^{l-1,-n-1}(t, \vec{r}) - \\
 & -\mathcal{H}_l^n \frac{\partial}{\partial z} \psi_g^{l+1,-n-1}(t, \vec{r}) - \Sigma_{tg}(t, \vec{r}) \psi_g^{ln}(t, \vec{r}) + \\
 & + \sum_{g'=1}^G \left[\Sigma_{s+gg'}^l(t, \vec{r}) \psi_{g'}^{ln}(t, \vec{r}) \right] + \delta_{l0} \sum_{j_f=1}^{J_f} \sum_{g'=1}^G \left[\chi \nu_p \Sigma_{f_{gg'}}^{j_f}(t, \vec{r}) \psi_{g'}^{00}(t, \vec{r}) \right] + \\
 & + \delta_{l0} \sum_{j_f=1}^{J_f} \left[\sum_{m=1}^{M^{j_f}} \chi_{dg,m}^{j_f} \lambda_m^{j_f} c_m^{j_f}(t, \vec{r}) \right] + s_{exg}^{ln}(t, \vec{r}).
 \end{aligned} \tag{2.58}$$

With the above expansions, the scalar flux in the multigroup DNPE (Eq. (2.33)) is replaced with $\psi_g^{00}(t, \vec{r})$, and the spherical harmonics form of the linear multigroup DNPE becomes:

$$\begin{aligned}
 \frac{\partial}{\partial t} c_m^{j_f}(t, \vec{r}) = & \sum_{g'=1}^G \left[\nu_{d,m} \Sigma_{f_{g'}}^{j_f}(t, \vec{r}) \psi_{g'}^{00}(t, \vec{r}) \right] - \\
 & - \lambda_m^{j_f} c_m^{j_f}(t, \vec{r}) \quad \forall m \in [1, \dots, M^{j_f}], j_f \in [1, \dots, J_f],
 \end{aligned} \tag{2.59}$$

After angular discretization, the equations must be discretized in geometric space, after which the resulting system of ordinary differential equations (ODEs) may be integrated in time. Conventionally, the spherical harmonics method is usually discretized using a form of finite difference method; a form of superlinear finite element (FE)-based spatial discretization has recently been proposed [39]. Here, a form of finite volume (FV) discretization with upwind flux (UF) is used. Upwind flux is chosen to approximate the neutron advection (streaming, first order spatial derivative) operator because of the typically poor (oscillatory) performance of the more conventional cell boundary-centered fluxes for advection operators [40, Chapter 4].

Generally, the finite volume with upwind flux scheme has linear order of convergence in space, but is simple, stable and robust [40, Chapter 4]. This is desirable for transient neutron transport problems, in which anisotropies typically propagate one way (e.g., away from a control assembly or a boundary).

Finite volume discretization is arrived at by considering a cell (i, j, k) with volume $\Delta V_{i,j,k}$:

$$\Delta V_{i,j,k} = \Delta x \cdot \Delta y \cdot \Delta z, \quad (2.60)$$

in which:

- i, j, k = Indices numbering the cells in the positive x -, y - and z -directions, respectively.
- $\Delta V_{i,j,k}$ = Volume of a single parallelepiped cell (i, j, k) , over which the solution is approximated as flat. Units: cm^3 .
- $\Delta x, \Delta y, \Delta z$ = x -, y - and z -dimensions of the parallelepiped cell (i, j, k) . Note, that here the cells are assumed to be the same size and orientation. Units: cm .

The cell's faces' areas are given by:

$$A_x = \Delta y \cdot \Delta z, \quad (2.61a)$$

$$A_y = \Delta x \cdot \Delta z, \quad (2.61b)$$

$$A_z = \Delta x \cdot \Delta y, \quad (2.61c)$$

in which:

- A_x, A_y, A_z = Surface areas of the parallelepiped cell (i, j, k) normal to the x -, y - and z -axes, respectively. Units: cm^2 .

Finite volume method uses cell-averaged and cell-total quantities, given by Eqs. (2.62) and (2.63), respectively:

$$\bar{f}_{i,j,k} = \frac{1}{\Delta V_{i,j,k}} \iiint_{\Delta V_{i,j,k}} dV f(\vec{r}), \quad (2.62)$$

$$F_{i,j,k} = \iiint_{\Delta V_{i,j,k}} dV f(\vec{r}), \quad (2.63)$$

in which:

- $f(\vec{r})$ = A position-dependent quantity.

$\bar{f}_{i,j,k}$ = Spatial average of $f(\vec{r})$ over cell (i, j, k) .

$F_{i,j,k}$ = Total of the quantity of which $f(\vec{r})$ is the spatial density, in cell (i, j, k) .

Integrating Eq. (2.58) over cell (i, j, k) and using upwind flux to approximate the interface fluxes yields the FV with UF spatial discretization of the multigroup spherical harmonics linear NTE with a combined scattering term. The source neighboring cell for the upwind flux depends on the sign of the “velocity,” which here is the sign of the streaming coefficient. By inspection of Eqs. (2.55), their signs depend on the moment degree n . This yields Eqs. (2.64):

$$\begin{aligned}
 \frac{d}{dt} N_{g,i,j,k}^{ln}(t) &= \mathcal{I}_l^n A_x \left(\bar{\psi}_{g,i-1,j,k}^{l+1,n}(t) - \bar{\psi}_{g,i,j,k}^{l+1,n}(t) \right) + \mathcal{J}_l^n A_x \left(\bar{\psi}_{g,i-1,j,k}^{l-1,n}(t) - \bar{\psi}_{g,i,j,k}^{l-1,n}(t) \right) + \\
 &+ \mathcal{A}_l^n A_y \left(\bar{\psi}_{g,i,j-1,k}^{l-1,n-1}(t) - \bar{\psi}_{g,i,j,k}^{l-1,n-1}(t) \right) + \mathcal{B}_l^n A_y \left(\bar{\psi}_{g,i,j,k}^{l+1,n-1}(t) - \bar{\psi}_{g,i,j+1,k}^{l+1,n-1}(t) \right) + \\
 &+ \mathcal{C}_l^n A_y \left(\bar{\psi}_{g,i,j,k}^{l-1,n+1}(t) - \bar{\psi}_{g,i,j+1,k}^{l-1,n+1}(t) \right) + \mathcal{D}_l^n A_y \left(\bar{\psi}_{g,i,j-1,k}^{l+1,n+1}(t) - \bar{\psi}_{g,i,j,k}^{l+1,n+1}(t) \right) + \\
 &+ \mathcal{E}_l^n A_z \left(\bar{\psi}_{g,i,j,k}^{l-1,-n+1}(t) - \bar{\psi}_{g,i,j,k+1}^{l-1,-n+1}(t) \right) + \mathcal{F}_l^n A_z \left(\bar{\psi}_{g,i,j,k-1}^{l+1,-n+1}(t) - \bar{\psi}_{g,i,j,k}^{l+1,-n+1}(t) \right) + \\
 &+ \mathcal{G}_l^n A_z \left(\bar{\psi}_{g,i,j,k}^{l-1,-n-1}(t) - \bar{\psi}_{g,i,j,k+1}^{l-1,-n-1}(t) \right) + \mathcal{H}_l^n A_z \left(\bar{\psi}_{g,i,j,k-1}^{l+1,-n-1}(t) - \bar{\psi}_{g,i,j,k}^{l+1,-n-1}(t) \right) - \\
 &- \Delta V_{i,j,k} \bar{\Sigma}_{tg,i,j,k} \bar{\psi}_{g,i,j,k}^{ln}(t) + \Delta V_{i,j,k} \sum_{g'=1}^G \left[\bar{\Sigma}_{s+gg',i,j,k}^l \bar{\psi}_{g',i,j,k}^{ln}(t) \right] + \\
 &+ \delta_{l0} \Delta V_{i,j,k} \sum_{j_f=1}^{J_f} \sum_{g'=1}^G \left[\overline{\chi \nu_p \Sigma_{fgg',i,j,k}^{j_f}} \bar{\psi}_{g',i,j,k}^{00}(t) \right] + \\
 &+ \delta_{l0} \sum_{j_f=1}^{J_f} \left[\sum_{m=1}^{M^{j_f}} \chi_{dg,m}^{j_f} \lambda_m^{j_f} C_{m,i,j,k}^{j_f}(t) \right] + S_{exg,i,j,k}^{ln}(t) \quad \text{if } n \geq 0,
 \end{aligned} \tag{2.64a}$$

$$\begin{aligned}
 \frac{d}{dt} N_{g,i,j,k}^{ln}(t) &= \mathcal{I}_l^n A_x \left(\bar{\psi}_{g,i-1,j,k}^{l+1,n}(t) - \bar{\psi}_{g,i,j,k}^{l+1,n}(t) \right) + \mathcal{J}_l^n A_x \left(\bar{\psi}_{g,i-1,j,k}^{l-1,n}(t) - \bar{\psi}_{g,i,j,k}^{l-1,n}(t) \right) + \\
 &+ \mathcal{A}_l^n A_y \left(\bar{\psi}_{g,i,j,k}^{l-1,n-1}(t) - \bar{\psi}_{g,i,j+1,k}^{l-1,n-1}(t) \right) + \mathcal{B}_l^n A_y \left(\bar{\psi}_{g,i,j-1,k}^{l+1,n-1}(t) - \bar{\psi}_{g,i,j,k}^{l+1,n-1}(t) \right) + \\
 &+ \mathcal{C}_l^n A_y \left(\bar{\psi}_{g,i,j-1,k}^{l-1,n+1}(t) - \bar{\psi}_{g,i,j,k}^{l-1,n+1}(t) \right) + \mathcal{D}_l^n A_y \left(\bar{\psi}_{g,i,j,k}^{l+1,n+1}(t) - \bar{\psi}_{g,i,j+1,k}^{l+1,n+1}(t) \right) + \\
 &+ \mathcal{E}_l^n A_z \left(\bar{\psi}_{g,i,j,k-1}^{l-1,-n+1}(t) - \bar{\psi}_{g,i,j,k}^{l-1,-n+1}(t) \right) + \mathcal{F}_l^n A_z \left(\bar{\psi}_{g,i,j,k}^{l+1,-n+1}(t) - \bar{\psi}_{g,i,j,k+1}^{l+1,-n+1}(t) \right) + \\
 &+ \mathcal{G}_l^n A_z \left(\bar{\psi}_{g,i,j,k-1}^{l-1,-n-1}(t) - \bar{\psi}_{g,i,j,k}^{l-1,-n-1}(t) \right) + \mathcal{H}_l^n A_z \left(\bar{\psi}_{g,i,j,k}^{l+1,-n-1}(t) - \bar{\psi}_{g,i,j,k+1}^{l+1,-n-1}(t) \right) - \\
 &- \Delta V_{i,j,k} \bar{\Sigma}_{tg,i,j,k} \bar{\psi}_{g,i,j,k}^{ln}(t) + \Delta V_{i,j,k} \sum_{g'=1}^G \left[\bar{\Sigma}_{s+gg',i,j,k}^l \bar{\psi}_{g',i,j,k}^{ln}(t) \right] + \\
 &+ \delta_{l0} \Delta V_{i,j,k} \sum_{j_f=1}^{J_f} \sum_{g'=1}^G \left[\overline{\chi \nu_p \Sigma_{fgg',i,j,k}^{j_f}} \bar{\psi}_{g',i,j,k}^{00}(t) \right] + \\
 &+ \delta_{l0} \sum_{j_f=1}^{J_f} \left[\sum_{m=1}^{M^{j_f}} \chi_{dg,m}^{j_f} \lambda_m^{j_f} C_{m,i,j,k}^{j_f}(t) \right] + S_{exg,i,j,k}^{ln}(t) \quad \text{if } n < 0.
 \end{aligned} \tag{2.64b}$$

The following nomenclature was used here:

- $N_{g,i,j,k}^{ln}(t)$ = Total effective number of neutrons contributing to spherical harmonics moment (l, n) in group g , cell (i, j, k) . This quantity is related to $n_g^{ln}(t, \vec{r})$ through Eq. (2.63). It can yield the group g cell (i, j, k) average flux spherical harmonics moment $\bar{\psi}_{g,i,j,k}^{ln}(t)$ using Eq. (2.65). Units: neutrons.
- $\bar{\psi}_{g,i,j,k}^{ln}(t)$ = Group g flux spherical harmonics moment of order l and degree n , averaged over cell (i, j, k) . It is derived from $\psi_g^{ln}(t, \vec{r})$ using Eq. (2.62), and is the unknown in Eqs. (2.64). Units: neutrons/cm²s.
- $\bar{\Sigma}_{tg,i,j,k}(t)$ = Group g macroscopic total cross section, averaged over cell (i, j, k) . It is derived from $\Sigma_{tg}(t, \vec{r})$ using Eq. (2.62). Units: cm⁻¹.
- $\bar{\Sigma}_{s+gg',i,j,k}^l(t)$ = l^{th} Legendre moment of group g' to g macroscopic combined scattering cross section, averaged over cell (i, j, k) . It is derived from $\Sigma_{s+gg'}^l(t, \vec{r})$ using Eq. (2.62). Units: cm⁻¹.
- $\overline{\chi\nu_p\Sigma_{fgg',i,j,k}^{jf}}(t)$ = Group g' to g macroscopic prompt fission production cross section for nuclide j_f , averaged over cell (i, j, k) . It is derived from $\chi\nu_p\Sigma_{fgg'}^{jf}(t, \vec{r})$ using Eq. (2.62). Units: cm⁻¹.
- $C_{m,i,j,k}^{jf}(t)$ = Total number of delayed neutron precursors of family m produced by fissions of nuclide j_f , related to $c_m^{jf}(t, \vec{r})$ through Eq. (2.63). Units: precursors.
- $S_{exg,i,j,k}^{ln}(t)$ = Total group g cell (i, j, k) spherical harmonics moment (l, n) of the external source. It is obtained from $s_{exg}^{ln}(t, \vec{r})$ through Eq. (2.63). Units: neutrons/s.

$N_{g,i,j,k}^{ln}(t)$ and $\bar{\psi}_{g,i,j,k}^{ln}(t)$ are related via:

$$\bar{\psi}_{g,i,j,k}^{ln}(t) = \frac{V_{ng}}{\Delta V_{i,j,k}} N_{g,i,j,k}^{ln}(t). \quad (2.65)$$

Equation (2.59) is discretized through finite volumes by simply integrating it over the volume $\Delta V_{i,j,k}$ and averaging the properties over the cells, to yield:

$$\begin{aligned} \frac{d}{dt} C_{m,i,j,k}^{jf}(t) = & \sum_{g'=1}^G \left[\overline{\nu_{d,m}\Sigma_{fg',i,j,k}^{jf}}(t) \bar{\psi}_{g',i,j,k}^{00}(t) \right] - \\ & - \lambda_m^{jf} C_{m,i,j,k}^{jf}(t) \quad \forall m \in [1, \dots, M^{jf}], j_f \in [1, \dots, J_f], \end{aligned} \quad (2.66)$$

with the following nomenclature:

- $\overline{\nu_{d,m}\Sigma_{fg',i,j,k}^{jf}}(t)$ = Group g' delayed neutron precursor family m production cross section for nuclide j_f , averaged over cell (i, j, k) . It is derived from $\nu_{d,m}\Sigma_{fg'}^{jf}(t, \vec{r})$ using Eq. (2.62). Units: cm⁻¹.

This concludes the summary of the spherical harmonics discretization of the transient linear

multigroup NTE and DNPE. The other popular angular discretization method is the discrete ordinates method, discussed in the following subsection.

2.1.1.7 Discrete Ordinates Angular Discretization Methods

While the discrete ordinates (also known as S_N) method, unlike spherical harmonics, does not strictly require the group quantities to be weighted by region-specific separable fluxes, and can work if the group velocity and total macroscopic group cross section are direction-dependent, it is almost always only used with the more conventional, separable flux-based group properties. Therefore, like with spherical harmonics, Eqs. (2.32) and (2.33) are discretized here.

The discrete ordinates angular discretization method works by discretizing the direction domain into a set of specific directions (quadrature directions), and making use of angular quadratures to approximate the integral quantities, including the flux. To accurately represent isotropic fluxes (and more generally, angular fluxes symmetric about a plane), the most common class of angular quadratures is the “level-symmetric quadrature,” adapted for neutron transport by Carlson [31]. In this class of angular quadratures, the direction domain is divided into 8 octants, and the opposite-facing octants’ directions are matching and opposite-facing, with identical quadrature weights.

The N -level-symmetric quadrature has N distinct direction cosines μ , so $N/2$ per the direction cosine’s Cartesian axis’ ray (i.e., there are $N/2$ distinct values of μ with $\mu > 0$, and $N/2$ with $\mu < 0$). There are therefore a total of $\left(\frac{N}{2} + 1\right) \frac{N}{4}$ distinct directions per octant, and $N_d = N(N + 2)$ overall. Level-symmetric quadratures are typically specified as direction cosine and quadrature direction weight pairs for a single octant; the weights are normalized as:

$$\sum_{d=1}^{\left(\frac{N}{2}+1\right)\frac{N}{4}} w_n = 1. \quad (2.67)$$

The quadrature approximation to the integral over all directions is therefore given by:

$$\iint_{4\pi} d\Omega f(\hat{\Omega}) \cong \frac{\pi}{2} \sum_{d=1}^{N_d} w_d f^d, \quad (2.68)$$

with:

$$N_d = N(N + 2), \quad (2.69)$$

$$\hat{\Omega}_d = \mu_d \hat{x} + \eta_d \hat{y} + \xi_d \hat{z}, \quad (2.70)$$

$$f^d = f(\hat{\Omega}_d). \quad (2.71)$$

Here, the following nomenclature was used:

- d = Quadrature direction index.
- N = Number of unique values that a direction cosine takes in a given quadrature set.
- N_d = Total number of quadrature directions.
- $\hat{\Omega}_d$ = Quadrature direction n unit vector. Dimensionless.

- μ_d, η_d, ξ_d = Cartesian direction cosines of quadrature direction n . Dimensionless.
 w_d = Quadrature direction n weight. Dimensionless.
 f^d = A direction-dependent quantity's value at quadrature direction n .

There exist numerous methods to choose the level-symmetric quadrature direction cosines and weights; Ref. [4, Chapter 4] provides a summary, and Ref. [32] gives a more detailed procedure for generating many of the more conventional quadrature sets. The choice of quadrature sets does not affect the form of the discretized equations.

The most primitive form of discrete ordinates discretization uses Eq. (2.68) to directly discretize all direction-dependent quantities in Eq. (2.32), including the scattering and (n, yn) terms [6, Chapter 4]. This approach to discretizing these two terms is inaccurate, and does not directly work with the combined scattering matrices' Legendre moments that group property generation codes output (see subsection 2.1.1.6). For this reason, these two terms are instead discretized by using angular quadrature to approximate the integrals in their spherical harmonics expansions; this way, group-to-group combined scattering matrices' Legendre moments can be used instead. This approach yields the discrete ordinates discretization of the multigroup neutron transport equation (adapted from Ref. [4, Chapter 3]):

$$\begin{aligned}
 \frac{\partial}{\partial t} n_g^d(t, \vec{r}) &= -\nabla \cdot \hat{\Omega}_d \psi_g^d(t, \vec{r}) - \Sigma_{tg}(t, \vec{r}) \psi_g^d(t, \vec{r}) + \\
 &+ \sum_{g'=1}^G \sum_{l=0}^L \sum_{n=-l}^l \left[\frac{2l+1}{4\pi} \Sigma_{s+gg'}^l(t, \vec{r}) R_l^n(\hat{\Omega}_d) \tilde{\psi}_{g'}^{ln}(t, \vec{r}) \right] + \\
 &+ \frac{1}{4\pi} \sum_{j_f=1}^{J_f} \sum_{g'=1}^G \left[\chi \nu_p \Sigma_{fj_f gg'}^{j_f}(t, \vec{r}) \left(\frac{\pi}{2} \sum_{d'=1}^{N_d} w_{d'} \psi_{g'}^{d'}(t, \vec{r}) \right) \right] + \\
 &+ \frac{1}{4\pi} \sum_{j_f=1}^{J_f} \sum_{m=1}^{M^{j_f}} \left[\chi_{dg,m}^{j_f} \lambda_m^{j_f} c_m^{j_f}(t, \vec{r}) \right] + s_{exg}^d(t, \vec{r}).
 \end{aligned} \tag{2.72}$$

Here, the following nomenclature was used:

- $n_g^d(t, \vec{r})$ = Group g neutron density in quadrature direction d , evaluated from $n_g(t, \vec{r}, \hat{\Omega})$ using Eq. (2.71). It can be related to group g direction d flux $\psi_g^d(t, \vec{r})$ using Eq. (2.73). Units: neutrons/cm³ sr.
 $\psi_g^d(t, \vec{r})$ = Group g flux in quadrature direction d , evaluated from $\psi_g(t, \vec{r}, \hat{\Omega})$ using Eq. (2.71). It is the unknown in Eq. (2.72). Units: neutrons/cm² s sr.
 $\tilde{\psi}_g^{ln}(t, \vec{r})$ = Group g flux spherical harmonics moment of order l and degree n , approximated from flux values at quadrature directions using Eq. (2.74). Units: neutrons/cm² s.
 $s_{exg}^d(t, \vec{r})$ = Group g external source in quadrature direction d , evaluated from $s_{exg}(t, \vec{r}, \hat{\Omega})$ using Eq. (2.71). Units: neutrons/cm³ s sr.

Equations (2.73) and (2.74) relate $n_g^d(t, \vec{r})$ and $\psi_g^d(t, \vec{r})$ and define the approximate flux spherical harmonics moment, respectively:

$$\psi_g^d(t, \vec{r}) = V_{ng} n_g^d(t, \vec{r}), \quad (2.73)$$

$$\tilde{\psi}_g^{ln}(t, \vec{r}) = \frac{\pi}{2} \sum_{d'=1}^{N_d} w_{d'} \psi_g^{d'}(t, \vec{r}) R_l^n(\hat{\Omega}_{d'}). \quad (2.74)$$

To discretize the linear multigroup DNPE (Eq. (2.33)), the scalar flux is replaced with its quadrature approximation:

$$\begin{aligned} \frac{\partial}{\partial t} c_m^{jf}(t, \vec{r}) &= \sum_{g'=1}^G \left[\nu_{d,m} \Sigma_{fg'}^{jf}(t, \vec{r}) \left(\frac{\pi}{2} \sum_{d'=1}^{N_d} w_{d'} \psi_{g'}^{d'}(t, \vec{r}) \right) \right] - \\ &- \lambda_m^{jf} c_m^{jf}(t, \vec{r}) \quad \forall m \in [1, \dots, M^{jf}], j_f \in [1, \dots, J_f]. \end{aligned} \quad (2.75)$$

This completes the angular discretization of the multigroup linear NTE and DNPE via discrete ordinates. As with spherical harmonics, after angular discretization, the equations must be discretized in geometric space. Because the discrete ordinate method is almost always used for steady state problems, the conventional spatial discretization technique, called ‘‘diamond differencing,’’ is strictly a combination of spatial discretization and a ‘‘sweeping’’ procedure, in which the cell-averaged and cell surface fluxes are computed sequentially, sweeping the domain starting at a boundary. A recently proposed form of diamond differencing relies on approximating the intracell flux shapes through Legendre polynomials to achieve spatial superconvergence; this procedure is still steady state-only [41].

Fundamentally, diamond differencing can be thought of as a form of finite volume discretization with upwind fluxes: the surface fluxes in a specific direction d are determined by considering the average flux in the direction from which $\hat{\Omega}_d$ points. Transient discrete ordinates have been used for angular discretization of non-neutron radiation transport; FV with UF discretization was used [42]. For these reasons, FV with UF discretization is used here as well.

As with spherical harmonics method, the FV discretization of Eq. (2.72) is arrived at by integrating it over $\Delta V_{i,j,k}$, and using UFs to approximate the streaming term:

$$\begin{aligned} \frac{d}{dt} n_{g,i,j,k}^d(t) &= \mu_d A_x \left[\delta_+(\mu_d) \bar{\psi}_{g,i-1,j,k}^d(t) - \text{sgn}(\mu_d) \bar{\psi}_{g,i,j,k}^d(t) - \delta_-(\mu_d) \bar{\psi}_{g,i+1,j,k}^d(t) \right] + \\ &+ \eta_d A_y \left[\delta_+(\eta_d) \bar{\psi}_{g,i,j-1,k}^d(t) - \text{sgn}(\eta_d) \bar{\psi}_{g,i,j,k}^d(t) - \delta_-(\eta_d) \bar{\psi}_{g,i,j+1,k}^d(t) \right] + \\ &+ \xi_d A_z \left[\delta_+(\xi_d) \bar{\psi}_{g,i,j,k-1}^d(t) - \text{sgn}(\xi_d) \bar{\psi}_{g,i,j,k}^d(t) - \delta_-(\xi_d) \bar{\psi}_{g,i,j,k+1}^d(t) \right] - \\ &- \Delta V_{i,j,k} \bar{\Sigma}_{tg,i,j,k}^d(t) \bar{\psi}_{g,i,j,k}^d(t) + \\ &+ \Delta V_{i,j,k} \sum_{g'=1}^G \sum_{l=0}^L \sum_{n=-l}^l \left[\frac{2l+1}{4\pi} \bar{\Sigma}_{s+gg',i,j,k}^l(t) R_l^n(\hat{\Omega}_d) \tilde{\psi}_{g',i,j,k}^{ln}(t) \right] + \\ &+ \frac{\Delta V_{i,j,k}}{4\pi} \sum_{j_f=1}^{J_f} \sum_{g'=1}^G \left[\overline{\chi \nu_p \Sigma_{fgg',i,j,k}^{j_f}}(t) \left(\frac{\pi}{2} \sum_{d'=1}^{N_d} w_{d'} \bar{\psi}_{g',i,j,k}^{d'}(t) \right) \right] + \\ &+ \frac{1}{4\pi} \sum_{j_f=1}^{J_f} \sum_{m=1}^{M^{j_f}} \left[\chi_{dg,m}^{j_f} \lambda_m^{j_f} C_{m,i,j,k}^{j_f}(t) \right] + s_{exg,i,j,k}^d(t). \end{aligned} \quad (2.76)$$

The following nomenclature was used here:

$n_{g,i,j,k}^d(t)$ = Total angular density of neutrons streaming in quadrature direction d in group g , cell (i, j, k) . This quantity is related to $n_g^d(t, \vec{r})$ through Eq. (2.63). It can be related to the group g cell (i, j, k) direction d average flux $\bar{\psi}_{g,i,j,k}^d(t)$ using Eq. (2.77). Units: neutrons/sr.

$\bar{\psi}_{g,i,j,k}^d(t)$ = Group g cell (i, j, k) average flux in direction d . It is derived from $\psi_g^d(t, \vec{r})$ using Eq. (2.62), and is the unknown in Eq. (2.76). Units: neutrons/cm²s.

$\delta_{\pm}(\mu)$ = Sign delta functions, defined by Eqs. (2.79). Dimensionless.

$\bar{\psi}_{g,i,j,k}^{ln}(t)$ = Group g cell (i, j, k) average flux spherical harmonics moment of order l and degree n , approximated from flux values at quadrature directions using Eq. (2.78). Units: neutrons/cm²s.

$s_{exg,i,j,k}^d(t)$ = Total group g cell (i, j, k) external angular source density. It is obtained from $s_{exg}^d(t, \vec{r})$ using Eq. (2.63). Units: neutrons/s sr.

Equations (2.77) and (2.78) relate $\bar{\psi}_{g,i,j,k}^d(t)$ and $N_{g,i,j,k}^{ln}(t)$ and define the approximate cell average flux spherical harmonics moment, respectively:

$$\bar{\psi}_{g,i,j,k}^d(t) = \frac{V_{ng}}{\Delta V_{i,j,k}} N_{g,i,j,k}^{ln}(t), \quad (2.77)$$

$$\bar{\psi}_{g,i,j,k}^{ln}(t) = \frac{\pi}{2} \sum_{d'=1}^{N_d} w_{d'} \bar{\psi}_{g,i,j,k}^{d'}(t) R_l^n(\hat{\Omega}_{d'}). \quad (2.78)$$

The sign delta functions used to discretize the streaming term are given by:

$$\delta_+(\mu) = \begin{cases} 1 & \text{if } \mu > 0, \\ 0 & \text{otherwise,} \end{cases} \quad (2.79a)$$

$$\delta_-(\mu) = \begin{cases} 1 & \text{if } \mu < 0, \\ 0 & \text{otherwise.} \end{cases} \quad (2.79b)$$

Lastly, to discretize Eq. (2.75) through finite volumes, we integrate it over cell (i, j, k) and average the properties over the cell:

$$\begin{aligned} \frac{d}{dt} C_{m,i,j,k}^{j_f}(t) &= \sum_{g'=1}^G \left[\overline{\nu_{d,m} \Sigma_{f g'}^{j_f}}(t) \left(\frac{\pi}{2} \sum_{d'=1}^{N_d} w_{d'} \bar{\psi}_{g',i,j,k}^{d'}(t) \right) \right] - \\ &- \lambda_m^{j_f} C_{m,i,j,k}^{j_f}(t) \quad \forall m \in [1, \dots, M^{j_f}], j_f \in [1, \dots, J_f]. \end{aligned} \quad (2.80)$$

This concludes the summary of multigroup, angular and spatial discretization methods for the linear NTE and DNPE. Because, as discussed in subsection 2.1.1.5, it would be prohibitively expensive to solve these discretized equations directly in the full core geometry, the multigroup diffusion equations are typically used instead. They are discussed in subsection 2.1.2.

2.1.2 Neutron Diffusion

Multigroup neutron diffusion is a form of approximation to the multigroup neutron transport equation (Eq. (2.31)), based on approximating the net neutron current density using Fick's law:

$$\vec{\mathbf{J}}_g(t, \vec{\mathbf{r}}) = -\mathbb{D}_g(t, \vec{\mathbf{r}}) \nabla \phi_g(t, \vec{\mathbf{r}}), \quad (2.81)$$

where

$\vec{\mathbf{J}}_g(t, \vec{\mathbf{r}})$ = Group g net neutron current density vector, formally defined by Eq. (2.82). This quantity characterizes the net neutron streaming rate density magnitude and direction. Units: neutrons/cm²s.

$\mathbb{D}_g(t, \vec{\mathbf{r}})$ = Group g directional diffusion coefficient 3×3 diagonal tensor, given by Eq. (2.83). This quantity is used to account for the net streaming and anisotropic scattering rate densities. Units: cm.

The individual terms in Eq. (2.81) are defined by:

$$\vec{\mathbf{J}}_g(t, \vec{\mathbf{r}}) = J_{xg}(t, \vec{\mathbf{r}}) \hat{\mathbf{x}} + J_{yg}(t, \vec{\mathbf{r}}) \hat{\mathbf{y}} + J_{zg}(t, \vec{\mathbf{r}}) \hat{\mathbf{z}} = \iint_{4\pi} d\Omega \hat{\Omega} \psi_g(t, \vec{\mathbf{r}}, \hat{\Omega}), \quad (2.82)$$

$$\mathbb{D}_g(t, \vec{\mathbf{r}}) = \begin{bmatrix} D_{xg}(t, \vec{\mathbf{r}}) & 0 & 0 \\ 0 & D_{yg}(t, \vec{\mathbf{r}}) & 0 \\ 0 & 0 & D_{zg}(t, \vec{\mathbf{r}}) \end{bmatrix}, \quad (2.83)$$

in which:

$J_{eg}(t, \vec{\mathbf{r}})$ = Group g net current density component along the Cartesian axis e . Units: neutrons/cm²s.

$D_{eg}(t, \vec{\mathbf{r}})$ = Group g Cartesian axis e directional neutron diffusion coefficient. Methods to obtain these coefficients are discussed below. Units: cm.

Equation (2.81) is the more general, directional form of Fick's law. Additional assumptions, discussed below, result in:

$$D_g(t, \vec{\mathbf{r}}) = D_{xg}(t, \vec{\mathbf{r}}) = D_{yg}(t, \vec{\mathbf{r}}) = D_{zg}(t, \vec{\mathbf{r}}), \quad (2.84)$$

which yields the nondirectional Fick's law:

$$\vec{\mathbf{J}}_g(t, \vec{\mathbf{r}}) = -D_g(t, \vec{\mathbf{r}}) \nabla \phi_g(t, \vec{\mathbf{r}}), \quad (2.85)$$

where:

$D_g(t, \vec{\mathbf{r}})$ = Group g nondirectional neutron diffusion coefficient. Methods to obtain it are discussed below. Units: cm.

The fundamental assumptions underlying the neutron diffusion approximation are (Ref. [6, Chapter 4]):

1. Fractional net current density rate of change can be treated as quasistatic. This assumption can be written as:

$$\frac{1}{J_g(t, \vec{r})} \frac{\partial}{\partial t} J_g(t, \vec{r}) \ll V_{ng} \Sigma_{tg}(t, \vec{r}), \quad (2.86)$$

in which:

$$J_g(t, \vec{r}) = \text{Magnitude of } \vec{J}_g(t, \vec{r}). \text{ Units: neutrons/cm}^2 \text{ s.}$$

For a (very low) 1 cm^{-1} total cross section and a thermal neutron group with group velocity of 2200 m/s, the collision frequency per neutron $V_{ng} \Sigma_{tg}(t, \vec{r})$ is $2.2 \times 10^5 \text{ s}^{-1}$. Therefore, for this assumption to be invalid, the net current density has to vary by about a factor of 100 000 in one second, or faster. This is almost never the case in a reactor, even during a transient.

It is important to note that this assumption does not imply that the reactor is at steady state; instead, it only implies that the variation in net current density does not look like a shock front traveling through a region in a reactor.

2. The external source is assumed to be isotropic. Physically, this is almost always the case, because external neutron sources work through particle-stimulated neutron emission, which, assuming the incident particles are produced in the same material as the decaying nuclei, is isotropic.
3. Net neutron current density is assumed to be well modeled by Fick's law (Eqs. (2.81) and (2.85)). Depending on the origins of the diffusion coefficients, Fick's law holds on a full core scale, with homogenized regions, and tends to fail without spatial homogenization.

The group diffusion coefficients may be derived from the multigroup spherical harmonics equation (Eq. (2.58)). Assumption 3 sets L to 1. The scalar flux and net neutron current density current may be written as the zeroth and a combination of the first Legendre moments of the flux, respectively:

$$\phi_g(t, \vec{r}) = \psi_g^{0,0}(t, \vec{r}), \quad (2.87)$$

$$\vec{J}_g(t, \vec{r}) = \psi_g^{1,0}(t, \vec{r}) \hat{x} + \psi_g^{1,1}(t, \vec{r}) \hat{y} + \psi_g^{1,-1}(t, \vec{r}) \hat{z}. \quad (2.88)$$

Applying the 3 assumptions above to the spherical harmonics equations of order 1 and degrees 0, 1 and -1 yields the x -, y - and z -directional diffusion coefficients, respectively:

$$D_{eg}(t, \vec{r}) = \frac{1}{3 \left[\Sigma_{tg}(t, \vec{r}) - \frac{1}{J_{eg}(t, \vec{r})} \sum_{g'=1}^G \Sigma_{s+gg'}^1(t, \vec{r}) J_{eg'}(t, \vec{r}) \right]}. \quad (2.89)$$

As discussed in subsection 2.1.1.4, group properties' definitions are formal, and require a weighting flux. Here, $J_{eg}(t, \vec{r})$ is such weighting function, with the units of a flux. Several different approximations of varying fidelity can be used to construct usable diffusion coefficients based on Eq. (2.89) (adapted from Refs. [2, Chapter 10] and [4, Chapter 4]):

1. A group net current density $\vec{J}_g(t, \vec{r})$ is assumed, based on a lattice calculation. This is one of the most accurate possible approximations, but, because net currents generally converge much slower than scalar fluxes in full core solutions, and because they are frequently near zero (and therefore cannot serve as weighting fluxes), this approximation is rarely available. Because the energy spectrum in $\vec{J}_g(t, \vec{r})$ is generally direction-dependent, using this approximation produces directional diffusion coefficients with Eq. (2.89) directly.
2. The multigroup energy spectrum composed of the group scalar fluxes $\phi_g(t, \vec{r})$ is assumed to be sufficiently close to the multigroup energy spectrum of composed of the net current densities. As with the net current above, averaged group scalar fluxes may be obtained from a lattice calculation. This is also one of the more accurate approximations.

Scalar fluxes are nondirectional, and so, using a single weighting scalar flux $\phi_g(t, \vec{r})$ in place of $J_g(t, \vec{r})$ in Eq. (2.89) produces nondirectional diffusion coefficients:

$$D_g(t, \vec{r}) \cong \frac{1}{3 \left[\Sigma_{tg}(t, \vec{r}) - \frac{1}{\phi_g(t, \vec{r})} \sum_{g'=1}^G \Sigma_{s+gg'}^1(t, \vec{r}) \phi_{g'}(t, \vec{r}) \right]}. \quad (2.90)$$

3. It is assumed that there is no energy change due to anisotropic scattering. Effectively, this means that all anisotropic scattering is in-group (i.e., $g' = g$), or:

$$\Sigma_{s+gg'}^1(t, \vec{r}) \cong \begin{cases} \sum_{g''=1}^G \Sigma_{s+g''g}^1(t, \vec{r}) & \text{if } g = g', \\ 0 & \text{if } g \neq g'. \end{cases} \quad (2.91)$$

Note, that Eq. (2.91) conserves the overall anisotropic scattering rate density in group g . Substituting it into Eq. (2.89) cancels the effect of the weighting net current, and yields the nondirectional diffusion coefficient:

$$D_g(t, \vec{r}) \cong \frac{1}{3 \left[\Sigma_{tg}(t, \vec{r}) - \sum_{g'=1}^G \Sigma_{s+g'g}^1(t, \vec{r}) \right]}. \quad (2.92)$$

This approximation is very convenient, because it does not require a weighting spectrum, but it is of questionable accuracy. It works best for heavy isotopes, in elastic collisions with which neutrons cannot lose a significant amount of energy. Anisotropic inelastic scattering and $(n, \gamma n)$ reactions are neglected by this assumption; such reactions are nearly isotropic, and so, for heavy isotopes, this approximation works. It is important to note (for comparison with the next approximation) that approximation 3 works well in heavily absorbing materials, like the fuel.

Approximation 3 does not work at all in the presence of light water (or other moderators), where ^1H is present, and significant loss of energy occurs in anisotropic scattering.

4. It is assumed that the anisotropic inscattering and outscattering rates are balanced. Effectively, this means that the rate of anisotropic inscattering into group g is approximately equal to the rate of anisotropic outscattering from group g :

$$\sum_{g'=1}^G \Sigma_{s+gg'}^1(t, \vec{r}) J_{eg'}(t, \vec{r}) \cong \sum_{g'=1}^G \Sigma_{s+g'g}^1(t, \vec{r}) J_{eg}(t, \vec{r}). \quad (2.93)$$

For this approximation to be correct, absorption and spatial variation must be negligible; this is generally true in a low-absorption (i.e., hydrogenous) and homogeneous environment away from the boundaries. Substituting Eq. (2.93) into Eq. (2.89) again cancels the effect of the weighting net current, and yields the nondirectional diffusion coefficient given by Eq. (2.92).

Approximations 3 and 4 both resulted in the same expression for the $D_g(t, \vec{r})$ (Eq. (2.92)), despite starting with very different physical assumptions. This makes this technique for diffusion coefficient generation a good initial estimate for the diffusion coefficient; approximations 1 and 2 are still generally preferable.

5. Scattering is assumed to be fully isotropic in LCS:

$$\Sigma_{s+gg'}^1(t, \vec{r}) \cong 0, \quad (2.94)$$

which yields:

$$D_g(t, \vec{r}) \cong \frac{1}{3\Sigma_{tg}(t, \vec{r})}. \quad (2.95)$$

This is the simplest approximation here, but it is grossly inaccurate, and so should be avoided whenever possible.

There exist alternative ways of arriving at $\mathbb{D}_g(t, \vec{r})$, such as the linear buckling (B_1) method, which relies on spatially uniform regions, and, for such regions, is a refinement of the P_1 method. Methods exist to adjust the diffusion term to account for transverse leakage and spatial discontinuities [4, Chapter 5].

As with other multigroup properties, their origin is ultimately irrelevant for a multigroup diffusion equation solver. To obtain the linear multigroup neutron diffusion equation, we substitute Eq. (2.81) into Eq. (2.58) with $l = n = 0$ to yield:

$$\begin{aligned} \frac{\partial}{\partial t} n_g(t, \vec{r}) &= \nabla \cdot \mathbb{D}_g(t, \vec{r}) \nabla \phi_g(t, \vec{r}) - \Sigma_{tg}(t, \vec{r}) \phi_g(t, \vec{r}) + \\ &+ \sum_{g'=1}^G \left[\Sigma_{sgg'}(t, \vec{r}) \phi_{g'}(t, \vec{r}) \right] + \sum_{j_f=1}^{J_f} \sum_{g'=1}^G \left[\chi \nu_p \Sigma_{fj_gg'}^{j_f}(t, \vec{r}) \phi_{g'}(t, \vec{r}) \right] + \\ &+ \sum_{j_f=1}^{J_f} \left[\sum_{m=1}^{M^{j_f}} \chi_{dg,m}^{j_f} \lambda_m^{j_f} c_m^{j_f}(t, \vec{r}) \right] + S_{exg}(t, \vec{r}). \end{aligned} \quad (2.96)$$

Here the following nomenclature was used:

$n_g(t, \vec{\mathbf{r}})$ = Group g neutron density, defined by Eq. (2.97). It can be related to the scalar flux using Eq. (2.98). Units: neutrons/cm³.

$\Sigma_{sgg'}(t, \vec{\mathbf{r}})$ = Isotropic combined group g' to g macroscopic scattering cross section, defined by Eq. (2.99). Units: cm⁻¹.

$S_{exg}(t, \vec{\mathbf{r}})$ = Group g external source, defined by Eq. (2.100). Units: neutrons/cm³ s.

The following equations define the previously undefined quantities in Eq. (2.96):

$$n_g(t, \vec{\mathbf{r}}) = \iint_{4\pi} d\Omega n_g(t, \vec{\mathbf{r}}, \hat{\Omega}), \quad (2.97)$$

$$\phi_g(t, \vec{\mathbf{r}}) = V_{ng} n_g(t, \vec{\mathbf{r}}), \quad (2.98)$$

$$\Sigma_{sgg'}(t, \vec{\mathbf{r}}) = \Sigma_{s+gg'}^0(t, \vec{\mathbf{r}}), \quad (2.99)$$

$$S_{exg}(t, \vec{\mathbf{r}}) = \iint_{4\pi} d\Omega S_{exg}(t, \vec{\mathbf{r}}, \hat{\Omega}). \quad (2.100)$$

The multigroup DNPEs associated with the diffusion equation are identical to Eq. (2.59), differing only in notation:

$$\begin{aligned} \frac{\partial}{\partial t} c_m^{j_f}(t, \vec{\mathbf{r}}) &= \sum_{g'=1}^G \left[\nu_{d,m} \Sigma_{fg'}^{j_f}(t, \vec{\mathbf{r}}) \phi_{g'}(t, \vec{\mathbf{r}}) \right] - \\ &- \lambda_m^{j_f} c_m^{j_f}(t, \vec{\mathbf{r}}) \quad \forall m \in [1, \dots, M^{j_f}], j_f \in [1, \dots, J_f]. \end{aligned} \quad (2.101)$$

Equations (2.96) and (2.101) are clearly much simpler to discretize than the more complicated multigroup neutron transport equations. They are parabolic linear partial differential equations (PDEs). To solve them, they only need to be discretized in space and integrated in time.

Two major spatial discretization techniques exist for the multigroup NDE. They are described in the following subsections.

2.1.2.1 Pure Finite Difference Spatial Discretization

The strict finite difference approximation consists of simply evaluating all values at specified grid points $\{\vec{\mathbf{r}}_{i,j,k}\}$ (located at the corners of $\Delta x \times \Delta y \times \Delta z$ parallelepiped cells):

$$\vec{\mathbf{r}}_{i,j,k} = x_i \hat{\mathbf{x}} + y_j \hat{\mathbf{y}} + z_k \hat{\mathbf{z}}. \quad (2.102)$$

The evaluated values are denoted in one of two ways:

$$f_{i,j,k} = f(\vec{\mathbf{r}}_{i,j,k}), \quad (2.103a)$$

$$\left(f \Big|_{\vec{\mathbf{r}}=\vec{\mathbf{r}}_{i,j,k}} \right) = f(\vec{\mathbf{r}}_{i,j,k}). \quad (2.103b)$$

The streaming term is approximated using the centered finite difference approximations for the spatial derivatives (this choice is discussed in subsection 2.1.2.2).

The resulting discretized NDE becomes:

$$\begin{aligned}
 \frac{d}{dt}n_{g,i,j,k}(t) &= \mathcal{D}_{xg,i,j,k}^-(t)\phi_{g,i-1,j,k}(t) - \frac{2}{\Delta x^2}D_{xg,i,j,k}(t)\phi_{g,i,j,k}(t) + \mathcal{D}_{xg,i,j,k}^+(t)\phi_{g,i+1,j,k}(t) + \\
 &+ \mathcal{D}_{yg,i,j,k}^-(t)\phi_{g,i,j-1,k}(t) - \frac{2}{\Delta y^2}D_{yg,i,j,k}(t)\phi_{g,i,j,k}(t) + \mathcal{D}_{yg,i,j,k}^+(t)\phi_{g,i,j+1,k}(t) + \\
 &+ \mathcal{D}_{zg,i,j,k}^-(t)\phi_{g,i,j,k-1}(t) - \frac{2}{\Delta z^2}D_{zg,i,j,k}(t)\phi_{g,i,j,k}(t) + \mathcal{D}_{zg,i,j,k}^+(t)\phi_{g,i,j,k+1}(t) - \\
 &- \Sigma_{tg,i,j,k}(t)\phi_{g,i,j,k}(t) + \sum_{g'=1}^G \left[\Sigma_{sgg',i,j,k}(t)\phi_{g',i,j,k}(t) \right] + \\
 &+ \sum_{j_f=1}^{J_f} \sum_{g'=1}^G \left[\chi\nu_p \Sigma_{fgg',i,j,k}^{j_f}(t)\phi_{g,i,j,k}(t) \right] + \sum_{j_f=1}^{J_f} \left[\sum_{m=1}^{M^{j_f}} \chi_{dg,m}^{j_f} \lambda_m^{j_f} c_{m,i,j,k}^{j_f}(t) \right] + S_{exg,i,j,k}(t).
 \end{aligned} \tag{2.104}$$

The discretized diffusion term coefficients are defined by:

$$\mathcal{D}_{eg,i,j,k}^\pm(t) = \frac{1}{\Delta e^2}D_{eg,i,j,k}(t) \pm \frac{1}{2\Delta e} \left(\frac{\partial}{\partial e} D_{eg}(t, \vec{\mathbf{r}}) \Big|_{\vec{\mathbf{r}}=\vec{\mathbf{r}}_{i,j,k}} \right). \tag{2.105}$$

Here, the following nomenclature was used:

- $\vec{\mathbf{r}}_{i,j,k}$ = The position vector of grid point (i, j, k) , given by Eq. (2.102). Units: cm.
- Δe = Distance between neighboring mesh points along Cartesian axis e . Units: cm.
- $f(\vec{\mathbf{r}})$ = A position-dependent (and possibly, other variable-dependent) quantity.
- $f_{i,j,k}, n_{g,i,j,k}(t), \phi_{g,i,j,k}(t), D_{eg,i,j,k}(t), \Sigma_{tg,i,j,k}(t), \Sigma_{sgg',i,j,k}(t), \chi\nu_p \Sigma_{fgg',i,j,k}^{j_f}(t), c_{m,i,j,k}^{j_f}(t), S_{exg,i,j,k}(t)$ = Position-dependent quantities evaluated at grid point (i, j, k) .
- $\left(f \Big|_{\vec{\mathbf{r}}=\vec{\mathbf{r}}_{i,j,k}}, \left(\frac{\partial}{\partial e} D_{eg}(t, \vec{\mathbf{r}}) \Big|_{\vec{\mathbf{r}}=\vec{\mathbf{r}}_{i,j,k}} \right) \right)$ = Position-dependent quantities evaluated at grid point (i, j, k) .
- $\mathcal{D}_{eg,i,j,k}^\pm(t)$ = Discretized diffusion term coefficients, used to account for spatial variation in both $\phi_g(t, \vec{\mathbf{r}})$ and $D_{eg}(t, \vec{\mathbf{r}})$. They are given by Eq. (2.105). If $D_{eg}(t, \vec{\mathbf{r}})$ is spatially uniform, $\mathcal{D}_{eg,i,j,k}^\pm(t)$ reduces to $D_{eg,i,j,k}(t)/\Delta e^2$. Units: cm.

$n_{g,i,j,k}(t)$ and $\phi_{g,i,j,k}(t)$ are related through:

$$\phi_{g,i,j,k}(t) = V_{ng}n_{g,i,j,k}(t). \tag{2.106}$$

The multigroup DNPEs are discretized for NDE with finite difference by simply evaluating all quantities at grid points $\{\vec{\mathbf{r}}_{i,j,k}\}$:

$$\begin{aligned} \frac{d}{dt} c_{m,i,j,k}^{j_f}(t) &= \sum_{g'=1}^G \left[\nu_{d,m} \Sigma_{fg',i,j,k}^{j_f}(t) \phi_{g'}(t, \vec{\mathbf{r}}) \right] - \\ &- \lambda_m^{j_f} c_{m,i,j,k}^{j_f}(t, \vec{\mathbf{r}}) \quad \forall m \in [1, \dots, M^{j_f}], j_f \in [1, \dots, J_f], \end{aligned} \quad (2.107)$$

with:

$\nu_{d,m} \Sigma_{fg',i,j,k}^{j_f}(t)$ = Group g' delayed neutron precursor family m production cross section for nuclide j_f , evaluated at grid point (i, j, k) . Units: cm^{-1} .

The issue with this scheme is that, like any true finite difference scheme, it cannot enforce conservation of integral quantities — in this case, it cannot conserve neutrons. For this reason, true finite difference schemes like this are almost never used by diffusion codes. Instead, finite volume discretizations, which only use the finite difference approximations to approximate the intercell net currents, are used instead. One such discretization is described in the following subsection.

2.1.2.2 Finite Volume Spatial Discretization

The finite volume discretizations spherical harmonics and discrete ordinates methods are obtained by using the upwind flux method for evaluating intercell currents. As discussed in subsection 2.1.1.6, the upwind flux function has a 1st order of convergence in space, but is not prone to oscillation for advective operators, unlike the 2nd-order centered finite difference intercell flux function.

The diffusion term in Eq. (2.96) is symmetric, and so the centered finite difference discretization may be used here to approximate it. Such approximation will not be prone oscillation, and is of 2nd order of convergence in space [40, Chapter 4]. For the same reason, centered finite difference was used to discretize the diffusion term in Eq. (2.104).

As with spherical harmonics and discrete ordinates method, the FV discretization of Eq. (2.96) is arrived at by integrating it over $\Delta V_{i,j,k}$; however, here, centered finite difference is used to approximate the diffusion term. All cell properties are approximated as uniform within a cell; this is known as “*property homogenization.*” Cells with uniform properties are known as “*nodes,*” and this discretization technique is therefore called “*nodal diffusion.*” For completeness, variable parallelepiped cell dimensions will be used. Such mesh is known as the “*regular mesh.*” Cell volume is now defined by:

$$\Delta V_{i,j,k} = \Delta x_i \cdot \Delta y_j \cdot \Delta z_k. \quad (2.108)$$

Cell’s faces’ areas are now given by:

$$A_{x,j,k} = \Delta y_j \cdot \Delta z_k, \quad (2.109a)$$

$$A_{y,i,k} = \Delta x_i \cdot \Delta z_k, \quad (2.109b)$$

$$A_{z,i,j} = \Delta x_i \cdot \Delta y_j. \quad (2.109c)$$

Here, the notation is:

$\Delta x_i, \Delta y_j, \Delta z_k$ = x -, y - and z -dimensions of the parallelepiped cell (i, j, k) . Cells are assumed to be of the same orientation, but not necessarily of the same size. Units: cm.

$A_{x,j,k}, A_{y,i,k}, A_{z,i,j}$ = Surface areas of the parallelepiped cell (i, j, k) normal to the x -, y - and z -axes, respectively. Units: cm².

The resulting finite volume discretization of the multigroup linear NDE is:

$$\begin{aligned}
 \frac{d}{dt} N_{g,i,j,k}(t) = & \\
 & \mathcal{F}_{xg,i-1,j,k}(t) \left(\bar{\phi}_{g,i-1,j,k}(t) - \bar{\phi}_{g,i,j,k}(t) \right) - \mathcal{F}_{xg,i,j,k}(t) \left(\bar{\phi}_{g,i,j,k}(t) - \bar{\phi}_{g,i+1,j,k}(t) \right) + \\
 & \mathcal{F}_{yg,i,j-1,k}(t) \left(\bar{\phi}_{g,i,j-1,k}(t) - \bar{\phi}_{g,i,j,k}(t) \right) - \mathcal{F}_{yg,i,j,k}(t) \left(\bar{\phi}_{g,i,j,k}(t) - \bar{\phi}_{g,i,j+1,k}(t) \right) + \\
 & \mathcal{F}_{zg,i,j,k-1}(t) \left(\bar{\phi}_{g,i,j,k-1}(t) - \bar{\phi}_{g,i,j,k}(t) \right) - \mathcal{F}_{zg,i,j,k}(t) \left(\bar{\phi}_{g,i,j,k}(t) - \bar{\phi}_{g,i,j,k+1}(t) \right) - \\
 & - \Delta V_{i,j,k} \bar{\Sigma}_{tg,i,j,k}(t) \bar{\phi}_{g,i,j,k}(t) + \Delta V_{i,j,k} \sum_{g'=1}^G \left[\bar{\Sigma}_{sgg',i,j,k}(t) \bar{\phi}_{g',i,j,k}(t) \right] + \\
 & + \Delta V_{i,j,k} \sum_{j_f=1}^{J_f} \sum_{g'=1}^G \left[\overline{\chi \nu_p \Sigma}_{fgg',i,j,k}^{j_f}(t) \bar{\phi}_{g',i,j,k}(t) \right] + \sum_{j_f=1}^{J_f} \left[\sum_{m=1}^{M^{j_f}} \chi_{dg,m}^{j_f} \lambda_m^{j_f} C_{m,i,j,k}^{j_f}(t) \right] + \\
 & + S_{exg,i,j,k}(t).
 \end{aligned} \tag{2.110}$$

The discretized diffusion term coefficients are defined by:

$$\mathcal{F}_{xg,i,j,k}(t) = A_{x,j,k} \left[\frac{2\bar{D}_{xg,i,j,k}(t) \bar{D}_{xg,i+1,j,k}(t)}{\Delta x_{i+1} \bar{D}_{xg,i,j,k}(t) + \Delta x_i \bar{D}_{xg,i+1,j,k}(t)} \right], \tag{2.111a}$$

$$\mathcal{F}_{yg,i,j,k}(t) = A_{y,i,k} \left[\frac{2\bar{D}_{yg,i,j,k}(t) \bar{D}_{yg,i,j+1,k}(t)}{\Delta y_{j+1} \bar{D}_{yg,i,j,k}(t) + \Delta y_j \bar{D}_{yg,i,j+1,k}(t)} \right], \tag{2.111b}$$

$$\mathcal{F}_{zg,i,j,k}(t) = A_{z,i,j} \left[\frac{2\bar{D}_{zg,i,j,k}(t) \bar{D}_{zg,i,j,k+1}(t)}{\Delta z_{j+1} \bar{D}_{zg,i,j,k}(t) + \Delta z_k \bar{D}_{zg,i,j,k+1}(t)} \right]. \tag{2.111c}$$

The following nomenclature was used here:

$N_{g,i,j,k}(t)$ = Total number of neutrons in group g , cell (i, j, k) . This quantity is related to $n_g(t, \vec{r})$ through Eq. (2.63). It can yield the group g cell (i, j, k) average scalar flux $\bar{\phi}_{g,i,j,k}(t)$ through Eq. (2.112). Units: neutrons.

$\bar{\phi}_{g,i,j,k}(t)$ = Group g scalar flux, averaged over cell (i, j, k) . It is derived from $\phi_g(t, \vec{r})$ using Eq. (2.62), and is the unknown in Eq. (2.110). Units: neutrons/cm²s.

$\bar{\Sigma}_{sgg',i,j,k}(t)$ = Group g' to g macroscopic isotropic combined scattering cross section, averaged over cell (i, j, k) . It is derived from $\Sigma_{sgg'}(t, \vec{r})$ using Eq. (2.62). Units: cm⁻¹.

$S_{exg,i,j,k}(t)$ = Total group g external source in cell (i, j, k) . It is obtained from $S_{exg}(t, \vec{r})$ through Eq. (2.63). Units: neutrons/s.

$\mathcal{F}_{eg,i,j,k}(t)$ = Discretized diffusion term coefficient used for calculating the net current between cells (i, j, k) and either $(i + 1, j, k)$ (if $e = x$), $(i, j + 1, k)$ (if $e = y$) or $(i, j, k + 1)$ (if $e = z$). Units: cm^2 .

$\bar{D}_{eg,i,j,k}(t)$ = Discretized group g Cartesian axis e directional neutron diffusion coefficient, averaged over cell (i, j, k) . It is derived from $D_{eg}(t, \vec{r})$ using Eq. (2.62). Units: cm .

$N_{g,i,j,k}(t)$ and $\bar{\phi}_{g,i,j,k}(t)$ are related through:

$$\bar{\phi}_{g,i,j,k}(t) = \frac{V_{ng}}{\Delta V_{i,j,k}} N_{g,i,j,k}(t). \quad (2.112)$$

The multigroup DNPEs associated with the NDE are identical to Eq. (2.66), different only in notation:

$$\begin{aligned} \frac{d}{dt} C_{m,i,j,k}^{j_f}(t) = & \sum_{g'=1}^G \left[\overline{\nu_{d,m} \Sigma_{f g',i,j,k}^{j_f}}(t) \bar{\phi}_{g',i,j,k}(t) \right] - \\ & - \lambda_m^{j_f} C_{m,i,j,k}^{j_f}(t) \quad \forall m \in [1, \dots, M^{j_f}], j_f \in [1, \dots, J_f]. \end{aligned} \quad (2.113)$$

Equations (2.110) and (2.113), assuming an appropriate mesh, can be used to model the full core problem. To be solved, they also require initial and boundary conditions (ICs and BCs, respectively). The initial condition is often a steady state with a specified total reactor power; a variety of boundary conditions are possible, depending on the full core geometry.

The two most important types of BCs are the reflective and vacuum BCs, which are specific subsets of Neumann and Dirichlet BCs, respectively. To represent the reflective BC, consider a given boundary cell (I, J, K) . Here we assume that it is on the right boundary along the x -axis. A reflective BC means that no neutrons travel through the boundary, and therefore:

$$\mathcal{F}_{xg,I,J,K}(t) = 0. \quad (2.114)$$

A vacuum BC on the same surface can be approximated in several different ways, the simplest of which is to assume a zero boundary flux. This boundary flux replaces $\bar{\phi}_{g,I+1,J,K}(t)$ with 0 in Eq. (2.110), and adjusts the term coefficient:

$$\bar{\phi}_{g,I+1,J,K}(t) = 0, \quad (2.115a)$$

$$\mathcal{F}_{xg,I,J,K}(t) = A_{x,J,K} \left(\frac{2\bar{D}_{xg,I,J,K}(t)}{\Delta x_I} \right). \quad (2.115b)$$

As described above, the FV discretization of the NDE relies on homogenized properties, and so the resulting nodal fluxes, while reaction rate-preserving, are not resolved over individual pins. This is addressed in the following subsection.

2.1.2.3 Heterogeneous Intranodal Flux Reconstruction

The full core diffusion calculation, when used on a system of homogenized nodes, results in a set of nodal flux averages $\bar{\phi}_{g,i,j,k}(t)$. This set, together with the appropriate multigroup homogenized

cross sections, can be used to evaluate nodal reaction rates. Because they were obtained from homogenized properties, these fluxes are known as “homogeneous fluxes.” The lattice calculations do not homogenize multiple pins together, and therefore provide more detailed intranodal reaction rate profiles, which are scaled by the homogeneous fluxes. These intranodal fluxes are known as “heterogeneous fluxes,” and the process of constructing intranodal flux and reaction rate profiles is called “heterogeneous (intranodal) flux reconstruction.” Because this process is often used to evaluate the power profiles within individual pins (to be used for thermal hydraulic calculations, discussed in subsection 2.1.4), it is also called “pin power reconstruction.”

A number of heterogeneous flux reconstruction techniques exist; Refs. [43] and [44] contain a cursory and a more thorough overview, respectively. The FV discretization of the NDE above implicitly assumes flat homogeneous nodal flux shapes (hence $\bar{\phi}_{g,i,j,k}(t)$ is sufficient to represent the homogeneous flux), but in principle, higher order polynomial shapes may also be used to represent the homogeneous fluxes. In this case, there are multiple unknown group quantities per node (i, j, k) .

A common pin power reconstruction technique, given by Finck [45], assumes a fixed intranodal flux shape and a time-dependent homogeneous nodal flux shape (may be flat), combined as follows:

$$\phi_g(t, \vec{r}) = \bar{\phi}_{g,i,j,k}(t, \vec{r}) \tilde{\phi}_{g,i,j,k}(\vec{r}) \quad \text{with } \vec{r} \in \Delta V_{i,j,k}, \quad (2.116)$$

where

$\bar{\phi}_{g,i,j,k}(t, \vec{r})$ = Homogeneous flux function, representing the solution of the FV-discretized NDE in node $\Delta V_{i,j,k}$. In the discretization in subsection 2.1.2.2, this function is flat over the node. Together with homogenized nodal properties, it is sufficient to evaluate the nodal reaction rates. Units: neutrons/cm²s.

$\tilde{\phi}_{g,i,j,k}(\vec{r})$ = Heterogeneous intranodal flux shape function. This function is obtained through a steady-state lattice calculation. The nodal reaction rates are set by $\bar{\phi}_{g,i,j,k}(t, \vec{r})$, and are distributed across the node through $\tilde{\phi}_{g,i,j,k}(\vec{r})$ via the pin power reconstruction process. Dimensionless.

This concludes the summary of the finite volume discretization of the multigroup NDE and DNPEs with a regular mesh. These equations form the basic useful model for full core spatial kinetics analyses, and, in significantly simplified forms, are used in chapters 6 and 7.

In certain types of analyses, it is of interest to represent the reactor kinetics using several standard parameters. This approach is particularly useful for plant balance problems. It is discussed in the following subsection.

2.1.3 Neutron Point Kinetics

The neutron point kinetics equations (PKEs) are a system of ODEs which model the kinetics of a nuclear reactor by characterizing it with several representative parameters, which are derived from the full solution. They are used to model the full core behavior, and so are typically derived from full core NDE and DNPEs solutions (Eqs. (2.96) and (2.33)).

The PKEs represent the group fluxes as a product of “flux amplitude” $A(t)$ and “flux shape function” $S_{\phi g}(t, \vec{r})$:

$$\phi_g(t, \vec{r}) = S_{\phi g}(t, \vec{r}) A(t). \quad (2.117)$$

$A(t)$ is defined as a weighted full-core integral of $\phi_g(t, \vec{r})$:

$$A(t) = \sum_{g=1}^G \iiint_{V_{tot}} dV \left[\omega_g(\vec{r}) \frac{1}{V_{ng}} \phi_g(t, \vec{r}) \right], \quad (2.118)$$

from which follows the normalization of the flux shape function:

$$\sum_{g=1}^G \iiint_{V_{tot}} dV \omega_g(\vec{r}) \frac{1}{V_{ng}} S_{\phi_g}(t, \vec{r}) = 1. \quad (2.119)$$

The nomenclature here is:

$A(t)$ = Flux amplitude function. With $\omega_g(\vec{r}) = 1$ for all g , $A(t)$ is the total number of neutrons in the reactor. Units (assuming dimensionless $\omega_g(\vec{r})$): neutrons.

$S_{\phi_g}(t, \vec{r})$ = Group g flux shape function. Units (assuming dimensionless $\omega_g(\vec{r})$): $1/\text{cm}^2 \text{ s}$.

V_{tot} = Total reactor volume. Units: cm^3 .

$\omega_g(\vec{r})$ = Group g flux weighting function, also known as the ‘‘neutron importance function.’’ It is often simply 1, but may be another value to result in a smoother, less variable shape function. The adjoint flux (the solution of the adjoint steady state multigroup NDE) is often used, as in Ref. [2, Chapter 16]. The specific choice of $\omega_g(\vec{r})$ does not directly affect the structure of the PKEs.

The PKEs are arrived at through several steps (adapted from Ref. [5, Chapter 7]):

1. Equation (2.117) is used to replace $\phi_g(t, \vec{r})$ in Eqs. (2.96) and (2.33).
2. Equations (2.96) and (2.33) are multiplied by $\omega_g(\vec{r})$, and $\chi_{dg,m}^{j_f} \omega_g(\vec{r})$, respectively.
3. Both equations are integrated over total reactor volume and summed over all groups.

The resulting equations can be written in terms of several integral parameters, defined as reaction rate ratios. Equation (2.120) defines the destruction rate:

$$W_D(t) = \sum_{g=1}^G \iiint_{V_{tot}} dV \left[\omega_g(\vec{r}) \left(-\nabla \cdot \mathbb{D}_g(t, \vec{r}) \nabla S_{\phi_g}(t, \vec{r}) + \right. \right. \\ \left. \left. + \Sigma_{tg}(t, \vec{r}) S_{\phi_g}(t, \vec{r}) - \sum_{g'=1}^G \Sigma_{sgg'}(t, \vec{r}) S_{\phi_{g'}}(t, \vec{r}) \right) \right]. \quad (2.120)$$

Equations (2.121) define the total and family-specific production rates:

$$W_P(t) = \sum_{g=1}^G \sum_{j_f=1}^{J_f} \sum_{g'=1}^G \iiint_{\mathbf{V}_{tot}} dV \left[\omega_g(\vec{\mathbf{r}}) \left(\chi \nu_p \Sigma_{f g g'}^{j_f}(t, \vec{\mathbf{r}}) + \sum_{m=1}^{M^{j_f}} \nu_{d,m} \Sigma_{f g'}^{j_f}(t, \vec{\mathbf{r}}) \right) S_{\phi g'}(t, \vec{\mathbf{r}}) \right], \quad (2.121a)$$

$$W_{P,m}^{j_f}(t) = \sum_{g=1}^G \sum_{g'=1}^G \iiint_{\mathbf{V}_{tot}} dV \left[\omega_g(\vec{\mathbf{r}}) \chi_{dg,m}^{j_f} \nu_{d,m} \Sigma_{f g'}^{j_f}(t, \vec{\mathbf{r}}) S_{\phi g'}(t, \vec{\mathbf{r}}) \right]. \quad (2.121b)$$

Equations (2.122) define the reactor reactivity, family-specific delayed neutron fraction, total delayed neutron fraction and prompt neutron lifetime, respectively:

$$\rho(t) = \frac{W_P(t) - W_D(t)}{W_P(t)}, \quad (2.122a)$$

$$\beta_m^{j_f}(t) = \frac{W_{P,m}^{j_f}(t)}{W_P(t)}, \quad (2.122b)$$

$$\beta(t) = \sum_{j_f=1}^{J_f} \sum_{m=1}^{M^{j_f}} \beta_m^{j_f}(t), \quad (2.122c)$$

$$\Lambda(t) = \frac{1}{W_P(t)}. \quad (2.122d)$$

Equations (2.121) may be divided by k_{eff}^0 to adjust the initial reactivity and prompt neutron lifetime. With k_{eff} as the reactor eigenvalue at a given state, and assuming the corresponding flux eigenfunction is the eigenfunction at this state (therefore, not true in a transient), ρ may also be related to k_{eff} directly:

$$\rho = \frac{k_{eff} - 1}{k_{eff}}. \quad (2.123)$$

The point kinetics external source and total number of precursors in a family are defined by:

$$S_{ex}(t) = \sum_{g=1}^G \iiint_{\mathbf{V}_{tot}} dV \left[\omega_g(\vec{\mathbf{r}}) S_{exg}(t, \vec{\mathbf{r}}) \right], \quad (2.124)$$

$$C_m^{j_f}(t) = \sum_{g=1}^G \iiint_{\mathbf{V}_{tot}} dV \left[\omega_g(\vec{\mathbf{r}}) \chi_{dg,m}^{j_f} C_m^{j_f}(t, \vec{\mathbf{r}}) \right]. \quad (2.125)$$

Together, these parameters can be combined into the point kinetics equations (adapted from Ref. [5, Chapter 7]):

$$\frac{d}{dt} A(t) = \frac{\rho(t) - \beta(t)}{\Lambda(t)} A(t) + \sum_{j_f=1}^{J_f} \sum_{m=1}^{M^{j_f}} \lambda_m^{j_f} C_m^{j_f}(t) + S_{ex}(t), \quad (2.126a)$$

$$\frac{d}{dt} C_m^{j_f}(t) = \frac{\beta_m^{j_f}(t)}{\Lambda(t)} A(t) - \lambda_m^{j_f} C_m^{j_f}(t) \quad \forall m \in [1, \dots, M^{j_f}], j_f \in [1, \dots, J_f]. \quad (2.126b)$$

The following nomenclature was used:

- $W_D(t)$ = Total neutron destruction rate. Units: neutrons/s.
 $W_P(t)$ = Total neutron production rate. Units: neutrons/s.
 $W_{P,m}^{j_f}(t)$ = Total neutron production rate through precursor family m of nuclide j_f . Units: neutrons/s.
 $\rho, \rho(t)$ = Reactor reactivity. $\rho(t)$ is the true (“dynamic”) reactor reactivity, which may be evaluated for any flux configuration. If, instead ρ is computed with the flux eigenfunctions, it is known as the “static” reactor reactivity. In this case, the reactor is critical with $\rho = 0$; a reactor with $\rho > 0$ or $\rho < 0$ is supercritical and subcritical, respectively. Dimensionless.
 $\beta_m^{j_f}(t)$ = Effective delayed neutron fraction of precursor family m produced by fissions of nuclide j_f . Dimensionless.
 $\beta(t)$ = Total effective delayed neutron fraction. Dimensionless.
 $\Lambda(t)$ = Prompt neutron lifetime. This quantity, approximately, quantifies the length of time that an average neutron survives prior to being destroyed. Units: s.
 $S_{ex}(t)$ = Total external source. Units: neutrons/s.
 $C_m^{j_f}(t)$ = Total number of precursors of family m from nuclide j_f in the reactor. Units: precursors.

Equations (2.126) are formal, but exact, as long as the parameters $\rho(t)$, $\{\beta_m^{j_f}(t)\}$, $\Lambda(t)$ and $S_{ex}(t)$ are exact. They are all defined in terms of $\phi_g(t, \vec{r})$, and so are not directly usable.

To make them usable, additional assumptions are made:

1. $\{\beta_m^{j_f}(t)\}$ and $\Lambda(t)$ are assumed to vary sufficiently little during the transient of interest to be treated as constant (based on the shape function during the beginning of the transient). Usually, for this to be true, the group fluxes are approximated as separable:

$$\phi_g(t, \vec{r}) \cong S_{\phi_g}(\vec{r}) A(t), \quad (2.127)$$

in which:

$S_{\phi_g}(\vec{r})$ = The group g flux shape, assumed invariant. Normally, the initial (in most transients, steady state) flux shape is used. This quantity is an approximation for $S_{\phi_g}(t, \vec{r})$.
 Units: $1/\text{cm}^2 \text{ s}$.

Note, that even with the flux shape assumed invariant, $\{\beta_m^{j_f}(t)\}$ and $\Lambda(t)$ do not necessarily remain invariant: the macroscopic cross sections themselves may vary (due to, for example, rod movement). $\{\beta_m^{j_f}(t)\}$ and $\Lambda(t)$ are assumed to remain invariant, by similarly neglecting the subsequent variation in them.

2. The weighting functions are chosen sufficiently well to make assumption 1 acceptable.

A common additional simplification is that a single common set of M precursor families

is used. As discussed in subsection 2.1.1.2, this may be either: (a) by assuming a principal fissionable isotope, (b) by using a nuclear database with common precursor families, like JEFF-3.1, or (c) here, simply a notational simplification: m may index an (m, j_f) pair, and M be the total number of such pairs. The PKEs with constant parameters and with a single common set of precursor families become:

$$\frac{d}{dt}A(t) = \frac{\rho(t) - \beta}{\Lambda}A(t) + \sum_{m=1}^M \lambda_m C_m(t) + S_{ex}(t), \quad (2.128a)$$

$$\frac{d}{dt}C_m(t) = \frac{\beta_m}{\Lambda}A(t) - \lambda_m C_m(t) \quad \forall m \in [1, \dots, M]. \quad (2.128b)$$

Here the new nomenclature is:

- β_m = Effective reactor-wide delayed neutron fraction for the shared family m . Dimensionless.
- β = Effective reactor-wide delayed neutron fraction, assumed constant. Dimensionless.
- Λ = Prompt neutron lifetime, assumed constant. Units: s.
- $C_m(t)$ = Total number of precursors in a shared family m . Units: precursors.
- M = Total number of precursor families.

In Eqs. (2.128), $\rho(t)$ is frequently given in terms of β , or “dollars.” $1\beta = 1\$$.

The quasistatic reactor kinetics approach consists of executing the following three operations at every time step:

1. The steady state neutron transport equation (or the neutron diffusion equation) is solved to produce a flux shape function. Optionally, the adjoint equation (see Ref. [2, Chapter 16]) may also be solved, to produce the neutron importance function.
2. Reactor reactivity, DNFs and prompt neutron lifetime are recomputed based on the new shape function.
3. Equations (2.126) are integrated to the next time step.

This concludes the summary of various forms of uncoupled neutron transport analysis of nuclear reactors. In the next subsection, we discuss thermal hydraulics, the other important physics of interest.

2.1.4 Thermal Hydraulics

Three types of heat transfer mechanisms take place in a nuclear reactor. The first type is thermal conduction, which governs heat diffusion within solids. The second type is convection, which governs the removal of thermal energy from solids’ surfaces by the fluids. Fluids then transfer the thermal energy by advection. The third type is thermal radiation, which occurs as infrared radiation by the heated surfaces.

Thermal radiation is typically neglected in reactor analysis, because the materials are opaque, and the other two mechanisms dominate the heat transfer. It is not negligible outside of the reactor, where the pipes, in particular, do lose heat by infrared radiation. Reactor thermal hydraulics generally concern thermal conduction, convection and fluid advection.

This subsection details: 1) the fundamental equations governing fluid transport, 2) the fundamental equations governing heat transfer in solids, and 3) specific applications of these equations to various types of reactor analysis.

2.1.4.1 Fundamental Fluid Transport Equations

The basic assumptions made for fluid transport analysis of nuclear reactors are [3, Chapter 4]:

1. The fluid is sufficiently dense to be treated as a continuum. This is always the case in a nuclear reactor.
2. The geometric scale is sufficiently resolved to treat the fluid at any given point as single-phase. This is acceptable for the mathematical formulation of the equations, but in practice, mesh elements often cannot be refined to the point of meshing individual bubbles, and models are required to account for this.
3. The fluid is Newtonian: the viscous stresses are proportional to the strains, the stress-strain relation is isotropic, and bulk viscosity is negligible. This is true for most coolants in nuclear reactors, including light and heavy water, but may not necessarily be true for corium (molten core materials) [46].
4. The only external force acting on the fluid is gravity: there are no electromagnetic or other similar external forces. This is true in light and heavy water reactors, but may, in principle, be violated within electromagnetic pumps in liquid metal-cooled reactors, or in other similar situations.
5. The gravitational acceleration is assumed to be constant and unidirectional, which is generally true for all stationary reactors; it may be inappropriate for aerospace applications.
6. Heat transfer via thermal radiation is neglected. As discussed above, this is a sound assumption in nuclear reactors.

Under these assumptions, fluid flow is covered by mass, momentum and energy conservation equations. Together, these constitute 5 scalar equations (the momentum is a 3D vector field), and there are 5 unknowns. The momentum (or velocity) components are 3 of the unknowns, and the other 2 are intensive properties, chosen depending on the formulation of the problem. Below, fluid pressure and temperature are chosen as the unknown intensive properties. All other intensive properties, including fluid viscosity and density, are formulated as functions of the unknowns; fluid property tables (e.g., Refs. [47–49]) are used to relate the known and the unknown intensive properties.

Together, the 3 equations below are known as the Navier-Stokes equations.

Under the above assumptions, the mass conservation equation (also known as the “continuity equation”) is given by [3, Chapter 4]:

$$\frac{\partial}{\partial t}\rho(p, T) = -\nabla \cdot \left(\rho(p, T) \vec{V}(t, \vec{r}) \right), \quad (2.129)$$

with:

- $p = p(t, \vec{r})$ = Fluid pressure. In this formulation, this is an unknown intensive property. Units: Pa.
- $T = T(t, \vec{r})$ = Fluid temperature. In this formulation, this is an unknown intensive property. Units: K.
- $\rho(p, T)$ = Fluid density. This is an intensive property, formulated here as a known function of fluid temperature and pressure. Units: kg/m³.
- $\vec{V} = \vec{V}(t, \vec{r})$ = Fluid velocity vector. This is an unknown 3D vector field. Units: m/s.

Under the above assumptions, the momentum conservation equation (also sometimes individually referred to as the “Navier-Stokes equation”), are given by [3, Chapter 4]:

$$\rho(p, T) \frac{D}{Dt} \vec{V}(t, \vec{r}) = -\nabla p(t, \vec{r}) + \nabla \cdot \mathbb{T}(p, T, \vec{V}) + \rho(p, T) \vec{g}. \quad (2.130)$$

The Lagrangian derivatives of scalar and vector fields are defined by:

$$\frac{D}{Dt} f(t, \vec{r}) = \frac{\partial}{\partial t} f(t, \vec{r}) + \vec{V}(t, \vec{r}) \cdot \nabla f(t, \vec{r}), \quad (2.131a)$$

$$\begin{aligned} \frac{D}{Dt} \vec{f}(t, \vec{r}) &= \frac{\partial}{\partial t} \vec{f}(t, \vec{r}) + \vec{V}(t, \vec{r}) \cdot \nabla \vec{f}(t, \vec{r}) = \\ &= \frac{D}{Dt} f_x(t, \vec{r}) \hat{x} + \frac{D}{Dt} f_y(t, \vec{r}) \hat{y} + \frac{D}{Dt} f_z(t, \vec{r}) \hat{z}. \end{aligned} \quad (2.131b)$$

The following notation is used:

$f(t, \vec{r})$ = A scalar field.

$\vec{f}(t, \vec{r})$ = A vector field.

$f_e(t, \vec{r})$ = e -directed component of $f(t, \vec{r})$.

$\frac{D}{Dt} f(t, \vec{r})$ = Lagrangian derivative of a scalar field $f(t, \vec{r})$.

$\frac{D}{Dt} \vec{f}(t, \vec{r})$ = Lagrangian derivative of a vector field $\vec{f}(t, \vec{r})$.

$\mathbb{T}(p, T, \vec{V})$ = Viscous stress tensor, defined by Eq. (2.132). This is a 3×3 symmetric tensor which characterizes the stresses on the fluid at a given point due to viscous forces. Units: Pa s.

The viscous stress tensor $\mathbb{T}(p, T, \vec{V})$ is defined by:

$$\mathbb{T}(p, T, \vec{V}) = \begin{bmatrix} \tau_{1,1}(p, T, \vec{V}) & \tau_{1,2}(p, T, \vec{V}) & \tau_{1,3}(p, T, \vec{V}) \\ \tau_{2,1}(p, T, \vec{V}) & \tau_{2,2}(p, T, \vec{V}) & \tau_{2,3}(p, T, \vec{V}) \\ \tau_{3,1}(p, T, \vec{V}) & \tau_{3,2}(p, T, \vec{V}) & \tau_{3,3}(p, T, \vec{V}) \end{bmatrix}, \quad (2.132)$$

with:

$$\tau_{i,j}(p, T, \vec{V}) = \begin{cases} 2\mu(p, T) \frac{\partial}{\partial e_i} V_{e_i}(t, \vec{r}) - \frac{2}{3}\mu(p, T) \nabla \cdot \vec{V}(t, \vec{r}) & \text{if } i = j, \\ \mu(p, T) \left(\frac{\partial}{\partial e_i} V_{e_j}(t, \vec{r}) + \frac{\partial}{\partial e_j} V_{e_i}(t, \vec{r}) \right) & \text{if } i \neq j, \end{cases} \quad (2.133)$$

and

$$e_1 = x, \quad (2.134a)$$

$$e_2 = y, \quad (2.134b)$$

$$e_3 = z. \quad (2.134c)$$

Here the notation is:

$\tau_{i,j}(p, T, \vec{V})$ = Viscous stress acting on the surface of an infinitesimal fluid element, defined by Eq. (2.133). The surface is oriented normal to the e_i -direction, the stress is directed in the e_j -direction. With $i = j$, this is the compressive or tensile stress component along the e_i -direction. With $i \neq j$, this is a shear stress component. Units: Pa.

$\mu(p, T)$ = Fluid dynamic viscosity. This is an intensive property, formulated here as a known function of fluid temperature and pressure. Units: Pa s.

e_i = i^{th} Cartesian direction, defined by Eqs. (2.134).

\vec{g} = Gravitational acceleration vector, assumed constant and unidirectional. Units: m/s².

Lastly, under the above assumptions, the energy conservation equation is given by [3, Chapter 4]:

$$\rho(p, T) \frac{D}{Dt} h(p, T) = \nabla \cdot k(p, T) \nabla T(t, \vec{r}) + \frac{D}{Dt} p(t, \vec{r}) + \Phi(p, T, \vec{V}) + \dot{u}_{v,ex}(t, \vec{r}). \quad (2.135)$$

The energy dissipation function $\Phi(p, T, \vec{V})$ is defined as:

$$\Phi(p, T, \vec{V}) = \mathbb{T}(p, T, \vec{V}) : \nabla \vec{V} = \nabla \cdot \left[\mathbb{T}(p, T, \vec{V}) \cdot \vec{V} \right] - \vec{V} \cdot \left[\nabla \cdot \mathbb{T}(p, T, \vec{V}) \right]. \quad (2.136)$$

Here, the nomenclature is:

$h(p, T)$ = Fluid specific enthalpy. This is an intensive property, formulated here as a known function of fluid temperature and pressure. Units: kJ/kg.

$k(p, T)$ = Fluid thermal conductivity. This is an intensive property, formulated here as a known function of fluid temperature and pressure. Units: W/m K.

$\Phi(p, T, \vec{V})$ = Energy dissipation function. This quantity accounts for the work done on the fluid by the viscous forces, and is given by Eq. (2.136). It is often negligible. Units: W/m³.

$\dot{u}_{v,ex}(t, \vec{r})$ = External thermal energy source term. This quantity accounts for the direct energy deposition into the fluid, discussed in subsection 2.1.5.2. Units: W/m³.

Besides heat removal, fluid transport in a reactor is also responsible for solute transport. Assuming the solute concentrations are sufficiently low to not affect the fluid thermodynamic properties, the solutions for the density and velocity fields may be used to calculate the resultant solute concentrations in the fluid.

Energy transfer to the fluid, besides direct energy deposition, is imposed through the boundary conditions. If multiple immiscible fluids, or two phases, are present, the above equations still hold, but become more complicated to solve, because of the interphase boundaries. This phenomenon is frequently addressed through mixing and two-phase models.

This concludes the summary of the equations that govern fluid transport in a reactor. Heat diffusion in solids can be modeled similarly to Eq. (2.135), but with significant simplifications, discussed below.

2.1.4.2 Heat Diffusion Equation

Heat diffusion in solids in a reactor is modeled based on the following assumptions (adapted from Ref. [3, Chapter 8]):

1. Thermal expansion of the material is neglected, and all geometric deformations are due to known time-dependent movement of the solid parts, like the control assemblies.
2. Temperature is assumed to be a known function of the volumetric thermal energy density (also referred to as “volumetric internal energy density”). Because the material is assumed incompressible, this is necessarily true, as shown below.

In material properties databases (e.g., Ref. [50]), volumetric thermal energy density $u_v(T)$ is typically not given. Instead, for a given composition, material density $\rho(T)$ and specific heat capacity $c_p(T)$ are given. Together, these parameters yield the volumetric heat capacity $c_v(T)$, which, by definition, is the rate of change in volumetric thermal energy density due to temperature change:

$$\frac{d}{dT}u_v(T) = c_v(T) = \rho(T) c_p(T). \quad (2.137)$$

Assuming a zero internal energy density at some reference temperature T_0 , we solve Eq. (2.137) via separation of variables to obtain $u_v(T)$:

$$u_v(T) = \int_{T_0}^T dT c_v(T). \quad (2.138)$$

Because $c_v(T)$ is always positive, $u_v(T)$ is monotonically increasing, and therefore invertible. Its inverse yields $T(u_v)$, the temperature as a function of volumetric internal energy density.

The nomenclature used here was:

$u_v(T)$ = Material thermal energy density as a function of temperature. This is derived from tabulated, known quantities. Units: kJ/m³.

$c_v(T)$ = Material volumetric heat capacity as a function of temperature. This is derived from tabulated, known quantities. Units: $\text{kJ}/\text{m}^3 \text{K}$.

$\rho(T)$ = Material density as a function of temperature. This is a tabulated, known quantity. Units: kg/m^3 .

$c_p(T)$ = Material specific heat capacity. This is a tabulated, known quantity. Units: $\text{kJ}/\text{kg K}$.

T_0 = Reference temperature, with $u_v(T_0) = 0$. It may be chosen arbitrarily; it is easiest to choose room temperature $298 \text{ K} \approx 25^\circ \text{C}$ as reference, because this is usually the lowest temperature at which nuclear material properties are tabulated [50]. Units: K .

Heat diffusion in solids is governed by the heat diffusion equation (HDE) [3, Chapter 8]:

$$\frac{\partial}{\partial t} u_v(t, \vec{r}) = \nabla \cdot k(t, \vec{r}, T) \nabla T(t, \vec{r}, u_v) + \dot{u}_{v,ex}(t, \vec{r}). \quad (2.139)$$

Here, the following nomenclature was used:

$u_v(t, \vec{r})$ = Volumetric thermal energy density profile. This is the unknown in Eq. (2.139). Units: kJ/m^3 .

$k(t, \vec{r}, T)$ = Material thermal conductivity. This is a known material property. The time and position dependence is to account for material variation and movement. Units: $\text{W}/\text{m K}$.

$T(t, \vec{r}, u_v)$ = Material temperature profile. The time and position dependence is to account for material variation and movement. Units: K .

$\dot{u}_{v,ex}(t, \vec{r})$ = External thermal energy source term. In solids, this quantity accounts for fission heat generation (approximately 90% of the reactor's thermal power [51]), and for γ energy deposition into the structural elements (approximately 1% of the reactor's thermal power [51]). This quantity is discussed in greater detail in subsection 2.1.5.2. Units: W/m^3 .

Practical discretization methods for Eq. (2.139) are given in subsection 2.1.4.6.

Thermal energy removal from solids is governed by the boundary conditions. Together, Eqs. (2.129), (2.130), (2.135), coupled via BCs to Eq. (2.139), describe the fluid and heat transport in a nuclear reactor. This system of nonlinear PDEs is not analytically solvable, except in primitive special cases, and so must be solved numerically using appropriate discretizations. They are rarely solved directly for the full core geometry, and are instead normally solved under a number of simplifying assumptions. As with neutron transport analysis, different types of codes are used for different geometries and sets of assumptions; the coarsest, largest scale thermal hydraulic analysis is performed using the systems codes, described in the following subsection.

2.1.4.3 Systems-level Simulation

Systems codes are primarily used for plant balance problems. The reactor is part of such a problem, and so they can be used to model the thermal hydraulics of the vessel under a number of simplifying assumptions to the fundamental thermal hydraulic equations. These assumptions vary depending on the type of system the code is intended for; in this subsection, the assumptions

and resulting equations for RELAP5-3D are described [52, Volume 1] — they are typical for a systems-level analysis of water-cooled reactors.

The simplifying assumptions behind RELAP5-3D are:

1. In any given section, a single phase is assumed to be sufficiently well-mixed for its state to be adequately represented through bulk properties. This is the most fundamental assumption in systems-level analysis; if it cannot be made, systems-level analysis is inapplicable.
2. The flow is treated as a 1-dimensional, bulk fluid flow. This means that transverse momentum components and property variations are neglected. This is clearly a good approximation in pipe-like geometries, and gets worse as the geometry becomes more multidimensional. Stratified flow, which assumes a transverse hydrostatic pressure gradient in a horizontal channel, may be implemented via a special model; it effectively adjusts the pressure gradients in the momentum conservation equations. Below, “position” refers to the cross-sectional flow area at some x -coordinate along the 1-dimensional streamline.
3. The mass, momentum and energy conservation equations are written separately for the liquid and gas phases. In any given section, the phases are at the same pressure, as is the interphase interface, but they may have different temperatures and velocities. This model is adequate as long as the interphase terms are adequately modeled.
4. A single, potentially two-phase working fluid (e.g., light water), but multiple noncondensable gases may be present. At any given position, the noncondensables have the same velocity and temperature as the vapor at this position. This is normally an acceptable assumption because gases are prone to mixing, and in bulk motion, are rarely stratified enough with one another to have significantly different velocity and temperature.
5. Solutes, if present, are sufficiently dilute to not affect the properties of the solvent phase (liquid for boron, liquid or gas for other radionuclides), and share the velocity with the solvent phase. Solutes do not absorb or produce thermal energy, and do not contribute to the solvent phase’s inertia.
6. There is no axial heat diffusion in the fluid: all axial heat transfer is advective. This is a completely acceptable approximation in non-metallic fluids (and other fluids with a sufficiently high Prandtl number), but in a natural liquid metal flow, it may be unacceptable.
7. Heat structures (solids) are treated as 1-dimensional, with a given heat transfer area. This is an acceptable assumption for geometries cooled by one or two fluids, but for geometries cooled by more than two fluids (e.g., a radially closed parallel plate channel with different fluids on the two outer sides of the plates), it is unacceptable. Such geometries are rarely present in reactors.
8. A lumped parameter approach is used: the geometry is specified as a combination of discontinuous lumped volumes, connected with junctions. This approach is acceptable as long as the terms used to approximate bulk mass, momentum and heat transfer are sufficiently accurate for the modeled phenomena. It cannot adequately model local power or temperature peaking, and so is primarily used for large-scale, plant and system balance models.

9. Reynolds-averaged Navier-Stokes (RANS) model (see Ref. [3, Chapter 10]) for turbulence is used, with the Reynolds stresses and heat fluxes neglected. This, in effect, assumes that only the mean flow is present, and that the high-frequency fluctuations are negligible. Velocity covariance terms are also neglected. These approximations are a source of inaccuracy, but are made to make the problem feasible to solve. Making these approximations significantly reduces the size of the problem.
10. Intrapphase viscous stresses are neglected (but interfacial viscous stresses are not). Wall friction is not neglected, and is used to account for intraphase viscous friction. By design, reactor coolants are intentionally low-viscous fluids, therefore this assumption is acceptable.
11. Interfacial energy storage (energy stored in the tension of the interphase interfaces) is neglected. This energy is very small compared to the other energy terms, therefore this assumption is acceptable.
12. The wall and interfacial friction, and interphase heat and mass transfer are computed based on models (combinations of correlations) which are part of the code. These models rely on various geometric parameters, and, heavily, on the flow regime, which the code computes based on the fluid state and velocity profile in the region. The flow regime map is optimized for a specific fluid, and is a significant source of uncertainty in systems codes [14, Chapter 4].

Under the above assumptions, Eq. (2.139) is not formally changed, but becomes 1-dimensional. However, Eqs. (2.129), (2.130) and (2.135) are significantly modified, and split into multiple equations, to account for the two phases, multiple noncondensables, and solute (in RELAP5-3D, boron) concentration. Again, the entire liquid phase is assumed to be of a single working fluid, and the gas phase consists of the working fluid vapor and the noncondensables, if any are present.

With N_{nc} noncondensables and N_s solutes, the unknowns are:

$p, p(t, x)$ = Fluid pressure, the same for all phases and species at a given position. When the stratified flow model in a horizontal channel is used, a transverse pressure profile is assumed. Units: Pa.

$u_f, u_f(t, x)$ = Liquid phase bulk specific heat at a given position. Units: kJ/kg.

$u_g, u_g(t, x)$ = Gas phase bulk specific heat at a given position. Units: kJ/kg.

$\alpha_g, \alpha_g(t, x)$ = Bulk void fraction at a given position. This quantity ranges between 0 and 1, which correspond to single liquid and gas phases, respectively. Subcooled liquid and superheated vapor states are possible, but $\alpha_g(t, x)$ cannot be below 0 or above 1. Noncondensables count as parts of the gas phase for the purpose of $\alpha_g(t, x)$ evaluation. Dimensionless.

$V_{xf}(t, x)$ = Liquid phase bulk velocity in the x -direction, at a given position. Units: m/s.

$V_{xg}(t, x)$ = Gas phase bulk velocity in the x -direction, at a given position. Units: m/s.

$\chi_{nc}(t, x)$ = Total noncondensable quality. This is the ratio of the sum of all noncondensable masses to the overall gas mass at a given position (i.e., it is the noncondensable mass fraction). This quantity ranges between 0 and 1, with all gas being the working fluid vapor at $\chi_{nc}(t, x) = 0$, and all gas being noncondensable at $\chi_{nc}(t, x) = 1$. Dimensionless.

$\chi_{nc}^n(t, x)$ = Individual noncondensable gas n quality. This is the ratio of noncondensable gas n mass to the sum of all noncondensable masses at a given position (i.e., it is the specie n mass fraction among other noncondensable species). There are N_{nc} individual noncondensable gas qualities, which constitutes $N_{nc} - 1$ unknowns: they add up to 1. This quantity ranges between 0 and 1, with 0 meaning that noncondensable n is not present at the given position, and 1 meaning that noncondensable n is the only noncondensable specie at the given position. Dimensionless.

$\rho^s, \rho^s(t, x)$ = Solute s “spatial density.” A spatial density is the mass density in overall fluid volume, including the volume occupied by the phase not occupied by the volume. This is different from the phasic densities $\rho_g(p, u_g, \vec{\chi}_{nc})$ and $\rho_f(p, u_f)$, which are the ratios of the phases’ masses to their own volumes. There are a total of N_s unknown individual solute spatial mass and number densities. RELAP5-3D tracks the spatial density for boron, and number densities $N^s(t, x)$ for radionuclide solutes; the two quantities are linearly proportional to each other. Units: kg/m^3 .

$N^s(t, x)$ = Solute s number density. This quantity is the same as $N^j(t, \vec{r})$ in Eq. (2.7), for a given radionuclide solute. It is linearly proportional to the spatial density $N^s(t, x)$ for solute s . There are a total of N_s unknown individual solute spatial mass and number densities. Units: atoms/m^3 .

The resulting mass conservation equations become [52, Volume 1]:

$$\frac{\partial}{\partial t} \left[\alpha_g(t, x) \rho_g(p, u_g, \vec{\chi}_{nc}) \right] = -\frac{1}{A_x(x)} \frac{\partial}{\partial x} \left[\alpha_g(t, x) \rho_g(p, u_g, \vec{\chi}_{nc}) V_{xg}(t, x) A_x(x) \right] + \Gamma_{gf}(t, x, \vec{x}), \quad (2.140a)$$

$$\frac{\partial}{\partial t} \left[\alpha_f(\alpha_g) \rho_f(p, u_f) \right] = -\frac{1}{A_x(x)} \frac{\partial}{\partial x} \left[\alpha_f(\alpha_g) \rho_f(p, u_f) V_{xf}(t, x) A_x(x) \right] + \Gamma_{fg}(t, x, \vec{x}). \quad (2.140b)$$

The liquid volume and void fractions add up to 1:

$$\alpha_f(\alpha_g) = 1 - \alpha_g(t, x). \quad (2.141)$$

Equations (2.142) and (2.143) model the total and individual noncondensable mass conservation [52, Volume 1]:

$$\begin{aligned} \frac{\partial}{\partial t} \left[\alpha_g(t, x) \rho_g(p, u_g, \vec{\chi}_{nc}) \chi_{nc}(t, x) \right] &= \\ &= -\frac{1}{A_x(x)} \frac{\partial}{\partial x} \left[\alpha_g(t, x) \rho_g(p, u_g, \vec{\chi}_{nc}) \chi_{nc}(t, x) V_{xg}(t, x) A_x(x) \right], \end{aligned} \quad (2.142)$$

$$\begin{aligned} \frac{\partial}{\partial t} \left[\alpha_g(t, x) \rho_g(p, u_g, \vec{\chi}_{nc}) \chi_{nc}(t, x) \chi_{nc}^n(t, x) \right] &= \\ &= -\frac{1}{A_x(x)} \frac{\partial}{\partial x} \left[\alpha_g(t, x) \rho_g(p, u_g, \vec{\chi}_{nc}) \chi_{nc}(t, x) \chi_{nc}^n(t, x) V_{xg}(t, x) A_x(x) \right]. \end{aligned} \quad (2.143)$$

Individual noncondensable qualities are normalized to 1:

$$\sum_{n=1}^{N_{nc}} \chi_{nc}^n(t, x) = 1. \quad (2.144)$$

Solute mass conservation equations in terms of spatial density and number density are given by Eqs. (2.145a) and (2.145b), respectively. The transporting phase is denoted as ϑ , and, in both equations, may refer to liquid or gas transporting phases. As stated above, the two equations are fully equivalent:

$$\frac{\partial}{\partial t} \rho^s(t, x) = -\frac{1}{A_x(x)} \frac{\partial}{\partial x} \left[\rho^s(t, x) V_{x\vartheta}(t, x) A_x(x) \right], \quad (2.145a)$$

$$\frac{\partial}{\partial t} N^s(t, x) = -\frac{1}{A_x(x)} \frac{\partial}{\partial x} \left[N^s(t, x) V_{x\vartheta}(t, x) A_x(x) \right] + \dot{S}^s(t, x, \vec{\mathbf{x}}). \quad (2.145b)$$

If no noncondensables are present, Eqs. (2.142)–(2.144) do not apply. If no solutes are present, Eqs. (2.145) do not apply. In a single-phase system, one of Eqs. (2.140) does not apply, although RELAP5-3D models single-phase fluids as two-phase with zero mass in one of the phases.

The following nomenclature was used:

- x = Position along the 1-dimensional streamline, with which some flow area $A_x(x)$ is associated. Units: m.
- $A_x(x)$ = Flow area, orthogonal to the 1-dimensional streamline, at a given position along this streamline. Units: m^2 .
- $\vec{\mathbf{x}}$ = Thermohydraulic state vector. Many terms in systems-level equations rely on models (here, combinations of correlations); such models may be functions of any of the unknown variables, either at the location of interest, or elsewhere in the system. Here, dependence on the state vector formally shows that depending on the model, the quantity may be a linear or nonlinear function of any of the unknowns, or quantities derivable from the unknowns.
- $\vec{\mathbf{x}}_{nc}$ = The noncondensable qualities vector. This vector contains both the individual and total noncondensable qualities. Known properties of the gas phase, like its mass density $\rho_g(p, u_g, \vec{\mathbf{x}}_{nc})$, are dependent on the total and individual noncondensable qualities, because the gas phase is assumed to be a mixture of the noncondensables and the working fluid vapor. Dimensionless.
- $\alpha_f(\alpha_g)$ = Bulk liquid phase volume fraction at a given position. This quantity ranges between 0 and 1, which correspond to single gas and liquid phases, respectively. Subcooled liquid and superheated vapor states are possible, but $\alpha_f(\alpha_g)$ cannot be below 0 or above 1. Dimensionless.
- $\rho_g(p, u_g, \vec{\mathbf{x}}_{nc})$ = Gas phase density. This is the mass density (the ratio of the mass of the phase to the volume it occupies) of the gas phase, which may be composed of the working fluid vapor, and one or more noncondensables, all at the same temperature and pressure. Units: kg/m^3 .
- $\rho_f(p, u_f)$ = Liquid phase density. This is the mass density (the ratio of the mass of the phase to the volume it occupies) of the liquid phase, which is fully composed of the working fluid liquid. Units: kg/m^3 .

$\Gamma_{gf}(t, x, \vec{\mathbf{x}})$ = Vapor generation rate density. This is the spatial density of the rate of the liquid-to-vapor mass conversion. It is computed through a vaporization model; see Ref. [52, Volume 1] for details. Units: kg/m³ s.

$\Gamma_{fg}(t, x, \vec{\mathbf{x}})$ = Liquid generation rate density. This is the spatial density of the rate of the vapor-to-liquid mass conversion. It is computed through a condensation model; see Ref. [52, Volume 1] for details. Units: kg/m³ s.

$V_{x\vartheta}(t, x)$ = Phase ϑ velocity. $\vartheta = f$ for liquid, and $\vartheta = g$ for gas. Units: m/s.

$\dot{S}^s(t, x, \vec{\mathbf{x}})$ = Radionuclide solute s spatial net source rate density. This quantity accounts for the generation, via neutron-induced reactions and radioactive decay, of radionuclides of interest. For the purposes of systems-level codes, it is model-based. Units: atoms/m³ s.

The mass conservation equations above are species-specific, but by the above assumptions, velocity and energy conservation equations are not.

The momentum conservation equations are [52, Volume 1]:

$$\begin{aligned} \mathcal{L}_g^\rho(x, \alpha_g, p, u_g, \vec{\mathbf{x}}_{nc}) \frac{\partial}{\partial t} V_{xg}(t, x) &= -\frac{1}{2} \mathcal{L}_g^\rho(x, \alpha_g, p, u_g, \vec{\mathbf{x}}_{nc}) \frac{\partial}{\partial x} \left[\left(V_{xg}(t, x) \right)^2 \right] - \\ &- \mathcal{L}_g^p(x, \alpha_g) \frac{\partial}{\partial x} p(t, x) - \mathcal{L}_g^\rho(x, \alpha_g, p, u_g, \vec{\mathbf{x}}_{nc}) K_{wg}(x, \vec{\mathbf{x}}) V_{xg}(t, x) + \\ &+ \mathcal{L}_g^\rho(x, \alpha_g, p, u_g, \vec{\mathbf{x}}_{nc}) b_x(t, x) + A_x(x) \Gamma_{gf}(t, x, \vec{\mathbf{x}}) \left[V_{xgI}(t, x, \vec{\mathbf{x}}) - V_{xg}(t, x) \right] - \\ &- \mathcal{L}_g^\rho(x, \alpha_g, p, u_g, \vec{\mathbf{x}}_{nc}) K_{ig}(x, \vec{\mathbf{x}}) \left[V_{xg}(t, x) - V_{xf}(t, x) \right] - \\ &- K_{vm}(x, \vec{\mathbf{x}}) \mathcal{L}_m(x, \alpha_g, p, u_g, \vec{\mathbf{x}}_{nc}, u_f) \left[\frac{\partial}{\partial t} \left(V_{xg}(t, x) - V_{xf}(t, x) \right) + \right. \\ &\quad \left. + V_{xf}(t, x) \frac{\partial}{\partial x} V_{xg}(t, x) - V_{xg}(t, x) \frac{\partial}{\partial x} V_{xf}(t, x) \right], \end{aligned} \quad (2.146a)$$

$$\begin{aligned} \mathcal{L}_f^\rho(x, \alpha_g, p, u_f) \frac{\partial}{\partial t} V_{xf}(t, x) &= -\frac{1}{2} \mathcal{L}_f^\rho(x, \alpha_g, p, u_f) \frac{\partial}{\partial x} \left[\left(V_{xf}(t, x) \right)^2 \right] - \\ &- \mathcal{L}_f^p(x, \alpha_g) \frac{\partial}{\partial x} p(t, x) - \mathcal{L}_f^\rho(x, \alpha_g, p, u_f) K_{wf}(x, \vec{\mathbf{x}}) V_{xf}(t, x) + \\ &+ \mathcal{L}_f^\rho(x, \alpha_g, p, u_f) b_x(t, x) + A_x(x) \Gamma_{fg}(t, x, \vec{\mathbf{x}}) \left[V_{xfI}(t, x, \vec{\mathbf{x}}) - V_{xf}(t, x) \right] - \\ &- \mathcal{L}_f^\rho(x, \alpha_g, p, u_f) K_{if}(x, \vec{\mathbf{x}}) \left[V_{xf}(t, x) - V_{xg}(t, x) \right] - \\ &- K_{vm}(x, \vec{\mathbf{x}}) \mathcal{L}_m(x, \alpha_g, p, u_g, \vec{\mathbf{x}}_{nc}, u_f) \left[\frac{\partial}{\partial t} \left(V_{xf}(t, x) - V_{xg}(t, x) \right) + \right. \\ &\quad \left. + V_{xg}(t, x) \frac{\partial}{\partial x} V_{xf}(t, x) - V_{xf}(t, x) \frac{\partial}{\partial x} V_{xg}(t, x) \right]. \end{aligned} \quad (2.146b)$$

The parameter coefficients are given by:

$$\mathcal{L}_g^\rho(x, \alpha_g, p, u_g, \vec{\mathbf{x}}_{nc}) = \alpha_g(t, x) \rho_g(p, u_g, \vec{\mathbf{x}}_{nc}) A_x(x), \quad (2.147a)$$

$$\mathcal{L}_f^\rho(x, \alpha_g, p, u_f) = \alpha_f(\alpha_g) \rho_f(p, u_f) A_x(x), \quad (2.147b)$$

$$\mathcal{L}_g^p(x, \alpha_g) = \alpha_g(t, x) A_x(x), \quad (2.147c)$$

$$\mathcal{L}_f^p(x, \alpha_g) = \alpha_f(\alpha_g) A_x(x), \quad (2.147d)$$

$$\mathcal{L}_m(x, \alpha_g, p, u_g, \vec{\chi}_{nc}, u_f) = \alpha_g(t, x) \alpha_f(\alpha_g) \rho_m(\alpha_g, p, u_g, \vec{\chi}_{nc}, u_f) A_x(x). \quad (2.147e)$$

The mixture density $\rho_m(\alpha_g, p, u_g, \vec{\chi}_{nc}, u_f)$ is given by:

$$\rho_m(\alpha_g, p, u_g, \vec{\chi}_{nc}, u_f) = \alpha_g(t, x) \rho_g(p, u_g, \vec{\chi}_{nc}) + \alpha_f(\alpha_g) \rho_f(p, u_f). \quad (2.148)$$

The nomenclature here was:

$$\begin{aligned} \mathcal{L}_g^\rho(x, \alpha_g, p, u_g, \vec{\chi}_{nc}), \mathcal{L}_f^\rho(x, \alpha_g, p, u_f) \\ = \text{Gas and liquid phases' linear mass densities, respectively. Units: kg/m.} \end{aligned}$$

$$\begin{aligned} \mathcal{L}_g^p(x, \alpha_g), \mathcal{L}_f^p(x, \alpha_g) \\ = \text{Gas and liquid phases' cross-sectional areas, respectively. Units: m}^2. \end{aligned}$$

$$\begin{aligned} \rho_m, \rho_m(\alpha_g, p, u_g, \vec{\chi}_{nc}, u_f) \\ = \text{Fluid mixture mass density. Units: kg/m}^3. \end{aligned}$$

$$\begin{aligned} \mathcal{L}_m(x, \alpha_g, p, u_g, \vec{\chi}_{nc}, u_f) \\ = \text{Reduced linear mixture mass density. Units: kg/m.} \end{aligned}$$

$$\begin{aligned} b_x(t, x) \\ = \text{Body force acceleration component projected onto the streamline (} x\text{-axis). Gravity, turbine resistance and pump thrust are the possible body forces. Units: m/s}^2. \end{aligned}$$

$$\begin{aligned} K_{wg}(x, \vec{\mathbf{x}}), K_{wf}(x, \vec{\mathbf{x}}) \\ = \text{Gas and liquid wall friction coefficients, respectively. These quantities are computed through wall friction models; see Ref. [52, Volume 1] for details. Units: s}^{-1}. \end{aligned}$$

$$\begin{aligned} V_{xgI}(t, x, \vec{\mathbf{x}}), V_{xfI}(t, x, \vec{\mathbf{x}}) \\ = \text{Bulk average velocities with which vapor and liquid phases are born during vaporization and condensation, respectively. These quantities are often assumed equal (then called "interface velocity"), and are computed through interface momentum transfer models; see Ref. [52, Volume 1] for details. Units: m/s.} \end{aligned}$$

$$\begin{aligned} K_{ig}(x, \vec{\mathbf{x}}), K_{if}(x, \vec{\mathbf{x}}) \\ = \text{Gas and liquid interphase friction coefficients, respectively. These quantities are computed through interphase friction models; see Ref. [52, Volume 1] for details. Units: s}^{-1}. \end{aligned}$$

$$\begin{aligned} K_{vm}(x, \vec{\mathbf{x}}) \\ = \text{Virtual mass coefficient. This quantity is computed through dynamic drag models; see Ref. [52, Volume 1]. It is often assumed constant. Dimensionless.} \end{aligned}$$

Lastly, the energy conservation equations are [52, Volume 1]:

$$\begin{aligned} \frac{\partial}{\partial t} u_{vg}(\alpha_g, p, u_g, \vec{\chi}_{nc}) &= -\frac{1}{A_x(x)} \frac{\partial}{\partial x} \left[V_{xg}(t, x) A_x(x) u_{vg}(\alpha_g, p, u_g, \vec{\chi}_{nc}) \right] - \\ &- p(t, x) \frac{\partial}{\partial t} \alpha_g(t, x) - \frac{p(t, x)}{A_x(x)} \frac{\partial}{\partial x} \dot{V}_g(x, \alpha_g, V_{xg}) + q_{wg}(t, x, \vec{\mathbf{x}}) + \dot{u}_{v,ex,g}(t, x, \vec{\mathbf{x}}) + \\ &+ \dot{h}_{gf}(t, x, \vec{\mathbf{x}}) + \Phi_g(t, x, \vec{\mathbf{x}}), \end{aligned} \quad (2.149a)$$

$$\begin{aligned} \frac{\partial}{\partial t} u_{vf}(\alpha_g, p, u_f) &= -\frac{1}{A_x(x)} \frac{\partial}{\partial x} \left[V_{xf}(t, x) A_x(x) u_{vf}(\alpha_g, p, u_f) \right] - \\ &- p(t, x) \frac{\partial}{\partial t} \alpha_f(\alpha_g) - \frac{p(t, x)}{A_x(x)} \frac{\partial}{\partial x} \dot{V}_f(x, \alpha_g, V_{xf}) + q_{wf}(t, x, \vec{\mathbf{x}}) + \dot{u}_{v,ex,f}(t, x, \vec{\mathbf{x}}) + \\ &+ \dot{h}_{fg}(t, x, \vec{\mathbf{x}}) + \Phi_f(t, x, \vec{\mathbf{x}}). \end{aligned} \quad (2.149b)$$

The spatial phasic thermal energy densities are given by:

$$u_{vg}(\alpha_g, p, u_g, \vec{\chi}_{nc}) = \alpha_g(t, x) \rho_g(p, u_g, \vec{\chi}_{nc}) u_g(t, x), \quad (2.150a)$$

$$u_{vf}(\alpha_g, p, u_f) = \alpha_f(\alpha_g) \rho_f(p, u_f) u_f(t, x). \quad (2.150b)$$

The volumetric flow rates are given by:

$$\dot{V}_g(x, \alpha_g, V_{xg}) = \alpha_g(t, x) V_{xg}(t, x) A_x(x), \quad (2.151a)$$

$$\dot{V}_f(x, \alpha_g, V_{xf}) = \alpha_f(\alpha_g) V_{xf}(t, x) A_x(x). \quad (2.151b)$$

The wall frictional energy dissipation functions are given by:

$$\Phi_g(t, x, \vec{\mathbf{x}}) = \alpha_g(t, x) \rho_g(p, u_g, \vec{\chi}_{nc}) K_{wg}(x, \vec{\mathbf{x}}) \left(V_{xg}(t, x) \right)^2 + \Phi_{g,tm}(t, x, \vec{\mathbf{x}}), \quad (2.152a)$$

$$\Phi_f(t, x, \vec{\mathbf{x}}) = \alpha_f(\alpha_g) \rho_f(p, u_f) K_{wf}(x, \vec{\mathbf{x}}) \left(V_{xf}(t, x) \right)^2 + \Phi_{f,tm}(t, x, \vec{\mathbf{x}}). \quad (2.152b)$$

The following nomenclature was used:

$$\begin{aligned} u_{vg}(\alpha_g, p, u_g, \vec{\chi}_{nc}), u_{vf}(\alpha_g, p, u_f) \\ = \text{Gas and liquid phases' spatial thermal energy densities, respectively. Units: kJ/m}^3. \end{aligned}$$

$$\begin{aligned} \dot{V}_g(x, \alpha_g, V_{xg}), \dot{V}_f(x, \alpha_g, V_{xf}) \\ = \text{Gas and liquid phases' volumetric flow rates in the } x\text{-direction, respectively. Units: m}^3/\text{s}. \end{aligned}$$

$$\begin{aligned} q_{wg}(t, x, \vec{\mathbf{x}}), q_{wf}(t, x, \vec{\mathbf{x}}) \\ = \text{Gas and liquid phases' bulk thermal energy addition rate densities via convection from the channel wall, respectively. These quantities are computed through heat transfer models, most of which tend to divide the cooling rates between the phases proportional to the void and liquid volume fractions; see Ref. [52, Volume 1] for details. Units: W/m}^3. \end{aligned}$$

$$\dot{u}_{v,ex,g}(t, x, \vec{x}), \dot{u}_{v,ex,f}(t, x, \vec{x})$$

= Bulk direct energy deposition rate densities into gas and liquid phases, respectively. These quantities are cross-sectional phasic area integrals of $\dot{u}_{v,ex}(t, \vec{r})$ from Eq. (2.135). They are discussed in more detail in subsection 2.1.5; in RELAP5-3D, they are modeled as constant fractions of the fission source terms within a solid that is convectively cooled by fluid channels. Units: W/m³.

$$\dot{h}_{fg}(t, x, \vec{x}), \dot{h}_{gf}(t, x, \vec{x})$$

= Enthalpy generation rate densities due to vaporization and condensation, respectively. These quantities are computed through interphase mass transfer (vaporization and condensation) models; see Ref. [52, Volume 1] for details. Units: W/m³.

$$\Phi_g(t, x, \vec{x}), \Phi_f(t, x, \vec{x})$$

= Wall frictional energy dissipation functions' contributions to the gas and liquid phases, respectively. These quantities account for the thermal energy generation rates due to wall friction and irreversible turbomachinery work, if present at the position of interest. Units: W/m³.

$$\Phi_{g,tm}(t, x, \vec{x}), \Phi_{f,tm}(t, x, \vec{x})$$

= Turbomachinery energy dissipation terms into the gas and liquid phases, respectively. Irreversible pumps and turbines, besides doing work on the fluid, also dissipate thermal energy. These quantities are calculated through pump and turbine models; see Ref. [52, Volume 1]. They are zero at the positions of interest that are not thrust by a pump or rotating a turbine. Units: W/m³.

Together, Eqs. (2.140), (2.142), (2.143), (2.145), (2.146) and (2.149), coupled to the 1D form of Eq. (2.139), describe the thermal hydraulic model used by systems-level codes. Their accuracy is primarily limited by the flow regime maps, which, in turn, limit the accuracy of the models.

A staggered, lumped volume-based approach (similar to finite volume) is usually used to discretize the systems-level equations. Prior to discretizing them, the constituent models have to be specified. An example of a simplified model set, together with additional simplifying assumptions, is given in section 3.2.

This concludes the summary of the systems-level thermal hydraulic equations. As stated previously, they are heavily model-dependent, often (depending on the model) do not benefit from mesh refinement, and so are limited to relatively coarse analysis. They also do not readily accept transverse mass transfer, although certain systems codes do approximate crossflow. This is the thermal hydraulic model most commonly coupled to full core neutron diffusion models; RELAP5-3D itself contains a basic nodal diffusion module suitable for certain LWR transients. Individual core subchannels, with more realistic assumptions than the ones used in systems-level simulation, must be modeled to resolve the thermal and power peaks. Subchannel codes are used for this type of analysis; they are briefly discussed in the following subsection.

2.1.4.4 Subchannel-level Simulation

Subchannel codes are primarily used for full core and single fuel assembly thermal hydraulic modeling. VIPRE-01 [53] and COBRA-IV [54–56] are two prominent examples of LWR subchannel codes. There exist many different versions of them, with varying underlying assumptions; below, the assumptions underlying VIPRE-01 MOD-02 [53] and COBRA-IV-I [54] are discussed.

The main simplifying assumptions for subchannel-level simulation are:

1. As in systems-level simulation, in a given channel, a single phase is assumed to be sufficiently well-mixed for its state to be adequately represented through bulk properties. Because subchannel codes model individual core subchannels (one per fuel element), this is a good assumption.
2. The flow is treated as an array of 1-dimensional flows coupled through crossflows. This is a coupled 1D, not a full 2D, flow model. This is a reasonable assumption for a subchannel model, because the flow in a reactor is primarily axial, but lateral (transverse) crossflow is also present.
3. Equal phase temperatures are assumed. This assumption would not work in a systems-level model, but works on the individual subchannel level.
4. To model momentum conservation, phases are modeled as having potentially different axial velocities, but lateral velocities must be the same for both phases. Fluid drag forces are represented by wall friction and form drag models. Turbulent momentum exchange is modeled, but in-fluid shear is neglected. Interphase surface tension and axial turbulent mixing are neglected. These are reasonable assumptions in the core with a sufficiently fine mesh and sufficiently accurate friction models. The models may be optimized for the specific fuel assembly rod bundle geometries, and therefore are generally sufficiently accurate for these assumptions. The assumptions may break down if a coarser mesh, or a different reactor system, are modeled.
5. Transverse body force (gravity) is neglected. This is a good assumption for vertical cores, which both PWRs and BWRs are. This assumption is less accurate in horizontal reactors, such as CANada Deuterium Uranium (CANDU).
6. Flow is assumed to be subsonic. This assumption would not necessarily work in turbomachinery, which is modeled in systems-level simulation, but it is completely accurate in the core.
7. Fluid properties are evaluated as functions of specific enthalpy and a fixed, reference pressure. This assumption is reasonable because the pressure drop across the core is sufficiently small: approximately 171 kPa and 197 kPa across BWR and PWR cores, compared to the nominal core pressures of 7.14 MPa and 15.51 MPa, respectively [3, Appendix K]. Axial pressure variation (same for both phases) is still modeled, but it is not used to evaluate the fluid properties.
8. Noncondensables are not supported. Inside the core, in most transients, this is a good assumption.
9. Internal energy changes are assumed to only be due to convection from the fuel elements and structures, in-fluid conduction and turbulent mixing. This is a good assumption in the core, where these three mechanisms dominate all others.
10. As in systems codes, the solids are approximated as either 1-dimensional cylinders or plates with given effective heat transfer areas. In LWRs, which subchannel codes are optimized

for, this is an accurate approximation for all solids except possible the spacers, which have very small, if any, internal heat generation and heat transfer.

The subchannel assumptions are far too restrictive to be used for plant balance-wide thermal hydraulic modeling, because they are optimized for the core; systems-level assumptions must be used for plant balance simulation. Assuming an appropriate mesh and well-chosen friction and heat transfer models, the subchannel-level core models are generally more accurate than systems-level core models, but are also significantly more expensive.

There have been recent efforts to couple full core neutron transport and subchannel codes; one notable example is the Practical Numerical Reactor (PNR) package consisting of nTRACER neutron transport (method of characteristics) and MATRA subchannel codes [57]. Such simulation is still very costly (on the order of 2 h for a single steady state calculation on a 12-node Linux cluster with dual hexacore Xeon E5650 CPUs and 144 threads), and so, while now possible, it is primarily limited to steady state problems and slow transients, and is presently not yet fit for fast transient analysis. For this reason, thermal hydraulic systems and neutron diffusion code coupling is much more common.

The last class of approaches to thermal hydraulic modeling of nuclear reactors is the computational fluid dynamics (CFD) analysis, briefly discussed in the following subsection.

2.1.4.5 Computational Fluid Dynamics

CFD analyses encompass a broad class of methods which solve the fundamental fluid transport equations with little or no simplifying assumptions. A variety of models are still used, but these models generally only serve to coarsen the required mesh sizes, and do not encompass entire physical phenomena using correlations the way systems-level and subchannel models do.

CFD is the most expensive and the most accurate approach to thermal hydraulic analysis. As with subchannel codes, recently, CFD codes have been coupled to lattice neutron transport codes for steady state analyses. Numerical Nuclear Reactor (NNR), which consists of the coupling of DeCART (an MOC code) and STAR-CD (a CFD code), is a recent notable example [58]. As expected, the simulation is presently prohibitively expensive for practical transient analysis. MCNP5 (a Monte-Carlo neutron transport code) and STAR-CD have also recently been coupled [59]; such analysis is even more expensive, and, because it includes a Monte-Carlo code, is less prone to be extended to transient simulation than deterministic transport codes.

In summary, while systems, subchannel and CFD approaches have all been used in reactor analysis and coupled to neutron transport and diffusion codes, practical, fast transient simulation is typically performed using the thermal hydraulic systems and neutron diffusion or point kinetics approaches.

Regardless of the fluid transport model used, heat conduction in solids must also be modeled for a full thermal hydraulic model of a nuclear reactor. Several approaches to analyzing the heat transfer in fuel rods (the most common type of fuel element) are discussed in subsection 2.1.4.6.

2.1.4.6 Thermal Analysis of Fuel Rods

The two most common types of solid fuel elements are rectangular plates, used in many research reactors (i.e., MIT Nuclear Research Reactor), and cylindrical fuel rods. Annular fuel rods may also be used, but solid cylindrical fuel pins are used in almost all power reactors, and are focused on here.

Several assumptions apply to Eq. (2.139) when modeling fuel rods:

1. Azimuthal and axial dependence is dropped. As discussed in subsection 2.1.1.5, fuel rods consist of concentric fuel, gap and clad. The typical outer diameters of BWR and PWR clads are 11.20 mm and 9.5 mm, respectively. In comparison, typical active fuel heights are 3.588 m and 3.658 m, respectively [3, Appendix K]. Radial heat transfer therefore clearly dominates the axial and azimuthal heat transfer, and while axial power profile is not flat, the resulting axial temperature variation is always much smaller in a reactor at power than radial variation. For these reasons, this is an excellent ubiquitous approximation for fuel rods, with the exception of near the ends, where axial conduction may be significant. This approximation is made in both systems-level and subchannel-level simulation.
2. The fuel is assumed stationary. The properties are therefore not time-dependent. This is clearly a reasonable assumption for all accident types, with the exception of severe accidents in which fuel restructuring or melting can occur.
3. The fuel rod is assumed to be piecewise uniform: the clad, gap and fuel pellet are 3 possible concentric regions, and the fuel material may undergo restructuring, which separates it into multiple concentric zones (possibly with an annulus) [3, Chapter 8]. The values of the material properties themselves may vary due to their dependence on the temperature profile.

Under the above assumptions, the 1-dimensional cylindrical form of the heat diffusion equation (Eq. (2.139)) for a given radial region, like the fuel pellet, is [3, Chapter 8]:

$$\frac{\partial}{\partial t} u_v(t, r) = \frac{1}{r} \frac{\partial}{\partial r} \left(k(T) r \frac{\partial}{\partial r} T(u_v) \right) + \dot{u}_{v,ex}(t, r), \quad (2.153)$$

in which:

- r = Radial distance from pin centerline. Units: m.
- $u_v(t, r)$ = 1-dimensional (dependent on radius only) version of $u_v(t, \vec{r})$. Units: kJ/m³.
- $T(u_v)$ = Material temperature as a function of u_v . Compared to $T(t, \vec{r}, u_v)$, the time and position dependencies are dropped. Units: K.
- $k(T)$ = Thermal conductivity as a function of temperature. Compared to $k(t, \vec{r}, T)$, the time and position dependencies are dropped. Units: W/m K.
- $\dot{u}_{v,ex}(t, r)$ = 1-dimensional (dependent on radius only) version of $\dot{u}_{v,ex}(t, \vec{r})$. The in-fuel power profile is often assumed uniform, but except at the beginning of the fuel cycle, it actually peaks near the surface of the fuel pin; this is known as “the rim effect.” [60]. For this reason, radial dependence is retained here. Units: W/m³.

Equation (2.153) models the heat conduction in the solid parts of the fuel rod: the fuel pellet and the clad. The gap is modeled differently.

The gap between the fuel and the clad is a closed, thin (on the order of 0.09 mm [3, Appendix K]) annulus filled with a gas mixture, an inert gas like helium at the beginning of the cycle, and later in the cycle, a mixture of inert gas and fission gas products like xenon and kryp-

ton [3, Chapter 8]. Thermal convection and radiation are the two active heat transfer mechanisms across the gap; the effective heat flux is proportional to the difference between the pellet outer and clad inner temperatures. This proportionality constant is known as the “gap conductance,” and its value is determined by gap conductance models. Reference [3, Chapter 8] provides a summary of gap conductance models for use in LWRs.

Because the rate of conductive heat transfer between two points is also proportional to the difference between the temperatures at these two points, for simplicity, an effective thermal conductivity for the gap may be chosen, such that Eq. (2.153) applies in the gap as well. RELAP5-3D models occasionally make use of this approximation. Gap conductance models are generally more accurate, and are preferable for use.

Equation (2.153) is analytically solvable if $\dot{u}_{v,ex}(t, r)$ and $k(T)$ are spatially uniform, time-independent constants. Otherwise, it is not, generally, analytically solvable, and must be discretized. Once again, for thermal energy conservation, radial finite volume discretization will be used.

In a given neutron node $\Delta V_{i,j,k}$, we divide the radial profile into N_ι concentric annular shells and N_ς axial segments. Consider a thin cylindrical shell ι with inner and outer radii $r_{\iota-1}$ and r_ι , respectively. Assuming axial length Δz_ς , this shell’s volume and thickness are given by:

$$\Delta V_{\iota,\varsigma} = \pi (r_\iota^2 - r_{\iota-1}^2) \Delta z_\varsigma, \quad (2.154a)$$

$$\Delta r_\iota = r_\iota - r_{\iota-1}, \quad (2.154b)$$

where:

- ι = Radial shell index, increasing outward.
- ς = Axial segment index.
- N_ι = Number of concentric annular shells.
- N_ς = Number of axial segments in neutron node (i, j, k) ; normally, $N_\varsigma = 1$, and k is used to index both neutron nodes and adjacent fuel rod segments in the z -direction.
- $\Delta V_{\iota,\varsigma}$ = Volume of the cylindrical shell in the ι^{th} radial, ς^{th} axial segment. Units: m^3 .
- r_ι = Outer radius of the ι^{th} cylindrical shell. Units: m.
- Δr_ι = Radial thickness of cylindrical shell ι . Units: m.
- Δz_ς = Axial length of the ς^{th} axial segment. Units: m.

The cylindrical shell segment averages and totals are defined by:

$$\bar{f}_{\iota,\varsigma} = \frac{1}{\Delta V_{\iota,\varsigma}} \iiint_{\Delta V_{\iota,\varsigma}} dV f(\vec{r}), \quad (2.155)$$

$$F_{\iota,\varsigma} = \iiint_{\Delta V_{\iota,\varsigma}} dV f(\vec{r}), \quad (2.156)$$

in which:

- $\bar{f}_{\iota,\varsigma}$ = Spatial average of $f(\vec{r})$ over shell segment (ι, ς) .

$F_{\iota,\varsigma}$ = Total of the quantity of which $f(\vec{r})$ is the spatial density, in cell (ι, ς) .

For this discretization to be accurate, the shell needs to be sufficiently thin for the following approximations to work:

$$\ln\left(\frac{r_\iota - r_{\iota-1}}{r_\iota}\right) \approx \frac{\Delta r_\iota}{r_\iota}, \quad (2.157a)$$

$$\ln\left(\frac{r_{\iota+1} - r_\iota}{r_\iota}\right) \approx \frac{\Delta r_{\iota+1}}{r_\iota}. \quad (2.157b)$$

These approximations are equivalent to approximating the thin shells as flat plates of equal thickness; they are taken out of Ref. [61, Appendix D]. Assuming sufficiently thin cylindrical shells, and integrating Eq. (2.153) over $\Delta\mathcal{V}_{\iota,\varsigma}$ yields the discretized fuel heat diffusion equation:

$$\begin{aligned} \frac{d}{dt}U_{\iota,\varsigma}(t) = & \left[\frac{1}{2\pi r_{\iota-1} \Delta z_\varsigma} \left(\frac{\Delta r_{\iota-1}/2}{k(\bar{T}_{\iota-1,\varsigma})} + \frac{\Delta r_\iota/2}{k(\bar{T}_{\iota,\varsigma})} \right) \right]^{-1} (\bar{T}_{\iota-1,\varsigma} - \bar{T}_{\iota,\varsigma}) - \\ & - \left[\frac{1}{2\pi r_\iota \Delta z_\varsigma} \left(\frac{\Delta r_\iota/2}{k(\bar{T}_{\iota,\varsigma})} + \frac{\Delta r_{\iota+1}/2}{k(\bar{T}_{\iota+1,\varsigma})} \right) \right]^{-1} (\bar{T}_{\iota,\varsigma} - \bar{T}_{\iota+1,\varsigma}) + \dot{U}_{ex,\iota,\varsigma}(t). \end{aligned} \quad (2.158)$$

Here the following nomenclature was used:

$U_{\iota,\varsigma}(t)$ = Total thermal energy in cell (ι, ς) . This is the unknown in Eq. (2.158), related to $u_v(t, \vec{r})$ and $u_v(t, r)$ via Eq. (2.156). Units: kJ.

$\bar{T}_{\iota,\varsigma}$ = Average temperature in cell (ι, ς) , related to $U_{\iota,\varsigma}(t)$ via Eq. (2.159). Units: K.

$\dot{U}_{ex,\iota,\varsigma}(t)$ = Total rate of thermal energy production in cell (ι, ς) , related to $\dot{u}_{v,ex}(t, \vec{r})$ and $\dot{u}_{v,ex}(t, r)$ via Eq. (2.156). Units: W.

Average shell segment temperature is given by:

$$\bar{T}_{\iota,\varsigma} = T\left(\frac{U_{\iota,\varsigma}(t)}{\Delta\mathcal{V}_{\iota,\varsigma}}\right). \quad (2.159)$$

As will be discussed in subsection 2.1.5, fuel temperature dependence is often parametrized in terms of average fuel temperature. Besides the exact average fuel temperature, Ref. [62] outlines several different weighted averages that are used as parameters in temperature-dependent neutron properties. Assuming a known radial temperature profile $T(t, r)$ in a solid pellet with radius r_f , these averages are:

1. The exact average fuel temperature, denoted here as BE1 (Best Estimate 1) temperature average:

$$\bar{T}_{BE1}(t) = \frac{\int_0^{r_f} dr r \cdot T(t, r)}{\int_0^{r_f} dr r}, \quad (2.160)$$

Using the FV discretization above, axial segment ς BE1 average temperature is given by:

$$\bar{T}_{BE1,\varsigma}(t) = \sum_{\iota=1}^{N_\iota} f_{\nu\iota} \bar{T}_{\iota,\varsigma}, \quad (2.161)$$

with

$$f_{\nu\iota} = \frac{r_\iota^2 - r_{\iota-1}^2}{r_f^2}. \quad (2.162)$$

The nomenclature here was:

r_f = Fuel pellet radius. Units: m.

$T(t, r)$ = Known temperature profile. Units: K.

$f_{\nu\iota}$ = Radial cell ι volume fraction, used for spatial average calculation. Dimensionless.

$\bar{T}_{BE1}(t)$ = Exact average fuel pellet temperature, also known as ‘‘BE1 average.’’ Units: K.

$\bar{T}_{BE1,\varsigma}(t)$ = Axial segment ς BE1 average fuel temperature. Units: K.

2. The NEA effective fuel temperature, which is a weighted sum of only the pellet surface and centerline temperature [63]:

$$\bar{T}_{NEA}(t) = 0.7 T_{cl}(t) + 0.3 T_s(t), \quad (2.163)$$

where

$\bar{T}_{NEA}(t)$ = NEA effective fuel temperature (i.e., ‘‘NEA average’’), which makes the average more sensitive to pellet surface than BE1 average is. Units: K.

$T_{cl}(t)$ = Fuel centerline temperature. Units: K.

$T_s(t)$ = Fuel pellet surface temperature. Units: K.

3. The Goltsev-Davidenko-Tsibulsky-Lekomtsev effective Doppler temperature, which is a weighted integral [64]:

$$\bar{T}_{GDTL}(t) = \frac{\int_0^{r_f} dr \sqrt{T(t, r)}}{\int_0^{r_f} dr \frac{1}{T(t, r)}}, \quad (2.164)$$

where

$\bar{T}_{GDTL}(t)$ = Goltsev-Davidenko-Tsibulsky-Lekomtsev effective Doppler temperature, also known as ‘‘GDTL average.’’ Units: K.

4. A modified average fuel temperature, denoted here as BE2 (Best Estimate 2) average, which is a weighted sum of the exact average and pellet surface fuel temperatures [62]:

$$\bar{T}_{BE2}(t) = 0.92 \bar{T}_{BE1}(t) + 0.08 T_s(t), \quad (2.165)$$

where

$\bar{T}_{BE2}(t)$ = Modified average fuel temperature, also known as the BE2 average. Units: K.

The application of these exact and modified fuel temperature averages is discussed in subsection 2.1.5.

This concludes the summary of thermal analysis of fuel rods and of thermal hydraulic modeling of nuclear reactors. As discussed in the introduction to chapter 2, thermal hydraulics and neutron transport are a set of coupled physics, and must be modeled together. Subsection 2.1.5 summarizes the physics of this coupling; techniques used to address these coupled physics are discussed in section 2.2.

2.1.5 Thermal Hydraulics-Neutron Transport Coupling

Thermal hydraulics and neutron transport coupling is bilateral. The effect of thermal hydraulics on neutron transport is more complicated, and is discussed in subsection 2.1.5.1. The effect of neutron transport on thermal hydraulics is discussed in subsection 2.1.5.2.

2.1.5.1 Thermal Hydraulic Feedback Mechanisms

Thermal hydraulics affect neutron transport in several ways:

1. Coolant mass transfer affects the number densities in fluids, which affects the corresponding macroscopic cross sections (Eqs. (2.7) and (2.8)). Fluids primarily act as moderators (with the exception of solute absorbers, see below), and so while a change to a fluid density will not directly affect the absorption rates, it will affect the spectrum, which indirectly affects the absorption rates.
2. If a neutron absorber solute (like boron in PWRs) is present in the fluid, a change in its concentration will strongly affect the macroscopic properties. Assuming an evenly mixed, single-phase reactor coolant, this mechanism may be incorporated into the moderator density dependence.
3. All microscopic cross sections are heavily temperature-dependent, particularly the absorption cross sections [4, Chapter 2]. This dependence is due to the change in the apparent velocity of the neutron relative to a vibrating (via thermal motion) nucleus; this change in velocity is synonymous with a change in neutron energy E , which microscopic cross sections are dependent on. This is known as ‘‘Doppler broadening.’’ The absorption cross sections, which exhibit resonant behavior about certain specific neutron energies, are most strongly affected by a change in the nuclide’s temperature. Because the heavy nuclides in fuel tend to have more low-energy resonances than the lighter nuclides in the coolant, the temperature dependence of the fuel’s cross sections is more pronounced than the coolant’s.

4. Materials tend to expand with temperature change, which in certain reactor types may result in small changes in core geometry. Modifications in core geometry, particularly the distances between neighboring assemblies and control rod driveline expansion (CRDL), can have a strong effect on group neutron properties. This effect is particularly pronounced in sodium fast reactors (SFRs) [65, Chapter 15].

As a result of these dependencies, an appropriate full core model will modify Eqs. (2.96) and (2.101) to introduce temperature, moderator density and solute spatial density dependencies into the homogenized nodal properties. A common way of doing so involves running a number of lattice calculations with varying temperature and density profiles to then build a table of nodal group properties and interpolate between them during full core transient calculations.

As shown in subsections 2.1.1.5, 2.1.2.2, 2.1.4.3 and 2.1.4.4, full core neutron and thermal hydraulic models use different spatial meshes: neutron diffusion normally divides the core into a combination of homogenized, fuel assembly-sized nodes, while thermal hydraulic models tend to model the core as combinations of individual subchannels (subchannel codes), or lumped groups of subchannels (systems codes). The above dependencies of macroscopic cross sections are therefore parametrized for every individual node, with a given node's homogenized group properties parametrized in terms of average fuel temperature, moderator mixture density (or temperature) and solute spatial concentration(s) of this node.

As shown in Ref. [60], the power profile peaks at a fuel pellet's surface, therefore the rate of fission product production and absorption is higher at the surface. For this reason, the nodal group properties are more sensitive to the temperature variation at the surface of the pellet than at the centerline. The fuel pin averages discussed in subsection 2.1.4.6 are designed to emphasize this sensitivity, in order to more accurately parametrize the nodal properties in terms of fuel average temperature. Regardless of the type of weighted average chosen, it is important to be consistent when parametrizing neutron properties: when the property table is built, the averages used to build it must be of the same type used during the full core simulation. The GDTL and BE2 averages are generally recommended over the BE1 and NEA ones.

The thermohydraulic state of a given neutron node (i, j, k) is therefore a set of averages in terms of which the neutron properties are parametrized. Let such node's overall and fluid volumes be $\Delta V_{i,j,k}$ and $\Delta V_{i,j,k}^w$, respectively. For light water reactors, the most common parameters are the moderator mixture density (or, sometimes, void fraction), solute boron spatial density, and average fuel temperature:

$$\vec{\mathbf{x}}_{i,j,k}^{th}(t) = \left\{ \bar{\rho}_{m,i,j,k}^w(t), \bar{\rho}_{i,j,k}^s(t), \bar{T}_{i,j,k}^f(t) \right\}, \quad (2.166)$$

with:

$\vec{\mathbf{x}}_{i,j,k}^{th}, \vec{\mathbf{x}}_{i,j,k}^{th}(t)$ = Thermohydraulic state vector of neutron node (i, j, k) .

$\Delta V_{i,j,k}^w$ = Fluid volume in neutron node (i, j, k) . Units: m^3 .

$\bar{\rho}_{m,i,j,k}^w, \bar{\rho}_{m,i,j,k}^w(t)$ = Average moderator mixture density in neutron node (i, j, k) . This quantity is computed by applying Eq. (2.62) to ρ_m over volume $\Delta V_{i,j,k}^w$. Units: kg/m^3 .

$\bar{\rho}_{i,j,k}^s, \bar{\rho}_{i,j,k}^s(t)$ = Average solute s spatial density in neutron node (i, j, k) . This quantity is computed by applying Eq. (2.62) to ρ^s over volume $\Delta V_{i,j,k}^w$. Units: kg/m^3 .

$\overline{T}_{i,j,k}^f, \overline{T}_{i,j,k}^f(t)$ = Average fuel temperature in neutron node (i, j, k) . This quantity is evaluated by using one of Eqs. (2.160), (2.163), (2.164) or (2.165) to every axial pin segment in node (i, j, k) and averaging (possibly, weighted) the resulting averages. If a systems code is used, often a single “effective fuel pin” will be modeled in a given node (with the effective heat transfer area and segment length scaled by the number of pins in the node), so the interpin averaging becomes unnecessary. Units: K.

With the above considerations, a full core neutron diffusion model’s group properties then become parametrized in terms of nodal thermal hydraulic states $\vec{\mathbf{x}}_{i,j,k}^{th}(t)$:

$$\overline{D}_{eg,i,j,k}(t) \Rightarrow \overline{D}_{eg,i,j,k}\left(t, \vec{\mathbf{x}}_{i,j,k}^{th}\right), \quad (2.167a)$$

$$\overline{\Sigma}_{tg,i,j,k}(t) \Rightarrow \overline{\Sigma}_{tg,i,j,k}\left(t, \vec{\mathbf{x}}_{i,j,k}^{th}\right), \quad (2.167b)$$

$$\overline{\Sigma}_{sgg',i,j,k}(t) \Rightarrow \overline{\Sigma}_{sgg',i,j,k}\left(t, \vec{\mathbf{x}}_{i,j,k}^{th}\right), \quad (2.167c)$$

$$\overline{\chi\nu_p\Sigma}_{fgg',i,j,k}^{jf}(t) \Rightarrow \overline{\chi\nu_p\Sigma}_{fgg',i,j,k}^{jf}\left(t, \vec{\mathbf{x}}_{i,j,k}^{th}\right), \quad (2.167d)$$

$$\overline{\nu_{d,m}\Sigma}_{fg',i,j,k}^{jf}(t) \Rightarrow \overline{\nu_{d,m}\Sigma}_{fg',i,j,k}^{jf}\left(t, \vec{\mathbf{x}}_{i,j,k}^{th}\right). \quad (2.167e)$$

where:

$$\overline{D}_{eg,i,j,k}\left(t, \vec{\mathbf{x}}_{i,j,k}^{th}\right), \overline{\Sigma}_{tg,i,j,k}\left(t, \vec{\mathbf{x}}_{i,j,k}^{th}\right), \overline{\Sigma}_{sgg',i,j,k}\left(t, \vec{\mathbf{x}}_{i,j,k}^{th}\right), \overline{\chi\nu_p\Sigma}_{fgg',i,j,k}^{jf}\left(t, \vec{\mathbf{x}}_{i,j,k}^{th}\right),$$

$$\overline{\nu_{d,m}\Sigma}_{fg',i,j,k}^{jf}\left(t, \vec{\mathbf{x}}_{i,j,k}^{th}\right)$$

= Thermohydraulic state-dependent versions of the corresponding group properties, typically obtained through perturbations of the nominal steady state. The time dependence here refers to explicit geometry modification, such as control assembly movement.

2.1.5.2 Energy Deposition Mechanisms

Neutron transport affects two related phenomena in thermal hydraulics: fission heat rate and direct energy deposition. They are discussed separately below.

Fission heat rate in the fuel is the rate of heat generation produced by fission reactions. The fission heat rate density is given by:

$$\dot{u}_{v,ex}(t, \vec{\mathbf{r}}) = (1 - \gamma - \gamma_s) \sum_{j_f=1}^{J_f} \int_0^\infty dE' \kappa^{j_f}(E') \Sigma_{j_f}^{j_f}(t, \vec{\mathbf{r}}, E') \phi(t, \vec{\mathbf{r}}, E'), \quad (2.168)$$

in which:

$\kappa^{j_f}(E')$ = Energy generated per fission of nuclide j_f by a neutron of energy E' . As discussed below, this quantity may also account for heating due to neutron capture-generated γ and α particles. Units: MeV.

- γ = Coolant direct energy deposition factor, discussed below. Dimensionless.
 γ_s = Structure direct energy deposition factor, discussed below. Dimensionless.

Equation (2.168) is exact, and applies within a fuel pellet. The product $\kappa^{jf}(E') \Sigma_f^{jf}(t, \vec{r}, E')$ may be viewed as a “heat production cross section”, which makes the fission heat rate density a scalar reaction rate density. We can therefore apply Eq. (2.27c) to the sum in Eq. (2.168) to get:

$$\kappa \Sigma_{fg'}(t, \vec{r}) \phi_{g'}(t, \vec{r}) = \sum_{jf=1}^{J_f} \int_{E_{g'}}^{E_{g'-1}} dE' \kappa^{jf}(E') \Sigma_f^{jf}(t, \vec{r}, E') \phi(t, \vec{r}, E'), \quad (2.169)$$

which yields the multigroup version of Eq. (2.168):

$$\dot{u}_{v,ex}(t, \vec{r}) = (1 - \gamma) \sum_{g'=1}^G \kappa \Sigma_{fg'}(t, \vec{r}) \phi_{g'}(t, \vec{r}). \quad (2.170)$$

To resolve the fission heat rate at an individual pin level using Eq. (2.170), the reconstructed (heterogeneous) flux is required, as discussed in subsection 2.1.2.3. A nodal heat production cross section $\overline{\kappa \Sigma}_{fg',i,j,k}(t)$ may also be constructed by applying Eq. (2.62) to Eq. (2.169). The total and fission nodal heat rates may therefore be computed directly from the nodal average fluxes:

$$P_{i,j,k}(t) = \Delta V_{i,j,k} \sum_{g'=1}^G \overline{\kappa \Sigma}_{fg',i,j,k}(t) \overline{\phi}_{g,i,j,k}(t), \quad (2.171)$$

$$P_{f,i,j,k}(t) = (1 - \gamma - \gamma_s) P_{i,j,k}(t). \quad (2.172)$$

Here the nomenclature was:

- $\kappa \Sigma_{fg'}(t, \vec{r})$ = Group g' macroscopic heat production cross section. Units: MeV cm⁻¹.
 $P_{i,j,k}(t)$ = Total heat rate in node (i, j, k) . Units: W.
 $P_{f,i,j,k}(t)$ = Total fission heat rate in node (i, j, k) . Units: W.
 $\overline{\kappa \Sigma}_{fg',i,j,k}(t)$ = Heat production cross section of node (i, j, k) , obtained by applying Eq. (2.62) to Eq. (2.169). Units: MeV cm⁻¹.

Lastly, we recognize that as discussed above, the group heat production cross section is also a function of the thermohydraulic state of the node it is in:

$$\overline{\kappa \Sigma}_{fg',i,j,k}(t) \Rightarrow \overline{\kappa \Sigma}_{fg',i,j,k}(t, \vec{x}_{i,j,k}^{th}), \quad (2.173)$$

in which:

- $\overline{\kappa \Sigma}_{fg',i,j,k}(t, \vec{x}_{i,j,k}^{th})$
 = Thermohydraulic state-dependent version of $\overline{\kappa \Sigma}_{fg',i,j,k}(t)$. Units: MeV cm⁻¹.

Direct energy deposition occurs when neutrons are slowed down by the moderator or by the structural materials, and when γ and α particles, released during fission reactions or neutron captures, are absorbed by the moderator or by the structural materials. The moderator is normally the coolant; the neutron capture-induced γ and α heating is usually incorporated into an equivalent energy per fission κ [66], which is assumed here as well. To fully resolve this phenomenon, as discussed in the introduction to subsection 2.1.1, it would be necessary to solve the γ particle transport equation. This is unnecessary, because, due to their relatively long mean free paths in a reactor, γ particle absorption is far more uniform than the neutron flux profile across the core, and nearly perfectly uniform within a given neutron node. For this reason, it is normally assumed that a given fraction γ of the power generated in a node (or in a pin) is absorbed in the coolant flowing through the node (or through the pin's subchannel), and another given fraction γ_s is absorbed by the node's structures. The structures here, most importantly, include the clad. This yields the direct energy deposition rate densities in the neutron node (i, j, k) :

$$\dot{\bar{u}}_{ex,i,j,k}^w(t) = \frac{\gamma}{\Delta V_{i,j,k}^w} P_{i,j,k}(t), \quad (2.174)$$

$$\dot{\bar{u}}_{ex,i,j,k}^s(t) = \frac{\gamma_s}{\Delta V_{i,j,k}^s} P_{i,j,k}(t). \quad (2.175)$$

in which:

$\dot{\bar{u}}_{ex,i,j,k}^w(t)$ = Coolant direct energy deposition rate density in neutron node (i, j, k) . Units: W/m^3 .

$\dot{\bar{u}}_{ex,i,j,k}^s(t)$ = Structure direct energy deposition rate in neutron node (i, j, k) . Units: W/m^3 .

$\Delta V_{i,j,k}^s$ = Volume of the structure in neutron node (i, j, k) . Units: m^3 .

El-Wakil cites up to 1% and 4% for γ_s and γ , respectively [51]. Because the thermal resistance of the fuel pellet is significantly higher than the convective resistance between the clad and the coolant, or the internal thermal resistance of the clad, γ_s is frequently simply summed into γ . To be conservative, this combined γ is usually assumed to be about 3% [52, `typwr.i` deck].

The effect of a nonzero γ or γ_s is only sensible during a very fast transient, in which direct energy deposition slightly accelerates the fuel cooling and coolant heating processes.

Lastly, the effects of coupling on the neutron point kinetics equations are discussed separately, in the following subsection.

2.1.5.3 Coupled Neutron Point Kinetics

To convert the formal neutron point kinetics Eqs. (2.126) into a coupled (reactor power-based) form, we first define the conversion ratio between the flux amplitude function and the reactor power:

$$K_P(t) = \sum_{g=1}^G \iiint_{V_{tot}} dV \kappa \Sigma_{fg}(t, \vec{r}) S_{\phi g}(t, \vec{r}), \quad (2.176)$$

where:

$$P(t) = K_P(t) A(t), \quad (2.177)$$

in which:

$K_P(t)$ = Conversion factor between the flux amplitude function and the reactor power. Units: W/neutron.

$P(t)$ = Total reactor thermal power. Units: W.

As the other integral quantities in Eqs. (2.126), $K_P(t)$ is defined in terms of $S_{\phi g}(t, \vec{r})$, and therefore impractical. Two simplifying assumptions are made in subsection 2.1.3 to render the PKEs practical; one of them is assuming an approximately invariant flux shape function $S_{\phi g}(\vec{r})$. To render the power-based PKEs usable, two more assumptions are required:

3. $K_P(t)$ is assumed to vary sufficiently little during the transient of interest to be treated as a constant K_P . Assuming all of the feedback is encompassed in the thermohydraulic dependence of the reactivity (discussed below), this is an acceptable assumption, which yields:

$$P(t) \cong K_P A(t), \quad (2.178)$$

where:

K_P = Effective flux amplitude to reactor power conversion constant. Units: W/neutron.

4. Reactivity's thermohydraulic state dependencies are assumed to sufficiently capture the feedback. Properties used to define the integral parameters in both Eqs. (2.126) and Eqs. (2.128) are, potentially, thermohydraulic state-dependent, as discussed in subsection 2.1.5.1. However, reactivity is the most sensitive parameter, and so we ignore the other parameters' thermohydraulic state dependencies.

Under these two additional assumptions, the practical point kinetics Eqs. (2.128) can be modified. We multiply them by K_P and rearrange to obtain:

$$\frac{d}{dt} P(t) = \frac{\rho(t, \vec{x}^{th}) - \beta}{\Lambda} P(t) + \sum_{m=1}^M \lambda_m \tilde{C}_m(t) + \tilde{U}_{ex}(t), \quad (2.179a)$$

$$\frac{d}{dt} \tilde{C}_m(t) = \frac{\beta_m}{\Lambda} P(t) - \lambda_m \tilde{C}_m(t) \quad \forall m \in [1, \dots, M]. \quad (2.179b)$$

The derived quantities here are given by:

$$\tilde{C}_m(t) = K_P C_m(t), \quad (2.180)$$

$$\tilde{U}_{ex}(t) = K_P S_{ex}(t). \quad (2.181)$$

The modified nomenclature here is:

\vec{x}^{th} = Thermohydraulic state of the reactor: the set of solutions to the full core thermal hydraulic problem.

- $\rho(t, \vec{x}^{th})$ = Reactivity with feedback. There exist numerous ways to parametrize the dependence of the reactivity on the thermohydraulic state of the reactor; the simplest is a weighted sum of nodal averaged parameters for all nodes. The coefficients in such a sum are the reactivity feedback coefficients; their values characterize the stability of the reactor. Dimensionless.
- $\tilde{C}_m(t)$ = Precursor family m power. This is a derived quantity which replaced $C_m(t)$. Units: W.
- $\dot{\tilde{U}}_{ex}(t)$ = Rate of change of reactor power due to the external neutron source. This is a derived, known quantity. Units: W/s.

To couple Eqs. (2.179) to the local energy deposition rates, nodal and local section powers are expressed in terms of total reactor power:

$$P_{i,j,k}(t) = S_{i,j,k}(t) P(t), \quad (2.182a)$$

$$P_{f,i,j,k}^{\iota,\varsigma}(t) = S_{i,j,k}^{\iota,\varsigma}(t) P(t). \quad (2.182b)$$

The power fractions are given by (with $\Delta V_{\iota,\varsigma}$ that is within $\Delta V_{i,j,k}$):

$$S_{i,j,k}(t) = \frac{\sum_{\iota=1}^{N_\iota} \sum_{\varsigma=1}^{N_\varsigma} \sum_{g=1}^G \iiint_{\Delta V_{\iota,\varsigma}} dV \kappa \Sigma_{fg}(t, \vec{r}) S_{\phi g}(t, \vec{r})}{\sum_{g=1}^G \iiint_{V_{tot}} dV \kappa \Sigma_{fg}(t, \vec{r}) S_{\phi g}(t, \vec{r})}, \quad (2.183a)$$

$$S_{i,j,k}^{\iota,\varsigma}(t) = \left(1 - \gamma - \gamma_s\right) \frac{\sum_{g=1}^G \iiint_{\Delta V_{\iota,\varsigma}} dV \kappa \Sigma_{fg}(t, \vec{r}) S_{\phi g}(t, \vec{r})}{\sum_{g=1}^G \iiint_{V_{tot}} dV \kappa \Sigma_{fg}(t, \vec{r}) S_{\phi g}(t, \vec{r})}. \quad (2.183b)$$

The nomenclature here is:

$S_{i,j,k}(t)$ = Node (i, j, k) power fraction. Dimensionless.

$S_{i,j,k}^{\iota,\varsigma}(t)$ = Fuel shell segment (ι, ς) of node (i, j, k) fission heat rate fraction. Dimensionless.

From Eqs. (2.182), Eqs. (2.174) can be used to evaluate the direct energy deposition rate densities, if any.

One additional assumption is required for a usable coupling of Eqs. (2.179) to the local energy deposition rates:

5. The power fractions are assumed constant. This is a good assumption if PKE assumption 3 (constant K_P) holds, because in practice, the only way for K_P to stay nearly invariant is for the heat production cross sections to stay nearly invariant. If the heat production cross

sections stay constant, this assumption automatically holds. It yields:

$$P_{i,j,k}(t) \cong S_{i,j,k}P(t), \quad (2.184a)$$

$$P_{f,i,j,k}^{\iota,\varsigma}(t) \cong S_{i,j,k}^{\iota,\varsigma}P(t), \quad (2.184b)$$

with:

$S_{i,j,k}$ = Node (i, j, k) constant power fraction. Dimensionless.

$S_{i,j,k}^{\iota,\varsigma}$ = Fuel shell segment (ι, ς) of node (i, j, k) fission heat rate constant power fraction. Dimensionless.

Under these assumptions, Eqs. (2.179) can be readily coupled to a systems-level or a sub-channel code. The point kinetics kernel in RELAP5-3D, in particular, is very similar to the one described here [52, Volume 1].

This concludes the summary of thermal hydraulics-neutron transport coupling, and of the physics of interest for nuclear reactor transient analysis. Next, coupling methods are discussed.

2.2 Existing Multiphysics Simulation Methods

This section summarizes the two basic categories of time integration approaches: operator splitting and full coupling. The predominant modern ways of implementing the two approaches — physics-specific and general methods — are also discussed.

2.2.1 Operator Splitting and Full Coupling

Figure 2.5 illustrates the difference between the operator splitting and full coupling approaches to time integration, using the following notation:

t^n = Time at which the n^{th} time step is evaluated. Units: s.

$\vec{\mathbf{x}}_{\alpha}^n, \vec{\mathbf{x}}_{\beta}^n$ = Vector of unknowns (or “state vector” of different physics α and β at time step n , respectively. α and β here may correspond to neutron transport, precursor densities, thermal hydraulics, or a subset of thermal hydraulics like thermal conduction or pressure equation.

$\vec{\mathbf{x}}_{all}^n$ = Full state vector, which contains all sets of physics, at time step n . The simplest way to express this vector is to simply concatenate $\vec{\mathbf{x}}_{\alpha}^n$ and $\vec{\mathbf{x}}_{\beta}^n$, although in practice, it is sometimes beneficial to index the vectors such that the unknowns from a given spatial location are close together in the vector.

$\dot{\vec{\mathbf{x}}}_{\alpha}, \dot{\vec{\mathbf{x}}}_{\beta}$ = State derivative vectors of physics α and β , respectively. These are shorthands for $\frac{\partial}{\partial t} \vec{\mathbf{x}}_{\alpha}$ and $\frac{\partial}{\partial t} \vec{\mathbf{x}}_{\beta}$, respectively.

$\dot{\vec{\mathbf{x}}}_{all}$ = Full state derivative vector, a shorthand for $\frac{\partial}{\partial t} \vec{\mathbf{x}}_{all}$.

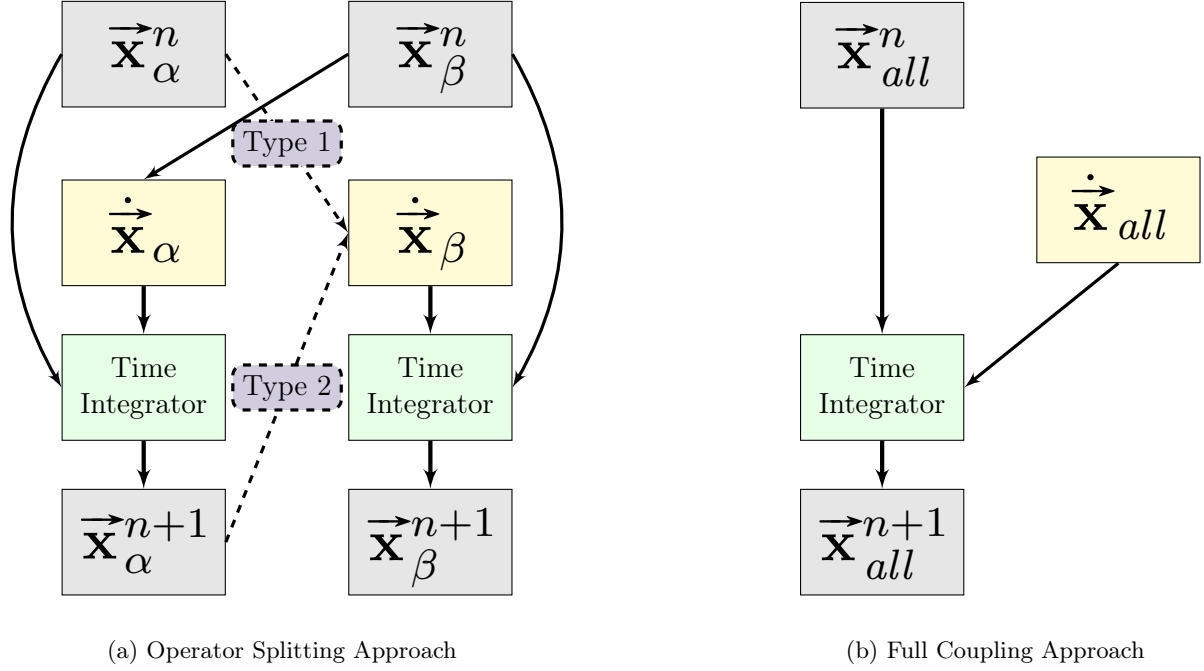


Figure 2.5: Time Integration Approaches

In the split operator approach, at every time step n , for every given physics, the state derivative function has to be reformulated at every time step, based on the other physics' states. In the thermal hydraulics and neutron transport coupled problem, this may mean that the state derivative vector of neutron transport (here, physics α) relies on having a vector of fuel temperatures and moderator and boron densities to evaluate the nodal multigroup properties. After the nodal multigroup properties are evaluated, the time integrator can simply integrate physics α from time t^n and state \vec{x}_α^n to time t^{n+1} and state \vec{x}_α^{n+1} .

After a given physics α has been integrated, there are two ways to proceed. In type 1 operator splitting (this is a nomenclature specific to this text), the new state \vec{x}_α^{n+1} is not used, and instead \vec{x}_α^n is used to formulate the other physics' state derivative vector functions. In type 2 operator splitting, \vec{x}_β is instead formulated using the newly available \vec{x}_α^{n+1} . Figure 2.5(a) illustrates both types of operator splitting.

In the other direction, if physics β corresponds to the thermal hydraulic model, operator splitting means that the thermal hydraulic state derivative vector requires the nodal fission heat rates prior to being able to integrate from state \vec{x}_β^n to \vec{x}_β^{n+1} . Neutron states n and $n+1$ are used to formulate \vec{x}_β in OS types 1 and 2, respectively.

To summarize, when OS is used, to integrate a given physics α from time t^n and state \vec{x}_α^n , a time integrator needs some part of the state vectors of all the other physics (here, β) at some time, here denoted \vec{x}_β^n , to formulate the state derivative vector function \vec{x}_α and integrate it to time step $n+1$. After \vec{x}_α^{n+1} is acquired, it may be supplied to \vec{x}_β , or \vec{x}_α^n may instead be used.

In the fully coupled approach, the physics are not treated separately. Instead, all physics are assembled together into a single state vector $\vec{\mathbf{x}}_{all}^n$, which is then integrated from time step n to time step $n + 1$ (state $\vec{\mathbf{x}}_{all}^{n+1}$). The state derivative vector function $\dot{\vec{\mathbf{x}}}_{all}$ itself is therefore unchanged; its values change when different time values and state vectors are supplied to it.

While the operator splitting approach involves more steps and is more complex, it is, in practice, much simpler to implement, because it allows the use of single-physics codes. For example, coupled systems-level thermal hydraulic and nodal diffusion modeling is routinely accomplished by coupling the systems-level code TRACE and nodal diffusion code PARCS [67]. Only limited, script-type development is required to accomplish this (although PARCS and TRACE are able to exchange data directly through memory, via specially developed modules). However, this approach has fundamental limitations, one of which is the difficulty of obtaining superlinear convergence in time for the time integrators used to integrate between the time steps [23].

For this reason, there has recently been a push to simulate the coupled physics using the fully coupled approach. There are two fundamentally different methods of implementing fully coupled simulation: by combining specific sets of physics in specialized codes, and by developing general frameworks for arbitrary sets of physics. The two approaches are explored in subsections 2.2.2 and 2.2.3, respectively.

2.2.2 Physics-specific Methods

Physics-specific codes can solve the coupled reactor multiphysics problem by either using a separate code for each given set of physics, or by implementing multiple sets of physics within a single code. Mathematically, the two approaches are generally equivalent, but, due to how information is managed in a machine's memory, the associated performance difference can be significant. A single code in which individual physics routines read data for the other physics from memory is expected to perform better than multiple codes that communicate data by writing their results to hard disk.

Subchannel thermal hydraulic codes generally do not have built-in neutron transport capabilities, and must therefore be coupled to external neutron codes. Practical Numerical Reactor (PNR), discussed earlier, is one significant example [57].

Systems codes, on the other hand, do sometimes have basic neutron capabilities added to them. For example, RELAP5-3D has a built-in point kinetics solver and a two-group multidimensional diffusion module, based on the NESTLE nodal diffusion code [52, 68]. Systems codes are also often coupled to separate neutron diffusion codes; examples include the already-mentioned PARCS and TRACE coupling, as well as the combination of SIMULATE-3K and RELAP5-3D [67, 69]. Both pairs of codes have linked executable versions to couple during operation; they may also be coupled through restart files.

Similarly, nodal diffusion codes often have basic built-in thermal hydraulic simulation capabilities. For example, SIMULATE-3K has a built-in 5-equation uncoupled subchannel (no crossflow) thermal hydraulic model [70, 71]. SIMULATE-5 has a more complex built-in thermal hydraulic model, which does allow for cross flow (but is still less robust than a full systems-level or sub-channel code's thermal hydraulic model) [37].

When separate codes are coupled via scripts (or even built-in coupling modules), the resulting approach is, necessarily, operator splitting, as the two codes cannot generally run simultaneous a single simultaneous time integrator on a combined state derivative vector. When multiple sets of physics are implemented within a single code, the situation is more complicated; potentially, the

code may treat the two physics as a single set of physics, and therefore implement a fully coupled simulation. However, because such codes have historically been developed with a focus on one of the sets of physics, and the other has been added as a basic replacement for a more complex coupling physics code (i.e., RELAP5-3D has historically been a systems code, with nodal diffusion being a relatively recent addition). Additionally, particularly the systems codes, often integrate even the different parts of the thermal hydraulic model separately, with the heat conduction and pressure equations being integrated after the mass conservation equations. It would therefore be impractical to treat the two physics as a single set of physics in such codes, without completely changing the physics-specific time integrators, and so such codes still integrate the physics separate, and exchange information at every time step, thereby constituting an operator splitting (usually type 2) approach. SIMULATE-3K and SIMULATE-5 are exceptions: they provide the option to iterate between the coupled physics at every time step, repeating the time step integration until the implicit time step is fully converged [72]. This option can have the same order of convergence in time as true fully coupled codes do, but, because multiple time integrations are required at every time step, it is generally less efficient.

An additional characteristic of physics-specific multiphysics codes arises from the fact that they are designed to solve a specific class of problems using a specific set of discretizations and time integration methods, and is therefore relatively difficult to modify. This makes them less than ideal for quickly trying an array of models, physical assumptions or discretization methods when attempting to model something the codes were not specifically developed for. Besides modifying the state derivative itself, the time integrators (which are normally physics-specific in such codes) would also have to be appropriately modified. This issue becomes even more complex if the code package is parallelized, because the code associated with modifying distributed state derivative vectors is significantly more complicated than modifying them locally.

As these examples indicate, while a fully coupled physics-specific code is in principle possible, the approach is limited by a lack of flexibility, and by the lack of truly fully coupled time integration methods in most existing codes. The latter arises from the fact that these codes are generally single-physics codes with added basic second physics capabilities.

For these reasons, another approach to implementing fully coupled multiphysics codes emerged: the development of codes based on general frameworks, which are designed to accommodate a wide range of physics in a standardized manner. The approach is discussed in the following subsection.

2.2.3 General Methods

In the terminology used in this text, “*multiphysics simulation frameworks*” (MSFs) are software packages and libraries which are not physics-specific, and allow the user to specify the physics involved, on top of the geometry, material properties, initial and boundary conditions, and other parameters which also have to be specified in the conventional physics-specific codes. After the physics, geometries and other conditions are specified, the multiphysics framework implements a discretization method, time integrator, associated with necessary nonlinear solvers, preconditioners, and any other required mathematical tools, as required, to solve and post-process the resulting system.

MSFs can differ by several parameters:

1. Range of supported physics. Many MSFs were intended to be used for a relatively narrow range of physics, and are therefore optimized for them.

2. Range of supported discretization methods. Certain discretization methods are necessarily physics-specific (e.g., spherical harmonics which do not apply to point kinetics or systems-level thermal hydraulics), but spatial discretization is the most ubiquitous one; MSFs often only support specific spatial discretization methods. Two important related characteristics of an MSF are:
 - a) Whether the MSF supports dissimilar meshes for a different physics.
 - b) Whether the MSF supports mesh adaptivity, and of what type.
3. Range of supported time integration and steady state solver methods. For fully coupled problems, time integrators may be supplied externally to the MSF, but most MSFs include some basic initial and static value problem capabilities.
4. Parallelization capability. Like physics-specific codes, MSFs may be serial-only, or support parallel communication, with or without distributed data.
5. Format in which information about the physics and the problem is supplied to the MSF.

Here, several existing MSFs are discussed.

SIERRA is an early example of a multiphysics simulation framework, intended for parallel, high-performance implementation of computational mechanics codes [73, 74]. It was developed to support finite element and finite volume spatial discretization methods, with dissimilar mesh support, and with several types spatial adaptivity. Being an early example of an MSF, SIERRA is very low level, and is essentially only a standardization of algebraic objects and solvers, as they pertain to discretized PDE solutions. Compared to later MSFs, SIERRA can be considered general and powerful, but difficult to use, and to have very limited automation of discretization and time integration procedures. It also was not intended for initially-discrete physics, such as electric circuit models.

MOOSE is a more recent example of a multiphysics simulation framework, developed, again, for high-performance, easy-to-use general finite element code development [75]. It only supports finite element spatial discretization, and has no dissimilar mesh support, but implements multiple spatial adaptivity algorithms. Compared to SIERRA, MOOSE is very easy to use, and has fully automated, arbitrary PDE discretization capabilities. Because of the greater degree of automation in its PDE discretization, it also relies on a specific class of nonlinear solvers for its static and initial value problem simulation: the Jacobian-Free Newton-Krylov (JFNK) solvers, reviewed in Ref. [23]. MOOSE's primary limitations are the restriction to finite element spatial discretization, lack of dissimilar mesh support, limited preconditioner and scaling capabilities, and a lack of support for nonspatial discretization. It is, nevertheless, one of the most actively used noncommercial MSFs currently available.

COMSOL is an example of a commercial multiphysics simulation framework [76]. While COMSOL does provide some rudimentary arbitrary physics capabilities, it is primarily reliant on a wide selection of physics-specific modules that are operated from a common core software package. It has recently been used for a neutron model of a reactor [77]. COMSOL is significantly slower and less scalable than SIERRA or MOOSE, its advantages being the powerful graphical user interface,

a very comprehensive array of tested modules, and a focus on the analyst, as opposed to the code developer.

All MSFs discussed above have significant limitations, primarily in physics and discretization flexibility and ease of use. Bond graphs is a fundamentally different formalism for an MSF, originally developed for manual (non-automated) use for instrumentation and control problems. The bond graph formalism is presented in detail in the following section.

2.3 Bond Graph Formalism

Historically, many different formalisms, or “modeling languages,” have been used to translate engineering systems into representations that can be mathematically modeled. Examples of these formalisms include electric circuit diagrams, kinematic schematics, operational block diagrams, and others. All of these formalisms have algorithmic procedures for formulating the mathematical models describing their corresponding systems. For example, the implementation of Kirchhoff’s laws or nodal voltage methods results in the mathematical model for electrical circuits, which is the basis of many codes for circuit analysis.

None of the conventional formalisms, such as the ones listed above, are fit for detailed dynamic modeling of mechatronic systems. For this reason, in the 1960s, a new formalism was developed by Henry Paynter. This formalism was named the “Bond Graph Method,” or simply “Bond Graphs” [1].

While it was originally only used for mechanical, electrical and hydroelectric systems, over time, the bond graph formalism grew into a complete research field, concerned with modeling mechanical, electrical, magnetic, hydraulic, thermal, agricultural, optical, financial and other systems [78, 79]. In 1979, Neff first applied bond graphs to a lumped parameter Loss-of-Fluid Test (LOFT) reactor dynamics model, using linearized point kinetics [80]. In the early 1980s, bond graphs were again applied by Tylee to linearized point kinetics models, and to model a PWR pressurizer [81–83]; the use of bond graphs for nuclear engineering has not been, until recently, further explored.

The research presented in this dissertation builds upon the work conducted as part of my S.M. thesis, in which bond graphs were first used to model 1D, one-group neutron diffusion coupled to heat diffusion, and a proof-of-concept bond graph processing code was developed [84]. This work is summarized in Ref. [85].

Bond graph formalism theory is summarized in some detail in this section; however, the interested reader is also directed to Refs. [86, 87], which provide a more thorough discussion of the approach, particularly with its application to initially-discrete systems. Note, that some nomenclature and notation used in this text differs from the ones in Refs. [86, 87] (which are also not always consistent with each other); where appropriate, this will be noted.

In this section, the theory of physical system modeling with bond graphs is summarized. The state of bond graph automation is also briefly discussed.

2.3.1 Bond Graph Representation Fundamentals

A bond graph representation of a physical system is an assembly of directed edges, called “*bonds*,” and graph elements, called “*bond graph elements*,” which represent the mathematical model of this system. More than one bond graph representation of a single mathematical model can exist; such

different representations may vary in performance, but their results must be exactly the same. The bond graph representation is also sometimes called the “*bond graph system*” (BGS).

The BGS can only represent a physical system in which all variables are discrete with respect to all independent variables except time. Many engineering systems, such as electrical circuits, are initially this way (“initially-discrete systems”); initially-continuous systems modeled by P(I)DEs have to be discretized prior to being represented with bond graphs. Numerous spatial, energy and angular discretization methods were discussed in section 2.1; for now, we may assume that the physical system of interest is already discrete.

BGSs consist of bond graph elements connected by bonds. The bonds carry bond variables (BVs) between the elements’ ports, and the elements’ constituent expressions impose the bond variables onto the bonds as functions of time, other bond variables, and state variables.

There exist multiple types of bonds, but for now, we only consider the “*full bond*.” With every full bond in a bond graph system are associated two distinct “*bond variables*”: an “*effort*” and a “*flow*”.

A bond always points from one element to another element. Some elements are multiport: they have multiple “*ports*” to which a bond can be connected. In certain elements, which specific port the bond is connected to makes no difference, but for most multiport elements, this matters.

In an “*augmented*” BGS a bond also has an associated causality, which determines which way the bond delivers its bond variables. A bond graph system without causalities assigned to its bonds is called “*acausal*.” Several topological properties are therefore associated with every full bond:

1. Element the bond points from.
2. Element the bond points to.
3. Port of the element the bond points from (if multiport).
4. Port of the element the bond points to (if multiport).
5. Bond causality: which way the effort is delivered — the flow is then delivered the opposite way.

Note, that causality is separate from bond direction. The notation and meaning of the possible direction and causality pairs on full bonds is given in Table 2.1. The “direction of positive flow” here refers to the fact that with a positive flow and effort, if “*true bond graphs*” are used, the product of the effort and the flow is the power transferred by the bond, and the direction of positive flow is the direction of positive power. From Table 2.1 we can see that effort is delivered in the direction that the causal stroke (the short mark at the end of the bond) is at; the direction of positive flow is the direction in which the bond half-arrow points.

Generally, causality is not inherently a part of the model, and is instead assigned to bonds in a BGS using an augmentation procedure. For many types of physical systems, this is the most difficult step of bond graph analysis; significant work has been done on augmentation procedure study and automation [88–91]. The most common and conventional augmentation procedure is the Sequential Causality Assignment Procedure (SCAP) [87, Chapter 5]. Sophisticated augmentation procedures are most useful when analyzing systems which are modeled by differential-algebraic equations (DAEs), instead of pure ODEs. This text focuses on large systems modeled by pure

Table 2.1: Full Bond Causality-Direction Configurations

Configuration	Causality	Direction of positive flow
$A \longrightarrow B$	Effort is delivered from A to B Flow is delivered from B to A	Positive flow is from A to B
$A \longleftarrow B$	Effort is delivered from B to A Flow is delivered from A to B	Positive flow is from A to B
$A \longleftarrow B$	Effort is delivered from B to A Flow is delivered from A to B	Positive flow is from B to A
$A \longrightarrow B$	Effort is delivered from A to B Flow is delivered from B to A	Positive flow is from B to A

ODEs (historically, bond graph models have been small), and so throughout the text, the use of only augmented bonds is implied.

For the purposes of this text, there exist three types of bonds: 1) full bonds, described above, 2) signal bonds, and 3) active bonds. The terms “*signal*” and “*active*” bonds are very often used interchangeably in literature [86, 87].

Multiport elements can have two different types of ports: 1) regular ports, and 2) signal ports. Full bonds can only be connected to regular ports; according to the bond’s causality, it delivers a bond variable to the port, and the element imposes the other bond variable onto the bond. Unlike full bonds, signal bonds only have one associated bond variable, which the bond delivers from a junction element to a signal port. Active bonds work similarly, but they instead are connected to regular ports; by using an active bond instead of a full bond, the element to whose regular port the active bond delivers the variable is prevented from imposing a value on the other variable of the bond. Signal and active bonds are therefore similar to the connections in operational block diagrams. Again, this differentiation between signal and active bonds is specific to this text.

In diagrams, both signal and active bonds are indicated by using full arrows, instead of half-arrows, with appropriate causalities.

Several bond graph elements, called “*storage elements*”, impose relations not only on bond variables, but on their time integrals: the (generalized) “*displacement*” and the (generalized) “*momentum*.” A displacement is a time integral of a flow, and a momentum is a time integral of an effort. The displacement and momentum variables are known as “*state variables*”, and for a given BGS, the vector of the displacements and momenta in the system is called the “*state vector*” of the system. The time derivative of the state vector is a “*state derivative vector*.”

The most important decision about modeling a physical domain with bond graphs is deciding what the effort and flow variables represent, and from this, what the displacements and momenta represent. In many domains, these choices have been long established, and are summarized in Table 2.2 [86].

From Table 2.2, it should be clear that bond graphs can be viewed as a formal generalization of the heat-electric current analogy, in which temperature plays the role of voltage, and heat transfer rate the role of current. Appropriate electric, or thermal, resistors are used to evaluate the heat/current as being proportional to the difference between temperatures/voltages.

For automated processing and for diagrams, bonds, elements, multiport element ports and

Table 2.2: Bond and State Variables in Different Physical Domains

Domain	Effort, Momentum	Flow, Displacement
Mechanical translational	Force Linear momentum	Velocity Linear displacement
Mechanical rotational	Torque Angular momentum	Angular velocity Angular displacement
Electric circuit	Voltage Flux linkage	Electric current Charge
Hydraulic	Pressure Pressure momentum	Volumetric flow rate Fluid volume
Thermal conduction	Temperature	Heat flow Thermal energy
Neutron diffusion ^a	Scalar flux	Reaction rate Number of neutrons

^a Coined in a precursor to this work, Ref. [84].

bond and state variables are indexed. The following shorthand expressions are used:

- “*Bond n* ”: bond with bond index n .
- “*Bond variable i* ”: bond variable with bond variable index i .
- “*Element j* ”: element with element index j .
- “*Element port k* ”: element port with port index k .
- “*State variable m* ”: state variable with state variable index m .

The following general nomenclature is used in this text:

- n = Bond index, aka bond ID.
- i = Bond variable index, aka BV ID.
- j = Element index, aka element ID.
- k = Element port index, aka port ID.
- m = State variable index, aka state ID.
- e, e_n = An effort (on bond n).
- f, f_n = A flow (on bond n).
- b, b_i = A bond variable (with BV ID i), either an effort or a flow, depending on context.
- p, p_j = A momentum (on inertial element j).
- q, q_j = A displacement (on capacitive element j).

- x, x_m = A state variable (with state ID m), either a momentum or a displacement, depending on context.
- b_{BGE} = Bond variable on the bond connected to element BGE.
- b_{BGE}^k = Bond variable on the bond connected to port k of element BGE.
- x_{XBGE} = State variable on the storage bond graph element XBGE.

For most physics in Table 2.2, the following holds:

$$P_n = e_n \cdot f_n, \quad (2.185)$$

in which:

P_n = Power transferred by bond n . Units: W.

For thermal conduction and neutron diffusion, Eq. (2.185) does not hold. It does not have to hold for the bond graph representation to be accurate; it was only implied to be true by an old convention. For this reason, when thermal bonds were first introduced, they were called “*pseudo-bonds*” [92–94]. There exists an alternative way to represent thermal conduction, and more generally, transient thermodynamics; it is to use temperature as the effort variable, and the entropy flow rate as the flow variable. This representation is used in many conventional bond graph texts, such as Ref. [86]. The temperature-entropy flow rate approach is particularly convenient for lumped thermodynamic analysis, in which it has been extended to account for both energy and mass conservation using “*convection bonds*” [95]. However, the thermodynamic representation is less convenient for treating discretized heat diffusion equation field problems, like the ones in reactor analysis. For this reason, in this work, the pseudo-bond graph approach shown in Table 2.2 is used; it is simpler and generally easier to work with.

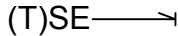
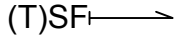
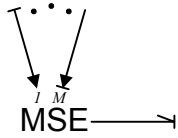
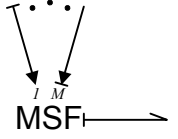
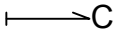
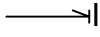
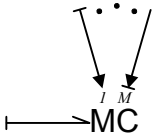
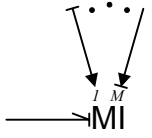
Table 2.3 summarizes the existing types of bond graph elements. The following nomenclature is used:

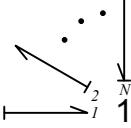
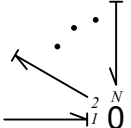
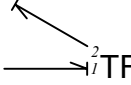
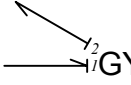

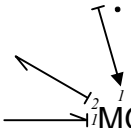
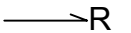
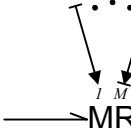
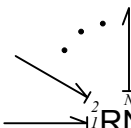
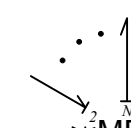
- $F(\dots), \vec{F}(\dots)$ = Scalar and vector functions of one or more variables, respectively.
- \vec{m} = A vector of modulating variables. Modulating variables usually include time and bond variables; the bond variables are delivered to the modulated element’s signal ports (indexed $1 \dots M$) via signal bonds. Some versions of the formalism also allow state variable modulation; state variable dependence is not shown explicitly on the diagram, and is instead specified directly through constituent expressions.
- C, I, μ, R = Corresponding single-port elements’ parameters.
- N = Number of ports on a multiport element.
- $\mathbf{Y}, \mathbf{Y}(\dots)$ = Constant matrix and matrix function, respectively. Here, they are used to define the constituent algebraic equations of the multiport resistive elements. If the causality on all full bonds connected to the multiport resistor is such that efforts are delivered to the element, and flows output, these matrices are called “*admittance matrices*.”

- $\vec{\mathbf{b}}_{in/out}$ = Input/output bond variable vectors for multiport resistors. They are configured as follows: on a multiport resistor, each individual port has an input and output bond variable (which is which depends on the port's bond's causality), therefore, if $e_{(M)RN}^k$ is an output, $f_{(M)RN}^k$ is an input, and vice versa.
- $\delta_{1/0}^k$ = Sign-setting delta function. $\delta_{1/0}^k = 1$ for bonds pointing to the junction, and 0 for bonds pointing from the junction.

Note, that the table includes all element types used in this thesis; most have been used in literature in some form, but some are new to the work.

Table 2.3: Bond Graph Elements

Element type	Constituent algebraic equation(s)	
Diagram	Name	
	(Time-dependent) Source of Effort	$e_{SE} = \text{constant}$ if SE, $e_{TSE} = F(t)$ if TSE
	(Time-dependent) Source of Flow	$f_{SF} = \text{constant}$ if SF, $f_{TSF} = F(t)$ if TSF
	Modulated Source of Effort	$e_{MSE} = F(\vec{\mathbf{m}})$
	Modulated Source of Flow	$f_{MSF} = F(\vec{\mathbf{m}})$
	Capacitive Element	$e_C = Cq_C$ or $e = F(q_C)$ and $\frac{\partial}{\partial t}q_C = f_C$
	Inertial Element	$f_I = Ip_I$ or $f = F(p_I)$ and $\frac{\partial}{\partial t}p_I = e_I$
	Modulated Capacitive Element	$e_{MC} = F(\vec{\mathbf{m}})q_{MC}$ or $e_{MC} = F(q_{MC}, \vec{\mathbf{m}})$ and $\frac{\partial}{\partial t}q_C = f_C$
	Modulated Inertial Element	$f_{MI} = F(\vec{\mathbf{m}})p_{MI}$ or $f_{MI} = F(p_{MI}, \vec{\mathbf{m}})$ and $\frac{\partial}{\partial t}p_I = e_I$

	1-junction	$f_1^1 = \dots = f_1^N$ and $\sum_{k=1}^N \delta_1^k e_1^k = 0$
	0-junction	$e_0^1 = \dots = e_0^N$ and $\sum_{k=1}^N \delta_0^k f_0^k = 0$
	Transformer	$e_{TF}^1 = \mu e_{TF}^2$ and $f_{TF}^2 = \mu f_{TF}^1$
	Gyrator	$e_{GY}^1 = \mu f_{GY}^2$ and $e_{GY}^2 = \mu f_{GY}^1$
	Modulated Transformer	$e_{MTF}^1 = F(\vec{m}) e_{MTF}^2$ and $f_{MTF}^2 = F(\vec{m}) f_{MTF}^1$
	Modulated Gyrator	$e_{MGY}^1 = F(\vec{m}) f_{MGY}^2$ and $e_{MGY}^2 = F(\vec{m}) f_{MGY}^1$
	Resistor	$e_R = R f_R$ or $e_R = F(f_R)$ or $f_R = F(e_R)$, depending on causality
	Modulated Resistor	$e_{MR} = F(\vec{m}) f_{MR}$ or $e_{MR} = F(f_{MR}, \vec{m})$ or $f_{MR} = F(e_{MR}, \vec{m})$, depending on causality
	Multiport Resistor	$\vec{b}_{out} = \mathbf{Y} \vec{b}_{in}$ or $\vec{b}_{out} = \vec{F}(\vec{b}_{in})$ with $\vec{b}_{out}, \vec{b}_{in} \in \mathbb{R}^N$, $\mathbf{Y} \in \mathbb{R}^{N \times N}$ and $\vec{F}(\dots) \in \mathbb{R}^N$
	Modulated Multiport Resistor	$\vec{b}_{out} = \mathbf{Y}(\vec{m}) \vec{b}_{in}$ or $\vec{b}_{out} = \vec{F}(\vec{b}_{in}, \vec{m})$ with $\vec{b}_{out}, \vec{b}_{in} \in \mathbb{R}^N$, $\mathbf{Y}(\dots) \in \mathbb{R}^{N \times N}$ and $\vec{F}(\dots) \in \mathbb{R}^N$

The numbers on the multiport elements' diagrams denote the elements' port IDs, including the signal (modulating) ports.

Source elements impose either a flow (SF, TSF, MSF) or an effort (SE, TSE, MSE) on the bond connected to them. They are primarily used to represent things like Dirichlet and Neumann

boundary conditions, external sources, or in circuits, grounds. If the bond points to the source element, it is sometimes instead referred to as a “*sink*” (of effort or flow).

Storage elements are the only elements that have variables associated with them, same way bonds do. The inertial and capacitive elements are the storage elements. These are the state variables, discussed above. Displacements are associated with capacitive elements, and momenta with inertial elements. A general capacitive element imposes an effort on its bond as a function of its displacement. A general inertial element imposes a flow on its bond as a function of its momentum. The number of differential state equations (the length of the state derivative vector) in a BGS corresponds to the number of storage elements. Note, that throughout the text, “*integral causality*” is assumed on all elements; for a capacitive element to be in integral causality, flow has to be delivered to it, for an inertial element to be in integral causality, effort has to be delivered to it.

Resistive elements control the rate of transfer and dissipation of the conserved quantity, by relating the bond variables delivered to them to each other. They do not have a preferred causality: as long as a resistive element’s causality is enforced by source and storage elements, the system will be fully causal. A single resistive element relates an effort to a flow (possibly using additional variables delivered via signal bonds); a multiport resistive element relates a group of efforts to a group of flows (possibly using additional variables delivered via signal bonds).

Junction elements are 1-junction, 0-junction, transformers and gyrators. In general, junction elements propagate and distribute the conserved quantity, without storing, generating or dissipating any of it, and without impeding its rate of transfer. Across a 1-junction, all flows are equal, and all efforts sum up to zero; the signs in the sum set by the bonds’ directions, which is the only time bond directionality comes into play. Similarly, across a 0-junction, all efforts are equal, and all flows sum up to zero. When applying the circuit representation to this, the 1-junction is often called the “series junction,” and the 0-junction the “parallel junction.” Exactly one bond may deliver flow to a 1-junction, and exactly one bond may deliver effort to a 0-junction.

The basic goal of a bond graph representation is assemble a bond graph system such that, under the bond variable convention chosen for the physics in question, the corresponding equations that the BGS represents would correspond to the physical system’s mathematical model. This generally requires discretizing the underlying P(I)DEs first.

The elements and bonds described in this subsection are the tools used for bond graph representation. Most existing bond graph literature focuses on representation techniques for various types of physical systems. Chapter 3 presents several new bond graph representation techniques for the physics described in section 2.1.

How the act of bond graph representation fits into the broader bond graph modeling procedure is described in the following subsection.

2.3.2 Bond Graph Process

Figure 2.6 illustrates the basic steps involved in fully coupled multiphysics code development using conventional and bond graph processes. An early iteration of the “bond graph process” was presented in Ref. [84], and is modified here. “AE” stands for “algebraic equation,” like the ones given in Table 2.3.

Conventional fully coupled single- or multiphysics codes for specific sets of physics are, generally, developed using the process in the bottom half of the figure. This process is not normally

presented this way, but numerous texts discuss it, usually without formally separating the steps, such as [96].

The conventional and bond graph processes share steps 1 and 2: it is necessary to discretize the P(I)DEs and order their unknowns before proceeding. All physical assumptions, discretizations, and data choices occur at step 1, in which the user decides which physics to include in the model, and which ones are left out. Transient continuous systems are usually described by systems of partial differential or integro-differential equations, which after discretization require appropriate indexing of the state vector. At the end of state 2, the governing system of equations is discretized and indexed. It should be noted that in step 1, the equations are discretized in all variables, except for time; the first two steps therefore constitute the “Method of Lines” [96].

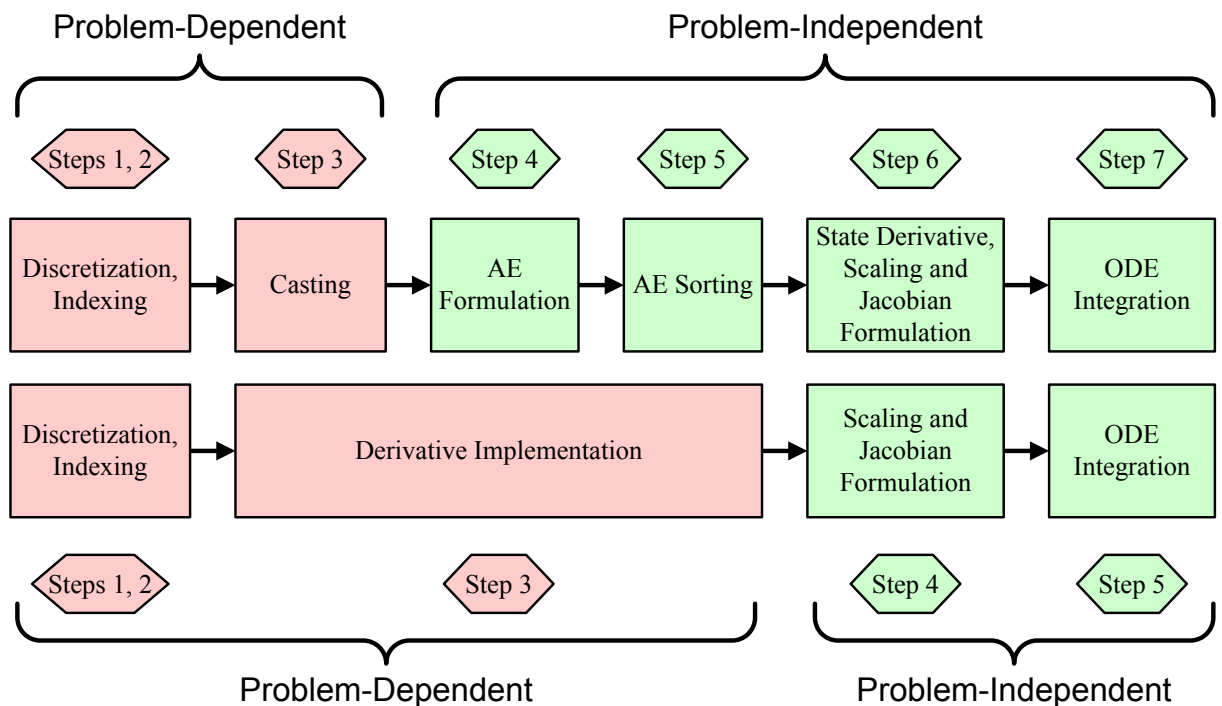


Figure 2.6: Conventional and Bond Graph Code Development Processes

The bond graph process replaces step 3 of the conventional process with the bond graph steps 3–5. The state derivative is assembled in step 6 of the bond graph process, and in step 3 of the conventional process. Otherwise, steps 4 and 5 of the conventional process are the same as steps 6 and 7 of the bond graph process.

Step 3 of the conventional process is, generally, the most complex and time-consuming step of code development. It needs to be fully or partially repeated every time the modeler wants to implement a new discretization, add new physics to the model, or parallelize the code. The step is clearly fully dependent on the problem. Step 4 is usually problem-independent, unless an analytic Jacobian is used. In most codes, Jacobian and scaling matrices, if used, are constructed using a finite difference approximation, and can therefore be build based only on the state derivative vector and Jacobian sparsity pattern, implemented in step 3.

Lastly, it is worth noting that steps 1 and 2 are not really part of the code, so they do not have

to be “implemented” very efficiently: the discretization must be adequate for the problem, but the code that implements steps 1 and 2 does not have to be optimized (with the exception of the meshing procedure). For this reason, the development of a code using the conventional process requires the developer to utilize both mathematical/physics skills, and software engineering skills, which can impede development. Steps 4 and 5 also have to be implemented very efficiently, but they are less dependent on the individual problem, and therefore step 4 can often be done efficiently once, and then reused.

The bond graph process replaces step 3 of the conventional process with its own 3 steps, of which only the first step (step 3) is problem-dependent. In the opinion of the author, the “casting” step (i.e., the bond graph representation, described in the previous subsection) process, since it deals only with specific, small elements of the problem, is a lot easier for an engineer to implement efficiently; ultimately, it involves only efficient implementation of the individual elements’ constituent equations, which are often linear, with constant coefficients. In step 3, the now-discrete dynamic system is represented with bond graphs, using the elements and bonds from subsection 2.3.1.

Steps 4 and 5 of the bond graph process, particularly step 5, are called the “*sorting steps*”, and do have to be implemented very efficiently. Like steps 6 and 7, they are problem-independent. For this reason, in the bond graph process, a larger section of the overall procedure can be done efficiently once, and then reused.

In step 4, the bond graph system is transformed into a large system of algebraic equations. In step 5, this algebraic system is solved, which, in step 6, yields a system of ordinary differential equations — the state derivative vector. The Jacobian, preconditioning, centering and scaling functions can also be assembled in step 6, depending on the time integrator’s requirements. In step 7, the ODE system is integrated, which constitutes the simulation of the physical dynamic system.

In addition, unlike in the conventional process, all problem-dependent steps of the bond graph process do not require efficient software engineering (although, as for the conventional process, efficient discretization is still, of course, necessary). This allows the developer to concentrate on developing an appropriate representation of the physics and their discretization; that is, to concentrate on the physics, and not on the code.

To summarize, steps 4–7 of the bond graph process can be automated by a “*bond graph processing code*.” In the next subsection, bond graph processing codes available prior to the beginning of this project are discussed.

2.3.3 Bond Graph Processing Codes

As stated above, bond graphs have historically primarily used for modeling initially discrete systems. For this reason, most available bond graph codes are not intended to deal with large problems; rather, they tend to concentrate on the causality assignment step, which can be very complicated for initially discrete systems. Continuous systems discretized through MOL, however, do not need complicated augmentation procedures, because they are known to be fully causal. However, codes capable of handling bond graphs which arise from such systems would need to be able to handle large numbers of unknowns (much larger than the few tens to a few hundreds typically encountered in a mechatronic system controls problem) and also strong nonlinearities in their elements’ equations. Unfortunately, most bond graph codes are primarily intended for

mechatronic and robotic system design, and therefore do not have the desired characteristics. Some examples include:

- 20-sim, a graphical dynamical system modeling application [97]. This application concentrates on letting the user enter the discrete model directly, by introducing system components such as controllers or measuring devices. It does use bond graphs internally, primarily for causality identification, but its focus on graphical system construction makes the code unfit for the application in this project.
- TUTSIM, Technical University of Twente SIMulation program [98]. TUTSIM is an ancestor of 20-sim, now commercialized but not maintained.
- ENPORT-6, one of the original bond graph processing codes; unfortunately, due to its age and original purpose, it is primarily limited to linear elements [99].
- CAMP-G, the Computer-Aided Modeling Program with Graphical Input [100]. This is essentially a preprocessor that takes a graphical bond graph representation and outputs MATLAB functions. This program is intended for mechatronics, and is highly limited in scale.
- `BGSolver` v1.01, developed as part of my S.M. thesis. The code relied on heavily on MATLAB's symbolic solving capabilities, and was not optimized, and so was only tested on small problems. The code was primarily developed as a proof-of-concept for the general bond graph processing automation procedure with highly nonlinear elements.

Three new bond graph processing codes, initially based on `BGSolver` v1.01, were developed as part of this dissertation; they, along with a more thorough discussion of bond graph automation theory, are described in chapter 4.

This concludes the summary of the bond graph formalism. As discussed previously, it has been, in a rudimentary fashion, used to model nuclear reactors. Section 2.4 discusses the bond graph representation techniques developed as part of this work.

2.4 Existing Bond Graph-Based Reactor Simulation Methods

In addition to my S.M. thesis, the only other existing applications of the bond graph formalism to reactor analysis were Neff's and Tylee's work in the early 1980s to model linearized PKEs with lumped thermal feedback [80–82]. The two representations are discussed in the following subsections.

2.4.1 Linearized Point Kinetics Models

One of the only available bond graph processing codes in the early 1980s was ENPORT-6, developed by Rosencode Associates, Inc. [99]. The code was intended for linear systems, and so the $\rho(t)P(t)$ product in Eq. (2.179) could not be represented and processed by ENPORT-6. Tylee linearized the problem by replacing, only in this term, $P(t)$ with the nominal power P^0 , which yielded the following power PKE:

$$\frac{d}{dt}P(t) = \frac{\rho(t, \vec{x}^{th})}{\Lambda}P^0 - \frac{\beta}{\Lambda}P(t) + \sum_{m=1}^M \lambda_m \tilde{C}_m(t). \quad (2.186)$$

This model is clearly very approximate: it may be sufficient for stability analysis (which is what it was used for in Refs. [81, 82]), but it cannot be used for reactivity-initiated accidents from zero-power cores, because in these, $P(t)$ can change by over 10 orders of magnitude.

In this work, true point kinetics equations are represented with bond graphs in section 3.1.

2.4.2 Coupled Neutron and Thermal Diffusion Models

In an earlier proof-of-concept work, I developed bond graph representation a method for 1D, one-group slab reactors, coupled with thermal diffusion [84, 85]. Such model, particularly in the absence of precursors, clearly cannot be adequate for almost any type of reactor analysis, and was only given as proof-of-concept. It was tested using a method of manufactured solutions (MMS) problem.

The representation was built by leveraging the significant similarities between neutron and heat diffusion equations. Section 3.3 develops a realistic bond graph representation for a multidimensional, multigroup spatial kinetics problem with multiple precursor families and thermal feedback.

This concludes the summary of the background required to understand this work. To summarize, Refs. [3, 6] are most likely sufficient for the general understanding of nuclear reactor physics required, and Refs. [86, 87] are sufficient for the general understanding of bond graph formalism.

In the next chapter, bond graph representations for realistic nuclear reactor physics are developed.

Chapter 3

Bond Graph Representation of Nuclear Reactor Physics

As was discussed in subsection 2.3.2, the first 3 steps of the bond graph process are P(I)DE discretization, indexing and bond graph representation of the resulting discretized ODEs. While bond graphs have been used for a wide range of physics (see section 2.3), until this project (prior to my S.M. thesis research [84]), the following was the state of the art of reactor multiphysics bond graph representation:

Neutron physics: Tylee developed representation methods for a linearized neutron point kinetics model with 6 precursor families [81, 82]. This representation is discussed in subsection 2.4.1. This model was linearized by replacing $P(t)$ in the $\rho(t, \vec{x}^{th}) P(t)$ product in Eq. (2.179a) with the nominal reactor power P^0 . In such model, the power cannot model appreciably from the nominal, therefore reactivity-initiated accidents (RIAs) with an initially cold core could not be modeled. No neutron transport or neutron diffusion representation methods existed.

Thermal hydraulic physics: One of the earliest applications of bond graphs were hydraulic power plant balance models [1]. Since then, a number of techniques have been developed for representing various thermofluid components with bond graphs; one of the most significant results is the invention of “*convection bonds*” by Brown [95]. Along with other existing bond graph representation techniques, convection bonds have primarily been developed for lumped, 0-dimensional parameter models of heat exchangers and chemical reactors. There do exist some studies in “distributed parameter system” (or “continuous system”) bond graph modeling; this terminology, effectively, usually refers to systems modeled by 1-dimensional PDEs [101]. A lumped parameter PWR pressurizer bond graph model has also been developed by Tylee [83], but no comprehensive systems- or subchannel-level thermal hydraulics bond graph representation techniques currently exist.

In my S.M. thesis, methods were developed for representing one-group 1D slab neutron diffusion using bond graphs; the approach was tested using a prototype bond graph processing code (BGSolver v1.01) [84]. The technique was published in Ref. [85]. This research acted as a proof of concept of the use of bond graphs for spatial kinetics reactor analysis, but both the resulting code and the representation techniques were significantly limited in the scale and type of the problems they could address.

In this chapter, the bond graph representation techniques for reactor multiphysics are significantly extended. Section 3.1 presents a method for representing a true neutron point kinetics equation system with thermal feedback, without the simplifications made in Refs. [81, 82]. A single-phase, systems-level thermal hydraulic model, which may be used in an approximate PWR core model to be coupled to a point kinetics model, is given in section 3.2. Section 3.3 contains the bond graph representation for a multidimensional, multigroup neutron diffusion model with precursors and thermal feedback, which can be used for a spatial kinetics reactor problem. Lastly, the P_n and S_n angular discretization representations are given in section 3.4.

3.1 Nonlinear Point Kinetics Models

Subsection 2.4.1 discussed the linearized point kinetics model representation; here, it is extended into a full nonlinear point kinetics model representation with distributed feedback.

Equations (2.179) are the neutron point kinetics equations, fit for coupling with thermal hydraulics. For the purposes of bond graph representation, it is convenient to rewrite them in terms of $\tilde{C}_m^*(t)$:

$$\tilde{C}_m^*(t) = \frac{\Lambda}{\beta_m} \tilde{C}_m(t). \quad (3.1)$$

Substituting Eq. (3.1) into Eqs. (2.179) yields the adjusted point kinetics equations, ready for bound graph representation:

$$\frac{d}{dt} P(t) = \frac{\rho(t, \vec{\mathbf{x}}^{th}) - \beta}{\Lambda} P(t) + \sum_{m=1}^M \frac{\beta_m \lambda_m}{\Lambda} \tilde{C}_m^*(t), \quad (3.2a)$$

$$\frac{d}{dt} \tilde{C}_m^*(t) = P(t) - \lambda_m \tilde{C}_m^*(t) \quad \forall m \in [1, \dots, M]. \quad (3.2b)$$

The following notation is used here:

$\tilde{C}_m^*(t)$ = Precursor family m energy. This is a derived quantity which replaces $\tilde{C}_m(t)$, which in turn replaced $C_m(t)$. Units: J.

Reactor reactivity $\rho(t, \vec{\mathbf{x}}^{th})$ accounts for both external reactivity sources, primarily due to control element movement, and for thermohydraulic feedback. Feedback in point kinetics models varies in complexity, but generally consists of $\rho(t, \vec{\mathbf{x}}^{th})$ being dependent on one or more of the following parameters:

- Fuel temperature (fuel Doppler feedback).
- Moderator temperature.
- Moderator density.
- Specific solute concentration, like boron.
- Structure temperature, structural motion (rarely modeled).

Regardless of the model of feedback, $\rho(t, \vec{\mathbf{x}}^{th})$ is given by:

$$\rho(t, \vec{\mathbf{x}}^{th}) = \rho_{ex}(t) - \rho_b + \rho_{fb}(t, \vec{\mathbf{x}}^{th}), \quad (3.3)$$

in which:

- $\rho_{ex}(t)$ = External reactivity (usually due to control rod/blade movement). This is a known function. Dimensionless.
- ρ_b = Bias reactivity. This is a constant, used to enforce a desired initial reactivity ρ^0 , which is zero for most transients. This quantity was not present in Tylee's model (subsection 2.4.1), which implicitly assumed $\rho_b = 0$. Such assumption is only reasonable if the initial steady state is known so well that $\rho_{fb}(t, \vec{\mathbf{x}}^{th})$ can be set to 0 at this steady state; this is rarely possible in practice, and so ρ_b is used to adjust the initial $\rho_{fb}(t, \vec{\mathbf{x}}^{th})$. Dimensionless.
- ρ^0 = Initial reactivity. $\rho^0 = \rho(0)$. Dimensionless.
- $\rho_{fb}(t, \vec{\mathbf{x}}^{th})$ = Reactivity feedback term. This term accounts for the change in reactivity due to the change in the thermohydraulic state of the system. Note, that the time dependence here is only present if the reactivity feedback coefficients or functions (discussed below) are time-dependent; explicit time dependence of the reactivity is modeled by $\rho_{ex}(t)$, and so in most models, $\rho_{fb}(t, \vec{\mathbf{x}}^{th})$ is state-dependent, but not time-dependent. Dimensionless.

The equations describing the reactivity model in this section are similar to the one used by RELAP5-3D v2.42 [52].

Reactivity feedback models can be either separable, or coupled; coupled feedback models are also referred to as "tabular" or "multidimensional" models in Ref. [52].

In a separable model, the change in feedback reactivity due to a change in a region's thermohydraulic state variable (e.g., temperature, density or boron concentration) is independent of all other regions' thermohydraulic states.

One can postulate a coupled feedback model in which the entire thermohydraulic state vector of the system affects the feedback reactivity, with each region's state's individual contributions being dependent on the thermohydraulic state of the system. Such model would be prohibitively expensive to implement, and so instead, coupled models are implemented by dividing the reactor into several compositions (typically, moderator, fuel and structure, possibly others). Weighted averages of each composition's thermohydraulic state variables are then constructed, and the coupled feedback model is written as a function of these weighted averages. The magnitude of the weighted averages' individual contributions is dependent on the thermohydraulic state of the system.

The separable reactivity feedback model is given by Eq. (3.4). The coupled feedback model, as described above, is given by Eq. (3.5).

$$\rho_{fb}(t, \vec{\mathbf{x}}^{th}) = \sum_{k=1}^{K_c} \sum_{r=1}^{N_k} \alpha_{T,r,k} T_{r,k}(t) + \sum_{k=1}^{K_c} \sum_{r=1}^{N_k} W_{T,r,k} R_{T,r,k}(T_{r,k}(t)), \quad (3.4)$$

$$\rho_{fb}(t, \vec{x}^{th}) = R \left(\sum_{r=1}^{N_1} W_{T,r,1} T_{r,1}(t), \dots, \sum_{r=1}^{N_{K_c}} W_{T,r,K_c} T_{r,K_c}(t) \right). \quad (3.5)$$

Both equations model thermal feedback only, but additional terms can be naturally added to account for other feedback thermohydraulic variables, as listed above. The following nomenclature is used here:

k	= Composition index. For the purposes of point kinetics thermohydraulic feedback, the reactor can be divided into K_c compositions, such as moderator, fuel or structure.
K_c	= Number of compositions in the model.
r	= Composition region index. Each composition is divided into multiple regions (e.g., hydraulic volumes in a systems-level thermohydraulic model), with N_k regions in a composition k .
N_k	= Number of regions in composition k .
$T_{r,k}(t)$	= Temperature of region r in composition k . Units: K.
$\alpha_{T,r,k}$	= Thermal feedback coefficient of region r in composition k . This quantity is used as part of the linear separable reactivity feedback model. Units: K^{-1} .
$R_{T,r,k}(T_{r,k}(t))$	= Nonlinear thermal feedback shape function of region r in composition k . This quantity is used as part of the nonlinear separable reactivity feedback model. It is normally set up to only control the shape of the feedback with respect to temperature, but not its magnitude. Units (assuming dimensionless $W_{T,r,k}$): K^{-1} .
$W_{T,r,k}$	= Nonlinear thermal feedback weighting factor. In the nonlinear, separable reactivity feedback model, this quantity scales the contribution of the region's nonlinear thermal feedback shape function $R_{T,r,k}(T_{r,k}(t))$. In the coupled reactivity feedback model, this quantity is used for the weighted average calculation for composition k , with the average then supplied to the K_c -dimensional coupled feedback reactivity feedback function $R(\dots)$. Dimensionless (assuming units of $R_{T,r,k}(T_{r,k}(t))$ to be K^{-1}).

$$R \left(\sum_{r=1}^{N_1} W_{T,r,1} T_{r,1}(t), \dots, \sum_{r=1}^{N_{K_c}} W_{T,r,K_c} T_{r,K_c}(t) \right) = K_c\text{-dimensional coupled reactivity feedback function. Dimensionless.}$$

The first and second terms in Eq. (3.4) represent the linear and nonlinear separable feedback models, respectively. Normally, one or the other, but not both, are used for a given composition and state variable type pair. Here, a general form of nonlinear separable feedback model is presented, where $R_{T,r,k}(T_{r,k}(t))$ may be different for every thermohydraulic region in the system, but in practice, only several simple feedback shapes are typically used. One common feedback function shape is $\sqrt{T_{r,k}(t)}$ [36, Problem 14].

For separable feedback, RELAP5-3D v2.42 utilizes a form of Eq. (3.4) that treats moderator

temperature linearly, moderator density nonlinearly (with the same nonlinear feedback shape function used for all thermohydraulic volumes), fuel temperature either linearly or nonlinearly, and boron concentration through control variables, which may be linear or nonlinear.

Tylee's bond graph representation of reactivity feedback was based on a complete lumped parameter representation, similar to Eq. (3.4) with $N_k = 1$ [81] and linear separable feedback.

To summarize, there are 3 possible point kinetics feedback representations: 1) linear separable, 2) nonlinear separable, and 3) coupled feedback. Bond graph representations for all 3 types of feedback are presented below.

The bond graph representation of a system of ODEs, like Eqs. (3.2), starts by matching the ODEs with appropriate bond graph element equations. As specified by Tylee, the PKEs are represented as follows [81]: total reactor power $P(t)$ is both a momentum and a flow, and the precursor energies $\tilde{C}_m^*(t)$ are displacements. The net rates of change of power and precursor energies, as well as individual contributions to them, are therefore efforts and flows, respectively. Reactivity is an effort, which is used to modulate a resistor; bias reactivity, which is a constant but unknown quantity (solved for during the steady state search, and kept constant throughout the transient), is a displacement and an effort. Again, Tylee does not use a bias reactivity [81].

The feedback equations derived above support an arbitrary number of compositions K_c ; below, thermal feedback only from the coolant w and the fuel f is represented. The resulting representations can be naturally extended to support more compositions.

Using the above conventions, we can proceed to construct the bond graph representations of the nonlinear neutron point kinetics equations.

Equation (3.2) is a rate balance equation, which in the bond graph context means a balance of efforts. This is enforced by a 1-junction. The rate of change of power is a rate of change of a momentum, which is the effort supplied to an inertial element. Reactor power is also an effort, so the first term in Eq. (3.2) can be viewed as a contribution to a rate of change of a momentum proportional to an effort; the proportionality coefficient is a combination of a function $(\rho(t, \vec{x}^{th})/\Lambda)$ and a constant (β/Λ) . These two components may be represented as an effort-modulated (reactivity is an effort) resistor, and a constant coefficient resistor, respectively. Lastly, the second term in Eq. (3.2) is a sum of individual contributions to a rate of change of momentum (power), which can be enforced by another 1-junction.

Combined, these equivalences result in the bond graph representation in Figure 3.1.

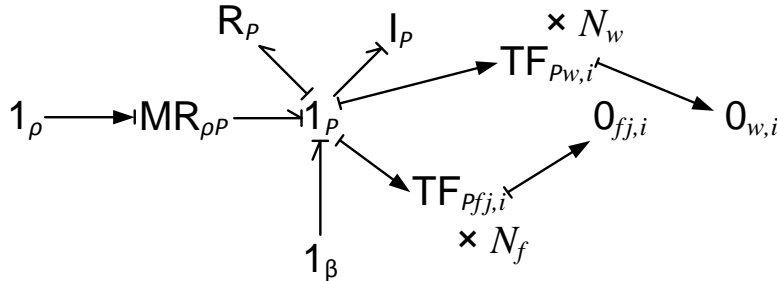


Figure 3.1: Power PKE Bond Graph Representation

The elements are denoted as follows:

- 1_ρ : Reactivity calculation 1-junction. The effort output by this junction is the reactor reactivity.
- 1_P : Power calculation 1-junction. The efforts supplied to this 1-junction add up to the rate of change of reactor power. The flows output from it are all equal to the power itself.
- 1_β : Precursor contribution calculation 1-junction. The efforts supplied to this 1-junction (defined below when representing precursor energies) add up to the total contribution to the rate of change of power from precursor decay. The flow supplied to it is the power itself.
- $MR_{\rho P}$: Reactivity-power relating modulated resistor. This resistor is modulated by the reactivity, and has the power supplied to it; it multiplies this power by $\rho_{fb}(t, \vec{x}^{th})/\Lambda$ to output the reactivity's contribution to the rate of change of power.
- R_P : DNF-power relating resistor. This is a constant coefficient resistor which the power is supplied to; it multiplies this power by β/Λ and outputs the DNF's contribution to the rate of change of power.
- I_P : Power inertial element. This is a storage element which implements the power time derivative; through it, power becomes an unknown.
- $0_{w,i}$: Coolant temperature 0-junction. The effort supplied to this 0-junction is the temperature of the coolant in coolant segment i ; it is discussed in more detail in section 3.2. There are N_w of these elements, one per axial coolant segment.
- $TF_{Pw,i}$: Direct coolant energy deposition transformer. This element distributes an appropriate portion of reactor power, via direct energy deposition, to the coolant segment i . There are N_w of these elements, one per axial coolant segment.
- $0_{fj,i}$: Fuel temperature 0-junction. The effort supplied to this 0-junction is the temperature of the fuel radial shell j in axial segment i ; it is discussed in more detail in section 3.2. There are N_f of these elements, one for every axial fuel shell segment.
- $TF_{Pfj,i}$: Fission heat rate transformer. This element distributes an appropriate portion of reactor power, via fission heat rate, to the fuel radial shell j in axial segment i . There are N_f of these elements, one for every axial fuel shell segment.

Note, that because here the x -direction is considered axial, and no spatial neutron nodes are present, a slightly different notation from subsection 2.1.4.6 is used for indices in this and the following sections:

- i = Axial segment index, for both fuel and coolant.
- j = Radial fuel shell segment, increasing outward.
- N_{fj} = Number of radial fuel shells in a given axial segment.
- N_i = Number of axial segments.
- N_f = Total number of fuel radial shells. $N_f = N_i \times N_{fj}$.
- N_w = Number of axial coolant segments. In a typical model, $N_i = N_w$.

The constituent expressions of the elements in Figure 3.1 are given by:

$$I_P = 1, \quad (3.6a)$$

$$R_P = \frac{\beta}{\Lambda}, \quad (3.6b)$$

$$R_{\rho P}(e_\rho) = e_\rho, \quad (3.6c)$$

in which:

I_P = The parameter of the I_P inertial element. Dimensionless.

R_P = The resistance of the constant power resistor R_P . Units: s^{-1} .

e_ρ = Reactivity effort, supplied as the modulating variable to $R_{\rho P}$. Dimensionless.

$R_{\rho P}(e_\rho)$ = The resistance of the reactivity-modulated $R_{\rho P}$ element. Dimensionless.

Figure 3.1 is very similar to a corresponding representation in Ref. [81], with the exception of the $MR_{\rho P}$ element, which is a new (modulated, and therefore nonlinear) addition, replacing a linear transformer. Tylee could not use modulated resistors, because they prevented the model from being fully linear, and therefore able to be processed by ENPORT-6 [99].

The precursor energy Eq. (3.2b) is represented the same way as in Ref. [81]; this representation is given in Figure 3.2. Note, that the 1_β and 1_P elements in it are the same elements as in Figure 3.1, through which the two physics are connected.

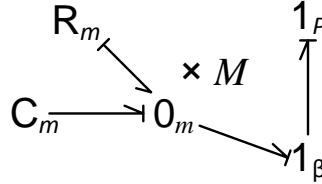


Figure 3.2: Precursor PKEs Bond Graph Representation

In the bond graph, there are M of each of the following elements in Figure 3.2:

0_m : Precursor family m energy calculation junction. The effort supplied to it is the precursor family m energy contribution to the rate of change of power. The flow it outputs is the rate of change of precursor family m energy.

C_m : Precursor family m energy capacitor. This is a storage element which implements the precursor family m energy time derivative; through it, the precursor family m energy becomes an unknown.

R_m : Precursor family m resistor. This is a constant coefficient resistor, chosen in such a way as to appropriately relate the rate of change of precursor family m energy and its contribution to the rate of change of power.

The capacitance and resistance here are chosen in such a way as to enforce the appropriate decay rate-based contributions into the 1_β -junction in Figure 3.1, while also representing Eq. (3.2b).

They are given by:

$$C_m = \frac{\lambda_m \beta_m}{\Lambda}, \quad (3.7a)$$

$$R_m = \frac{\beta_m}{\Lambda}, \quad (3.7b)$$

in which:

C_m = The capacitance of C_m . Units: s^{-2} .

R_m = The resistance of R_m . Units: s^{-1} .

Reactivity, which modulates $MR_{\rho P}$ in Figure 3.1, is given by Eq. (3.3); as stated above, it is an effort, algebraically computed by a 1-junction. The external reactivity contribution $\rho_{ex}(t)$ is only time-dependent, and therefore supplied by a time-modulated source of effort element. Bias reactivity is an unknown (but constant) effort supplied by a capacitor; the effort is equal to the capacitor's displacement. The feedback reactivity term is either a sum of efforts from individual contributions (separable feedback, Eq. (3.4)), or a single effort from a multidimensional function evaluation (coupled feedback, Eq. (3.5)). The separable linear contributions to feedback reactivity are a weighted sum of efforts, which can be accomplished by a single 1-junction and one constant coefficient transformer element per radial cell.

In a linear separable feedback model, all contributing efforts are multiplied by constant feedback coefficients. Because both the weighting volume fractions for the individual fuel temperatures and the feedback coefficients are constant multipliers, a single linear transformer element (one per contributing region) can represent both these coefficients. Active bonds must be used instead of full bonds, because reactivity does not directly contribute to the system: it only affects the rate of change of power through the $MR_{\rho P}$ resistor in Figure 3.1.

The bond graph representation of the reactivity with linear separable feedback is given in Figure 3.3. With the exception of the nonlinear resistor $MR_{\rho P}$ and the bias reactivity's capacitor $C_{\rho,b}$, this representation is identical to the one in Ref. [81].

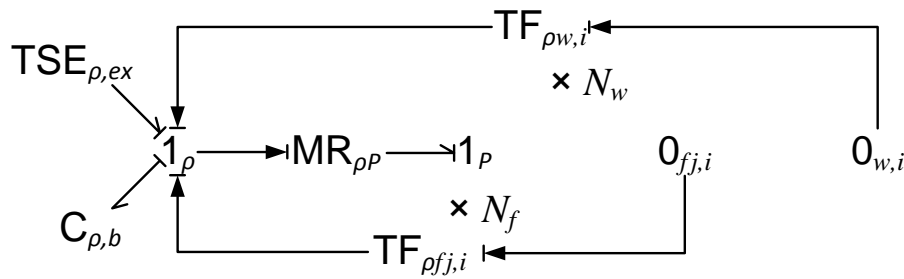


Figure 3.3: Reactivity with Linear Separable Feedback Bond Graph Representation

The elements here are:

$C_{\rho,b}$: Bias reactivity capacitor. A zero flow is supplied to it, which makes the bias reactivity it stores a constant, which it in turn supplies to 1_ρ , thereby enforcing the contribution of ρ_b on the total reactivity.

- $\text{TSE}_{\rho,ex}$: External reactivity time-modulated source of effort. This element outputs $\rho_{ex}(t)$.
- $\text{TF}_{\rho w,i}$: Coolant temperature reactivity feedback transformer. This transformer imposes the feedback effect of the temperature in coolant axial segment i on the reactivity. The bonds connected to it are active, because reactivity does not directly figure in the thermal hydraulic equations. There are N_w of these elements, one for every coolant axial segment.
- $\text{TF}_{\rho f j,i}$: Fuel temperature reactivity feedback transformer. This transformer imposes the feedback effect of the temperature in fuel radial shell j , axial segment i , on the reactivity. The bonds connected to it are active, because reactivity does not directly figure in the heat diffusion equation. There are N_f of these elements, one for every axial fuel shell segment.

The nonlinear separable feedback model almost always parametrizes feedback in terms of radial fuel average temperatures, as opposed to individual radial shell temperatures. Assuming the BE1 average is used (Eq. (2.160)), the average can be constructed as a single 1-junction per axial element, together with a single constant coefficient transformer per radial shell. Nonlinear 2-port resistive elements can then be used to calculate the nonlinear contributing efforts and supply them to the reactivity calculation junction 1_ρ . Such representation is presented in Figure 3.4.

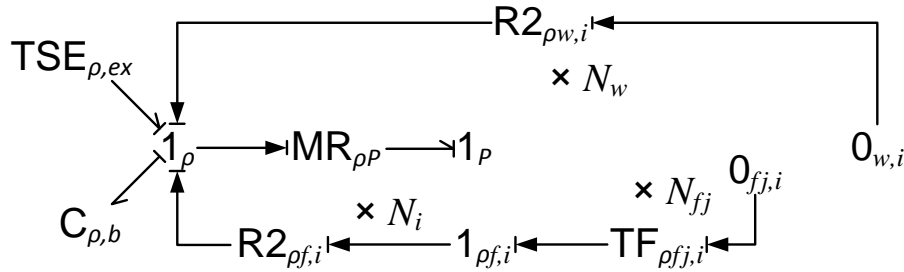


Figure 3.4: Reactivity with Nonlinear Separable Feedback Bond Graph Representation

The elements here are:

- $\text{TF}_{\rho f j,i}$: Fuel radial average-computing transformer. This transformer accepts, through an active bond, the temperature in fuel radial shell j of axial segment i , and outputs, through another active bond, its contribution to the segment average temperature depending on the fuel temperature averaging model used (see subsection 2.1.4.6). For every axial segment i , there are N_{fj} of these elements, one per radial shell.
- $1_{\rho f,i}$: Fuel radial average-computing 1-junction. This junction accepts the effort contributions from all of the $\text{TF}_{\rho f j,i}$ elements, and outputs the fuel radial temperature average in segment i . All bonds connected to this junction are active. There are a total of N_i of these elements, one per axial fuel segment.

- $R2_{\rho f, i}$: Fuel reactivity feedback-computing 2-port resistor. This nonlinear resistor accepts, through an active bond, the axial segment i fuel temperature average, and outputs its nonlinear separable reactivity feedback contribution, through another active bond. There are a total of N_i of these elements, one per axial fuel segment.
- $R2_{\rho w, i}$: Coolant reactivity feedback-computing 2-port resistor. This nonlinear resistor accepts, through an active bond, the axial segment i coolant temperature, and outputs its nonlinear separable reactivity feedback contribution, through another active bond. There are a total of N_w of these elements, one per axial coolant segment.

In a coupled feedback model, weighted averages (with constant weighting coefficients) for each composition (here, for coolant and fuel) are computed prior to evaluating the overall reactivity feedback. Such averages, as well as individual contributions to them, are efforts, summed by a 1-junction. As in the linear separable feedback model, the weighting is implemented through linear transformer elements, one per contributing region. These efforts (weighted averages) are then supplied to a multiport nonlinear resistor, which outputs the coupled reactivity feedback term — also an effort. Because in this example there are two inputs (fuel and coolant), this multiport resistor is an R3 element. As with the other reactivity feedback models, all bonds involved are active, because reactivity does not directly contribute to the thermal hydraulic or heat diffusion equations. The reactivity with coupled feedback representation is given in Figure 3.5.

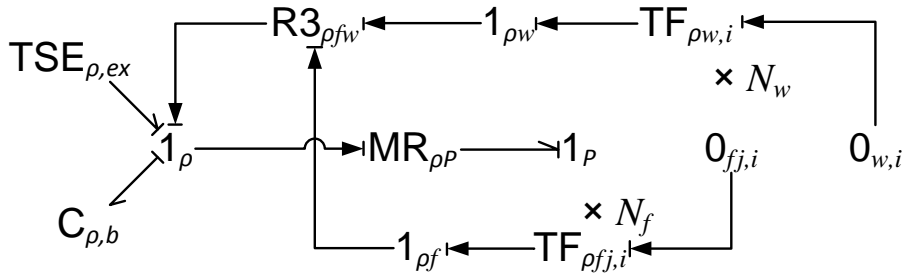


Figure 3.5: Reactivity with Coupled Feedback Bond Graph Representation

The elements here are:

- $TF_{\rho w, i}$: Coolant temperature weighting transformer. This element incorporates the contribution of the coolant axial segment i into the coolant's overall weighted average. There are N_w of these elements, one per coolant axial segment.
- $1_{\rho w}$: Coolant temperature weighted average 1-junction. This element sums the individual contributions from the coolant weighting transformers, as efforts, and outputs the overall coolant weighted average, which is then input into the coupled feedback resistor $R3_{\rho fw}$.
- $TF_{\rho f, j, i}$: Fuel temperature weighting transformer. This element incorporates the contribution of the fuel axial segment i , radial shell j into the fuel's overall weighted average. There are N_f of these elements, one per fuel radial shell.

- $1_{\rho f}$: Fuel temperature weighted average 1-junction. This element sums the individual contributions from the fuel weighting transformers, as efforts, and outputs the overall fuel weighted average, which is then input into the coupled feedback resistor $R3_{\rho fw}$.
- $R3_{\rho fw}$: Coupled feedback 3-port resistor. This element accepts the weighted compositional averages as inputs, and evaluates the nonlinear function, which yields the coupled reactivity feedback. This output is then supplied to the reactivity-summing 1_{ρ} -junction.

In all three of the above reactivity representations, the following constituent expressions are used:

$$e_{\text{TSE}_{\rho,ex}}(t) = \rho_{ex}(t), \quad (3.8a)$$

$$C_{\rho,b} = 1, \quad (3.8b)$$

in which:

$e_{\text{TSE}_{\rho,ex}}(t)$ = Effort output by the external reactivity time-modulated source of effort $\text{TSE}_{\rho,ex}$. Dimensionless.

$C_{\rho,b}$ = Bias reactivity capacitor's capacitance. Dimensionless.

In the linear separable feedback representation (Figure 3.3), the following constituent expressions are used:

$$\mu_{\rho w,i} = \frac{1}{\alpha_{w,i}}, \quad (3.9a)$$

$$\mu_{\rho f j,i} = \frac{1}{f_{vj}\alpha_{f,i}}, \quad (3.9b)$$

in which:

$\mu_{\rho w,i}$ = The parameter of the coolant thermal feedback transformer $\text{TF}_{\rho w,i}$. Units: K.

$\mu_{\rho f j,i}$ = The parameter of the fuel thermal feedback transformer $\text{TF}_{\rho f j,i}$. Units: K.

$\alpha_{w,i}$ = Coolant axial segment i reactivity feedback coefficient. Units: K^{-1} .

$\alpha_{f,i}$ = Fuel axial segment i reactivity feedback coefficient, intended to be used with the fuel average temperature (here assumed BE1). Units: K^{-1} .

f_{vj} = Fuel radial shell j volume fraction, used for spatial average calculation. It is given by Eq. (2.162). Dimensionless.

In the nonlinear separable feedback representation (Figure 3.4), the R2 elements have only one constituent expression each — because of the causality and the active bonds used, the other output is zero. The fuel thermal feedback transformers now only represent the radial averaging fractions. The constituent expressions are:

$$e_{R2_{\rho w,i}}(e_{T,i,w}) = W_{T,i,w} R_{T,i,w}(e_{T,i,w}), \quad (3.10a)$$

$$e_{R2_{\rho f,i}}(e_{T,i,f}) = W_{T,i,f} R_{T,i,f}(e_{T,i,f}), \quad (3.10b)$$

$$\mu_{\rho f j, i} = \frac{1}{f_{v j}}, \quad (3.10c)$$

in which:

- $e_{R2_{\rho w, i}}(e_{T, i, w})$ = Effort output by the coolant thermal feedback resistor $R2_{\rho w, i}$. Dimensionless.
- $e_{R2_{\rho f, i}}(e_{T, i, f})$ = Effort output by the fuel thermal feedback resistor $R2_{\rho f, i}$. Dimensionless.
- $e_{T, i, w}$ = Effort (temperature) supplied to $R2_{\rho w, i}$ from the $0_{w, i}$ junction. Units: K.
- $e_{T, i, f}$ = Effort (temperature) supplied to $R2_{\rho f, i}$ from the $1_{\rho f, i}$ junction. This is the average fuel temperature of axial segment i . Units: K.
- $\mu_{\rho f j, i}$ = The parameter of the radial averaging transformer $TF_{\rho f j, i}$. Dimensionless.

In the coupled feedback representation (Figure 3.5), the R3 element has only one constituent expression — again, because of the causality and the active bonds used. The other outputs are zero. The elements' constituent expressions are:

$$\mu_{\rho w, i} = \frac{1}{W_{T, i, w}}, \quad (3.11a)$$

$$\mu_{\rho f j, i} = \frac{1}{f_{v j} W_{T, i, f}}, \quad (3.11b)$$

$$e_{R3_{\rho f w}}(e_{\bar{T}, w}, e_{\bar{T}, f}) = R(e_{\bar{T}, w}, e_{\bar{T}, f}), \quad (3.11c)$$

in which:

- $\mu_{\rho w, i}$ = The parameter of the coolant temperature weighting transformer $TF_{\rho w, i}$. Dimensionless.
- $\mu_{\rho f j, i}$ = The parameter of the fuel temperature weighting transformer $TF_{\rho f j, i}$. Dimensionless.
- $e_{\bar{T}, w}$ = Coolant overall weighted temperature average, supplied as one of the inputs to the coupled reactivity feedback 3-port resistor. Units: K.
- $e_{\bar{T}, f}$ = Fuel overall weighted temperature average, supplied as one of the inputs to the coupled reactivity feedback 3-port resistor. Units: K.
- $e_{R3_{\rho f w}}(e_{\bar{T}, w}, e_{\bar{T}, f})$
= Effort output by the coupled reactivity feedback-computing $R3_{\rho f w}$ element. Dimensionless.

This completes the summary of the nonlinear point kinetics equations bond graph representation. This assembly of representation methods, together with the systems-level thermal hydraulic model representation in section 3.2, is expected to be most useful for bond graph-based systems and subchannel code development, because this is where point kinetics equations tend to be used the most. The coupled point kinetics and systems-level thermal hydraulic models were tested on a benchmark PWR RIA problem; this test is given in chapter 5.

3.2 Systems-level Thermal Hydraulics

Subsection 2.1.4.3 gave the complete systems-level equation set. Here, a simplified version of these equations, fit for modeling the approximately isobaric bulk flow in a PWR core, is presented, discretized and represented with bond graphs.

We make several assumptions, in addition to the ones made in subsection 2.1.4.3:

1. The flow is assumed to be single (liquid) phase, single channel, and isobaric. All properties are therefore only functions of temperature.
2. The axial mass flow rate is assumed constant.
3. The flow is assumed to be inviscid, with enthalpy density variations due only to cooling and direct energy deposition.
4. The coolant flow area is assumed constant and uniform.

Under these assumptions, Eq. (2.149b) alone is sufficient to model the enthalpy conservation in the core. It becomes:

$$\frac{\partial}{\partial t} h_v(t, x) = -\frac{\partial}{\partial x} \left[V_w(T_w) h_v(t, x) \right] + \dot{u}'''(t, x), \quad (3.12)$$

in which:

- $h_v(t, x), h_v(T_w)$ = Coolant volumetric enthalpy density. Units: kJ/m³.
- $T_w, T_w(h_v)$ = Coolant temperature. Units: K.
- $V_w(T_w)$ = Coolant axial velocity, defined below. Units: m/s.
- $\dot{u}'''(t, x)$ = Coolant heat rate density, which accounts for direct energy deposition, and fuel rod cooling. Units: W/m³.

Under the above assumptions, the axial velocity is given by:

$$V_w(T_w) = \frac{\dot{m}}{\rho_w(T_w) A_w}, \quad (3.13)$$

in which:

- \dot{m} = Core mass flow rate, assumed constant. Units: kg/s.
- $\rho_w(T)$ = Coolant density. Units: kg/m³.
- A_w = Coolant flow area, assumed uniform throughout the core. Units: m².

The following additional nomenclature will be used for thermohydraulic properties throughout this text:

- $h, h(T_w)$ = Heat transfer coefficient. Units: W/m² K.
- $h_w(T_w)$ = Coolant specific enthalpy. Units: kJ/kg.

$k_w(T_w)$ = Coolant thermal conductivity. Units: W/m K.

$\mu_w(T_w)$ = Coolant dynamic viscosity. Units: Pa s

$c_{pw}(T_w)$ = Coolant specific heat capacity at constant pressure. Units: kJ/kg K.

To discretize the flow into N_i axial regions, we integrate Eq. (3.12) over axial region i of height Δz_i to obtain:

$$\frac{d}{dt}H_{w,i}(t) = \dot{H}_{w,i-1}(t) - \dot{H}_{w,i}(t) + \dot{Q}_i^c(t) + \dot{Q}_i^\gamma(t), \quad (3.14)$$

in which:

Δz_i = Height of the axial region i . Units: m.

$H_{w,i}(t)$ = Total coolant enthalpy in axial region i . Units: J.

$\dot{H}_{w,i}(t)$ = Rate of enthalpy advection from axial region i to axial region $i + 1$. Units: W.

$\dot{Q}_i^c(t)$ = Clad cooling rate for axial coolant region i . Units: W.

$\dot{Q}_i^\gamma(t)$ = Direct energy deposition rate into axial coolant region i . Units: W.

Using the upwind differencing scheme to approximate the inter-segment advection rates, $\dot{H}_{w,i}(t)$ becomes:

$$\dot{H}_{w,i}(t) = \dot{m}h_w(\bar{T}_{w,i}), \quad (3.15)$$

in which:

$\bar{T}_{w,i}, \bar{T}_{w,i}(H_{w,i})$ = Average coolant temperature in axial region i . Units: K.

Equation (3.15) can be used directly to model a specified inlet temperature BC.

Volumetric enthalpy density $h_v(T_w)$ can be expressed as a function of temperature:

$$h_v(T_w) = \rho_w(T_w) h_w(T_w). \quad (3.16)$$

In general, even for single phase flow, this dependence is not necessarily strictly monotonic, and therefore is not necessarily invertible. To express T_w in terms of h_v , the function needs to be invertible. For this reason, the simplifications made in this section only work for a single phase fluid sufficiently far from saturation, where the monotonic dependence holds. The average axial region i coolant temperature $\bar{T}_{w,i}$ is therefore related to $H_{w,i}$ through:

$$\bar{T}_{w,i}(H_{w,i}) = T_{w,h_v} \left(\frac{H_{w,i}}{\Delta \mathcal{V}_{w,i}} \right), \quad (3.17)$$

in which:

$T_{w,h_v}(h_v)$ = Coolant temperature as a function of volumetric enthalpy density. As discussed above, this function is not always necessarily defined. Units: K.

$\Delta \mathcal{V}_{w,i}$ = Axial coolant segment i volume, given by $\Delta z_i A_w$. Units: m³.

Subsection 2.1.4.6 summarizes an already-discretized fuel rod model. Besides notation, only one minor adjustment needs to be made to it here. The equations in subsection 2.1.4.6 are given for a single fuel rod, however, in the present model, we are analyzing an effective ‘‘averaged’’ fuel rod, which represents all of the rods in the core combined. In effect, this simply means multiplying Δz_i by a scaling factor N_{pi} , which adjusts the effective shell volumes and heat transfer areas:

N_{pi} = Axial fuel segment i scaling factor. Normally, this is the effective number of rods in the core at axial level i . Dimensionless.

Assuming the fuel pellet to be divided into N_{fj} equal radial shells, and the fuel rods to be divided into N_i axial segments (which match the N_i coolant axial segments), the following equations, adapted from Eq. (2.158), govern the resulting discretized system:

$$\frac{d}{dt}U_{fj,i}(t) = \begin{cases} -\frac{\bar{T}_{fj,i} - \bar{T}_{fj+1,i}}{R_{fj,i}(\bar{T}_{fj,i}, \bar{T}_{fj+1,i})} + \dot{Q}_{j,i}^f(t) & \text{if } j = 1, \\ \frac{\bar{T}_{fj-1,i} - \bar{T}_{fj,i}}{R_{fj-1,i}(\bar{T}_{fj-1,i}, \bar{T}_{fj,i})} - \frac{\bar{T}_{fj,i} - \bar{T}_{fj+1,i}}{R_{fj,i}(\bar{T}_{fj,i}, \bar{T}_{fj+1,i})} + \dot{Q}_{j,i}^f(t) & \text{if } 1 < j < N_{fj}, \\ \frac{\bar{T}_{fj-1,i} - \bar{T}_{fj,i}}{R_{fj-1,i}(\bar{T}_{fj-1,i}, \bar{T}_{fj,i})} - \frac{\bar{T}_{fj,i} - \bar{T}_{g,i}}{R_{fg,i}(\bar{T}_{fj,i}, \bar{T}_{g,i})} + \dot{Q}_{j,i}^f(t) & \text{if } j = N_{fj}, \end{cases} \quad (3.18a)$$

$$\frac{d}{dt}U_{g,i}(t) = \frac{\bar{T}_{fN_{fj},i} - \bar{T}_{g,i}}{R_{fg,i}(\bar{T}_{fN_{fj},i}, \bar{T}_{g,i})} - \frac{\bar{T}_{g,i} - \bar{T}_{c,i}}{R_{gc,i}(\bar{T}_{g,i}, \bar{T}_{c,i})}, \quad (3.18b)$$

$$\frac{d}{dt}U_{c,i}(t) = \frac{\bar{T}_{g,i} - \bar{T}_{c,i}}{R_{gc,i}(\bar{T}_{g,i}, \bar{T}_{c,i})} - \frac{\bar{T}_{c,i} - \bar{T}_{w,i}}{R_{cw,i}(\bar{T}_{c,i}, \bar{T}_{w,i})}, \quad (3.18c)$$

Note, that the second term in Eq. (3.18c) is the same as $\dot{Q}_i^c(t)$ in Eq. (3.14):

$$\dot{Q}_i^c(t) = \frac{\bar{T}_{c,i} - \bar{T}_{w,i}}{R_{cw,i}(\bar{T}_{c,i}, \bar{T}_{w,i})}. \quad (3.19)$$

The following nomenclature is used here:

- f, g, c = Fuel, gap and clad subscripts, respectively.
- $\bar{T}_{fj,i}$ = Average fuel temperature in radial shell j , axial segment i . Units: K.
- $\bar{T}_{g,i}, \bar{T}_{c,i}$ = Axial segment i average gap and clad temperatures. Units: K.
- $U_{fj,i}(t)$ = Total thermal energy stored in fuel radial shell j , axial segment i . Units: J.
- $U_{g,i}(t), U_{c,i}(t)$ = Total thermal energies stored in gap and clad axial segment i , respectively. Units: J.
- $R_{fj,i}(\bar{T}_{fj,i}, \bar{T}_{fj+1,i})$ = Thermal resistance between fuel radial shell j and $j + 1$ at axial segment i . It is defined below. Units: W/K.

$R_{fg,i}(\bar{T}_{fN_{fj,i}}, \bar{T}_{g,i}), R_{gc,i}(\bar{T}_{g,i}, \bar{T}_{c,i}), R_{cw,i}(\bar{T}_{c,i}, \bar{T}_{w,i})$
 = Fuel outer shell-gap, gap-clad, and clad-coolant thermal resistances at axial segment i . They are defined below. Units: W/K.

$\dot{Q}_{j,i}^f(t)$ = Radial shell j , axial segment i fission heat rate. Units: W.

The thermal resistances are adapted directly from Eq. (2.158), and from Newton's law of cooling:

$$R_{fj,i}(\bar{T}_{fj,i}, \bar{T}_{fj+1,i}) = \frac{1}{2\pi r_{fj} N_{pi} \Delta z_i} \left[\frac{\Delta r_{fj}/2}{k_f(\bar{T}_{fj,i})} + \frac{\Delta r_{fj+1}/2}{k_f(\bar{T}_{fj+1,i})} \right], \quad (3.20a)$$

$$R_{fg,i}(\bar{T}_{fN_{fj,i}}, \bar{T}_{g,i}) = \frac{1}{2\pi r_f N_{pi} \Delta z_i} \left[\frac{\Delta r_{fj}/2}{k_f(\bar{T}_{fN_{fj,i}})} + \frac{\Delta r_g/2}{k_g(\bar{T}_{g,i})} \right], \quad (3.20b)$$

$$R_{gc,i}(\bar{T}_{g,i}, \bar{T}_{c,i}) = \frac{1}{2\pi r_g N_{pi} \Delta z_i} \left[\frac{\Delta r_g/2}{k_g(\bar{T}_{g,i})} + \frac{\Delta r_c/2}{k_c(\bar{T}_{c,i})} \right], \quad (3.20c)$$

$$R_{cw,i}(\bar{T}_{c,i}, \bar{T}_{w,i}) = \frac{1}{2\pi r_c N_{pi} \Delta z_i} \left[\frac{\Delta r_c/2}{k_c(\bar{T}_{c,i})} + \frac{1}{h(\bar{T}_{w,i})} \right]. \quad (3.20d)$$

The following nomenclature was used here:

r_{fj} = Outer radius of fuel shell j . Units: m.

$r_{f/g/c}$ = Fuel/gap/clad outer radius. Units: m.

Δr_{fj} = Fuel shell j thickness. Units: m.

$\Delta r_{g/c}$ = Gap/clad thickness. Units: m.

$k_{f/g/c}(\bar{T}_{f/g/c})$ = Fuel/gap/clad thermal conductivity. Units: W/mK.

Temperatures are related to the corresponding total thermal energies via Eq. (2.159).

Lastly, as discussed in subsection 2.1.5.3, the coolant direct energy deposition and fuel fission heat rates $\dot{Q}_i^\gamma(t)$ and $\dot{Q}_{j,i}^f(t)$ are simply fixed known (from the point kinetics specification) fractions of the core power $P(t)$:

$$\dot{Q}_i^\gamma(t) = \gamma_i P(t), \quad (3.21a)$$

$$\dot{Q}_{j,i}^f(t) = S_{j,i} P(t) = f_{vj} S_i P(t), \quad (3.21b)$$

in which:

γ_i = Coolant axial segment i direct energy deposition fraction. Dimensionless.

S_i = Axial segment i fuel fission heat rate fraction. This quantity is used if power is assumed to be radially uniform across the fuel pin. If this assumption is not made, $S_{j,i}$ must be given instead. Dimensionless.

$S_{j,i}$ = Axial segment i , radial shell j fuel fission heat rate fraction. This quantity is only given if non-uniform radial power distribution across the fuel pin is assumed. Dimensionless.

The source fractions must add up to 1:

$$\sum_{\text{all } i,j} S_{j,i} + \sum_{\text{all } i} \gamma_i = 1. \quad (3.22)$$

This completes the summary of the systems-level thermal hydraulic model; it can now be represented using bond graphs.

The bond graph representations of incompressible thermal hydraulic models in literature normally use internal energy, and not enthalpy, as the displacement on thermal capacitors [83, 93]; because the two quantities are similar, here, enthalpy will be used as the displacement instead. The various rates of change of enthalpy, as well as contributions to them, are therefore flows, and the temperatures are efforts. 1-dimensional conduction has been modeled with bond graphs in some detail; in it, internal energy is again a displacement, heat rates are flows, and temperature, again, an effort [84, 85]. Using these conventions, we can proceed to construct the bond graph representations of the model presented in this section.

Equation (3.14) is a rate balance equation; because enthalpy is a displacement, and its rate a change of flow, the equation is a balance of flows, which is enforced by a 0-junction. Coolant temperature (an effort) is a function of enthalpy (a displacement) through Eq. (3.17), which is enforced by a capacitor. The upwind enthalpy flow rate specified in Eq. (3.15) is a flow which is a function of temperature (an effort), and this flow is simultaneously removed from region i and added to region $i + 1$. This makes an R2 resistive element with an active bond on the outlet port a good candidate for setting the flow. For core inlet enthalpy flow rate, a time-modulated source of flow can be used instead; similarly, a flow sink can be used at the outlet. Lastly, the fission heat and direction energy deposition rates, modeled by Eqs. (3.21), are simply proportional to power (an effort in section 3.1), which, like the thermal feedback, can be easily enforced through a linear transformer with two active bonds. Such transformers were already mentioned in section 3.1.

Leaving the $\dot{Q}_i^c(t)$ term out for now, the advection physics are represented through Figure 3.6. In it, the elements are:

- $C_{w,i}$: Axial segment i coolant enthalpy capacitor. This element's output is axial segment i average coolant temperature, and through it, axial segment i enthalpy is an unknown.
- $R2_{w,i}$: 2-port resistive element which controls the enthalpy advection rate between axial segments i and $i + 1$. Because upwind differencing is used, only the temperature from axial segment i (delivered via a full bond) is used to compute the advection rates (flows), which are then taken from $0_{w,i}$ and added to $0_{w,i+1}$.

The following constituent expressions are used in Figure 3.6:

$$e_{C_{w,i}}(q_{w,i}) = T_{w,h_v} \left(\frac{q_{w,i}}{\Delta V_{w,i}} \right), \quad (3.23a)$$

$$f_{R2_{w,i}}^{in/out}(e_{T,w,i}) = \dot{m} h_w(e_{T,w,i}). \quad (3.23b)$$

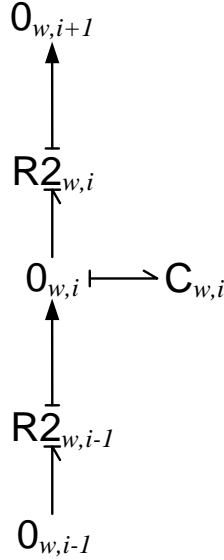


Figure 3.6: 1D Advection Bond Graph Representation

The following nomenclature is used:

- $q_{w,i}$ = Displacement stored by the enthalpy capacitor $C_{w,i}$. This is the total coolant enthalpy in axial segment i . Units: J.
- $e_{C_{w,i}}(q_{w,i})$ = Effort output by $C_{w,i}$ as a function of the displacement $q_{w,i}$ stored by this capacitor. Units: K.
- $e_{T,w,i}$ = Effort delivered to $R2_{w,i}$ via a full bond from $0_{w,i}$. This is the average segment i coolant temperature. Units: K.
- $f_{R2_{w,i}}^{in/out}(e_{T,w,i})$ = Flow into/out of $R2_{w,i}$, set by this 2-port resistor as a function of $e_{T,w,i}$. This flow is output onto both ports of the $R2_{w,i}$ element. Units: W.

The direct energy deposition rate provided to $0_{w,i}$ from 1_P through $TF_{Pw,i}$ is characterized by the transformer's parameter, given by:

$$\mu_{Pw,i} = \gamma_i, \quad (3.24)$$

in which:

- $\mu_{Pw,i}$ = The parameter of the axial segment i direct energy deposition transformer $TF_{Pw,i}$. Dimensionless.

Lastly, we can represent the fuel, gap and clad radial heat transfer, specified by Eqs. (3.18). In Ref. [84], diffusive heat transfer rates (individual terms of Eqs. (3.18) are set by modulated single port resistors, which are supplied a difference in neighboring efforts (imposed through 1-junctions) and the temperatures are imposed onto 0-junctions through nonlinear capacitors (same as for coolant, but using Eq. (2.159) instead).

The representation of the in-fuel radial heat transfer (Eq. (3.18a)) is shown in Figure 3.7; the fuel, gap and clad radial heat transfer (Eqs. (3.18b) and (3.18c)) representation is shown in Figure 3.8.

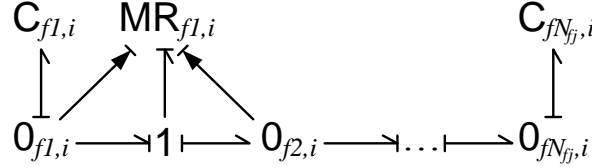


Figure 3.7: Fuel Cylindrical HDE Bond Graph Representation

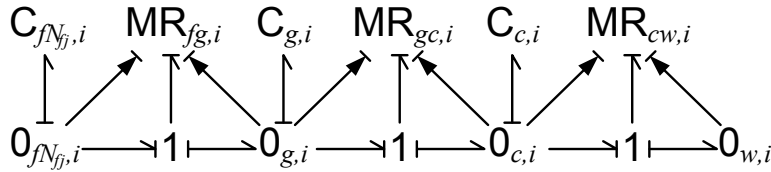


Figure 3.8: Fuel, Gap and Clad Cylindrical HDE Bond Graph Representation

The elements here are:

- $C_{fj,i}$: Fuel shell thermal capacitor. This element stores the thermal energy of axial segment i , radial fuel shell j ; it applies Eq. (2.159) to compute its output temperature (effort). For every axial segment i , there are N_{fj} such elements.
- $MR_{fj,i}$: Thermal resistor between radial fuel shells j and $j + 1$, modulated by these shells average temperatures. For every axial segment i , there are $N_{fj} - 1$ such elements.
- 1 : The 1-junctions are present here to evaluate the temperature difference between neighboring cells, and to enforce heat current continuity between these cells.
- $0_{g/c,i}$: Gap/clad 0-junctions, which enforce energy conservation in the gap and clad materials at a given axial segment i .
- $C_{g/c}$: Gap/clad thermal capacitors. These function exactly the same way as $C_{fj,i}$, but for the gap and clad regions.
- $MR_{fg,i}$, $MR_{gc,i}$: Fuel outer shell-gap and gap-clad temperature modulated resistive elements. These work exactly the same way as $MR_{fj,i}$.
- $MR_{cw,i}$: Clad-coolant thermal resistive element. This element combines the conductive and convection thermal resistances.

The capacitive elements' constituent expressions are:

$$e_{C_{g/c,i}}(q_{g/c,i}) = T \left(\frac{q_{g/c,i}}{\Delta V_{g/c,i}} \right), \quad (3.25)$$

in which:

$q_{g/c,i}$ = Gap/clad thermal displacement (total thermal energy) at axial segment i . Units: J.

$e_{C_{g/c,i}}(q_{g/c,i})$ = Effort output by $C_{g/c}$ as a function of $q_{g/c,i}$. Units: K.

$\Delta V_{g/c,i}$ = Gap/clad axial segment effective volume; this quantity is scaled by N_{pi} . Units: m^3 .

The temperature function in Eq. (3.25) is from Eq. (2.159).

The resistive parameters of the modulated resistors $MR_{fj,i}$, $MR_{fg,i}$, $MR_{gc,i}$ and $MR_{cw,i}$ come from Eqs. (3.20):

$$R_{fj,i}(e_{fj,i}, e_{fj+1,i}) = \frac{1}{2\pi r_{fj} N_{pi} \Delta z_i} \left[\frac{\Delta r_{fj}/2}{k_f(e_{fj,i})} + \frac{\Delta r_{fj+1}/2}{k_f(e_{fj+1,i})} \right], \quad (3.26a)$$

$$R_{fg,i}(e_{fN_{fj,i}}, e_{g,i}) = \frac{1}{2\pi r_f N_{pi} \Delta z_i} \left[\frac{\Delta r_{fj}/2}{k_f(e_{fN_{fj,i}})} + \frac{\Delta r_g/2}{k_g(e_{g,i})} \right], \quad (3.26b)$$

$$R_{gc,i}(e_{g,i}, e_{c,i}) = \frac{1}{2\pi r_g N_{pi} \Delta z_i} \left[\frac{\Delta r_g/2}{k_g(e_{g,i})} + \frac{\Delta r_c/2}{k_c(e_{c,i})} \right], \quad (3.26c)$$

$$R_{cw,i}(e_{c,i}, e_{w,i}) = \frac{1}{2\pi r_c N_{pi} \Delta z_i} \left[\frac{\Delta r_c/2}{k_c(\bar{T}_{c,i})} + \frac{1}{h(e_{w,i})} \right]. \quad (3.26d)$$

The nomenclature here is:

$e_{fj,i}, e_{g,i}, e_{c,i}, e_{w,i}$ = Efforts corresponding to average fuel shell j , gap, clad and coolant axial segment i temperatures. Units: K.

$R_{fj,i}(e_{fj,i}, e_{fj+1,i})$ = The modulated resistance parameter of $MR_{fj,i}$. Units: W/K.

$R_{fg,i}(e_{fN_{fj,i}}, e_{g,i})$ = The modulated resistance parameter of $MR_{fg,i}$. Units: W/K.

$R_{gc,i}(e_{g,i}, e_{c,i})$ = The modulated resistance parameter of $MR_{gc,i}$. Units: W/K.

$R_{cw,i}(e_{c,i}, e_{w,i})$ = The modulated resistance parameter of $MR_{cw,i}$. Units: W/K.

The fission heat rate is provided to $0_{fj,i}$ from 1_P through a previously-mentioned constant coefficient transformer $TF_{Pfj,i}$:

$$\mu_{Pfj,i} = S_{j,i}, \quad (3.27)$$

in which:

$\mu_{Pfj,i}$ = The parameter of the $TF_{Pfj,i}$ element. Dimensionless.

This completes the summary of the nonlinear PKE and systems-level thermal hydraulics bond graph representation. The representation, developed here, is tested in chapter 5 on a PWR RIA problem.

3.3 Multidimensional, Multigroup Neutron Diffusion

Subsection 2.1.2.2 describes the discretized multigroup diffusion and delayed neutron precursor equations. Here, they will be used with slight simplifications: the individual fissionable isotope dependence of cross sections will be dropped, and a single index m will now be used to index all of the precursors with M precursor families in the system.

Many types of feedback are possible in spatial kinetics; here, to focus on the neutron diffusion and not on the thermal hydraulic model, it is assumed that a single finite volume is characterized by a single temperature, which all neutron properties of the node are dependent on. Equation (2.171) represents the power production equation in such scenario, which is equal to the rate of change of the total thermal energy in the node:

$$\frac{d}{dt}U_{i,j,k} = \Delta V_{i,j,k} \sum_{g'=1}^G \kappa \bar{\Sigma}_{fg',i,j,k}(t) \bar{\phi}_{g,i,j,k}(t), \quad (3.28)$$

and, assuming, for simplicity, a constant volumetric heat capacity c_v , the total thermal energy $U_{i,j,k}$ may be related to the average nodal temperature $\bar{T}_{i,j,k}$ via:

$$\bar{T}_{i,j,k} = \frac{1}{c_v \Delta V_{i,j,k}} U_{i,j,k}. \quad (3.29)$$

The nomenclature here was:

$U_{i,j,k}$ = Total thermal energy in node (i, j, k) . Units: J.

$\bar{T}_{i,j,k}$ = Average temperature in node (i, j, k) . Units: K.

c_v = Volumetric heat capacity of the homogenized nodal material, here assumed constant for simplicity. Units: J/m³ K.

The 1D, one-group neutron diffusion has previously been represented with bond graphs [84]. We can use, where applicable, the same convention as set in Ref. [84]: average neutron (group) fluxes act as efforts, the reaction rates as flows, and numbers of neutrons in nodes as displacements. Here, the convention is extended: the rate of decay of delayed neutron precursors is proportional to their concentration, so the delayed neutron precursor density will act as both an effort and a displacement. This will require the precursor capacitors to have unity moduli. The delayed neutron precursor reaction rates (either generation or decay) act as flows. Lastly, to model the thermal feedback, we use the same convention as in section 3.2: total thermal energy in a node acts as a displacement, its rate of change, the nodal power, as a flow, and the nodal temperature as an effort.

Under these convention, we can proceed to build a bond graph representation. Equation (2.110) is a rate (flow) balance, which we know to be enforced by 0-junctions. Similarly, Eq. (3.28) is also a flow. The rates of change in these equations are rates of change in displacements, which are also the flows supplied to the capacitors from the 0-junctions. The group external source rates are also flows, supplied to the corresponding 0-junctions from time-dependent sources of flow.

A number of local reactions occur: total interaction, fission, fission power generation, precursor formation and decay, and intergroup scattering. In Ref. [84], an R2 element represented all the

local reactions. In this problem, the number of quantities coupled through local reactions is N :

$$N = G + M + 1, \quad (3.30)$$

where:

N = Total number of quantities coupled through local reactions in a single node (i, j, k) .

This local coupling will be represented with a special (new, for automated bond graph processing) modulated N -port element, denoted here MRN. The element is modulated because of potential time dependence, which may be used to account for control element movement. All N inputs to the MRN constitute various efforts, and the outputs are local reaction or heat rates — flows, which are functions of the input efforts.

The internodal currents in Eq. (2.110) are represented, in principle, identically to the internodal currents in Ref. [84]: the differences in the nodal group fluxes are modeled by 1-junctions, and the flows are imposed by potentially modulated (to account for temperature dependence of the diffusion coefficients) 1-port resistors.

Lastly, to correctly represent the BCs, we must first recognize that both zero flux and zero current BCs essentially determine the amount of net current through the boundary. A Neumann BC (zero current, Eq. (2.114)) sets this flow explicitly, which is done through time-modulated (or in this case, constant) external flow sources. A zero current flow source connected to the flow-summing 0-junction is also equivalent to not having a flow, or a flow source, at all, which is the simplest representation. A non-zero, explicitly defined current would have to be enforced through a TSF element connected to the boundary node's 0-junction.

A zero flux BC (Eqs. (2.115)) sets the boundary flows by providing a “sink” to which the boundary node is connected through the boundary resistor. The sink's group flux, being an effort and explicitly known, is set through a TSE element connected to a BC sink 0-junction, although if this sink flux is a constant zero, it may be set by a constant SE element instead. The boundary node's 0-junction and the sink node's 0-junction are then connected to the BC sink the same way internodal connections are — through a 1-junction and a boundary resistor. This resistor, due to being only half as long as the internodal one (assuming uniform grid), is expected to be significantly weaker (lower resistance).

Combined, these equivalences result in the bond graph representation for node (i, j, k) given in Figure 3.9. The elements present are:

- $0_{g,i,j,k}^n$: Group g flux 0-junction for node (i, j, k) . This junction outputs the net rate of change the number of neutrons to the group g flux capacitor, and it outputs the group g scalar flux to all other elements. Below, it is connected to the internodal 1-junctions. There are G such elements per spatial node.
- $C_{g,i,j,k}^n$: Group g flux capacitor. This element stores the group g neutron and outputs the group g scalar flux. Through it, the number of group g neutrons becomes an unknown. There are G such elements per spatial node.
- $\text{TSF}_{g,i,j,k}^n$: Group g external time-dependent neutron source of flow. This elements is optional, and is present only if a nonzero external neutron source is present in group g .

- $0_{m,i,j,k}^c$: Delayed neutron precursor family m 0-junction. As shown, this junction is a trivial 2-port, present only because capacitive and resistive elements, in some sorting algorithms, may only be connected to junction elements. There are M such elements per spatial node.
- $C_{m,i,j,k}^c$: Delayed neutron precursor family m capacitor. This element stores the precursor family m concentration, and outputs it as well; it is then, through $0_{m,i,j,k}^c$, supplied to the local reactions coupling element. These elements make the delayed neutron precursor densities the unknowns in the model. There are M such elements per spatial node.
- $0_{i,j,k}^T$: Thermal 0 junction in node (i, j, k) . Here, it is present as a trivial 2-port junction, but it may in principle be more useful: a thermal hydraulic model that makes use of the node (i, j, k) thermal energy would set a heat rate balance through this junction. This is not shown here. There is one such element per spatial node.
- $C_{i,j,k}^T$: Thermal capacitor in node (i, j, k) . This element converts the total thermal energy in node (i, j, k) to average nodal temperature, and makes the total thermal energy an unknown.
- $MRN_{i,j,k}$: Node (i, j, k) local reactions coupling element. This element models all of the local reactions that occur in the node.

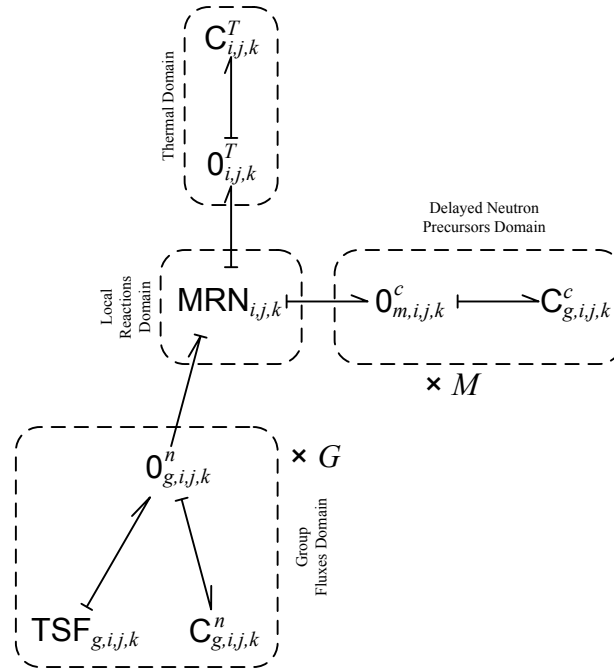


Figure 3.9: Multigroup NDE Node with Precursors Bond Graph Representation

The capacitive and source elements' constituent expressions are:

$$C_{g,i,j,k}^n = \frac{V_{ng}}{\Delta V_{i,j,k}}, \quad (3.31a)$$

$$C_{m,i,j,k}^c = 1, \quad (3.31b)$$

$$C_{i,j,k}^T = \frac{1}{c_v \Delta V_{i,j,k}}, \quad (3.31c)$$

$$f_{\text{TSF}_{g,i,j,k}^n}(t) = S_{\text{exg},i,j,k}(t), \quad (3.31d)$$

in which:

$C_{g,i,j,k}^n$ = The capacitance of $C_{g,i,j,k}^n$. Units: $\text{/cm}^2 \text{ s}$.

$C_{m,i,j,k}^c$ = The capacitance of $C_{m,i,j,k}^c$. Dimensionless.

$C_{i,j,k}^T$ = The capacitance of $C_{i,j,k}^T$. Units: K/J .

$f_{\text{TSF}_{g,i,j,k}^n}(t)$ = Flow output by $\text{TSF}_{g,i,j,k}^n$. Units: neutrons/s .

The ports on the $\text{MRN}_{i,j,k}$ are assumed to be ordered as follows: group fluxes ($g = 1, \dots, G$), then precursors ($m = 1, \dots, M$) and then the thermal port. With this assumption, the indices can be expressed in terms of port number p as follows:

$$g(p) = \begin{cases} p & \text{if } 1 \leq p \leq G, \\ \text{N/A} & \text{otherwise,} \end{cases} \quad (3.32a)$$

$$m(p) = \begin{cases} p - G & \text{if } G + 1 \leq p \leq G + M, \\ \text{N/A} & \text{otherwise,} \end{cases} \quad (3.32b)$$

in which:

p = $\text{MRN}_{i,j,k}$ port index. p can be between 1 and N .

$g(p)$ = Group index corresponding to port index p , if this port index corresponds to a group flux port. $g(p)$ can be between 1 and G , and the corresponding p are between 1 and G as well. When p is in this range, we denote it as $p \in \mathcal{G}$.

$m(p)$ = Precursor family index corresponding to port index p , if this port index corresponds to a precursor family port. $m(p)$ can be between 1 and M , and the corresponding p are between $G + 1$ and $G + M$. When m is in this range, we denote it as $p \in \mathcal{M}$.

$p = N$ always corresponds to the thermal port.

With this nomenclature, we can define the constituent expression of $\text{MRN}_{i,j,k}$ as a modulated matrix, in full admittance causality. This is an $N \times N$ matrix, the product of which with the vector of (i, j, k) efforts, ordered according to the convention above, is the vector of (i, j, k) flows, ordered according to the convention above.

Denoting this matrix $\mathbf{Y}_{i,j,k}^L(t, e_{\text{MRN}_{i,j,k}}^T)$, we now define its individual elements. Row p , column p' element is given by Eqs. (3.33):

$$\begin{aligned}
 Y_{i,j,k}^{L,pp'}(t, e_{\text{MRN}_{i,j,k}}^T) &= \\
 &= \Delta \mathcal{V}_{i,j,k} \left[\bar{\Sigma}_{tg,i,j,k}(t, e_{\text{MRN}_{i,j,k}}^T) - \bar{\Sigma}_{sgg,i,j,k}(t, e_{\text{MRN}_{i,j,k}}^T) - \overline{\chi \nu_p \Sigma}_{fgg,i,j,k}(t, e_{\text{MRN}_{i,j,k}}^T) \right] \\
 &\quad \text{with } p = p' \in \mathcal{G}, \quad g = g' = g(p),
 \end{aligned} \tag{3.33a}$$

$$\begin{aligned}
 &= \Delta \mathcal{V}_{i,j,k} \left[-\bar{\Sigma}_{sgg',i,j,k}(t, e_{\text{MRN}_{i,j,k}}^T) - \overline{\chi \nu_p \Sigma}_{fgg',i,j,k}(t, e_{\text{MRN}_{i,j,k}}^T) \right] \\
 &\quad \text{with } p, p' \in \mathcal{G}, \quad p \neq p', \quad g = g(p), \quad g' = g(p'),
 \end{aligned} \tag{3.33b}$$

$$\begin{aligned}
 &= \Delta \mathcal{V}_{i,j,k} \left[-\chi_{dg,m'} \lambda_{m'} \right] \\
 &\quad \text{with } p \in \mathcal{G}, \quad p' \in \mathcal{M}, \quad g = g(p), \quad m' = m(p'),
 \end{aligned} \tag{3.33c}$$

$$\begin{aligned}
 &= 0 \\
 &\quad \text{with } p \in \mathcal{G}, \quad p' = N,
 \end{aligned} \tag{3.33d}$$

$$\begin{aligned}
 &= -\lambda_m \\
 &\quad \text{with } p = p' \in \mathcal{M}, \quad m = m' = m(p),
 \end{aligned} \tag{3.33e}$$

$$\begin{aligned}
 &= \overline{\nu_{d,m} \Sigma}_{fg',i,j,k}(t, e_{\text{MRN}_{i,j,k}}^T) \\
 &\quad \text{with } p \in \mathcal{M}, \quad p' \in \mathcal{G}, \quad m = m(p), \quad g' = g(p'),
 \end{aligned} \tag{3.33f}$$

$$\begin{aligned}
 &= 0 \\
 &\quad \text{with } p \in \mathcal{M}, \quad p' = N,
 \end{aligned} \tag{3.33g}$$

$$\begin{aligned}
 &= \Delta \mathcal{V}_{i,j,k} \left[\overline{\kappa \Sigma}_{fg',i,j,k}(t, e_{\text{MRN}_{i,j,k}}^T) \right] \\
 &\quad \text{with } p = N, \quad p' \in \mathcal{G}, \quad g' = g(p'),
 \end{aligned} \tag{3.33h}$$

$$\begin{aligned}
 &= 0 \\
 &\quad \text{with } p = N, \quad p' \in \mathcal{M},
 \end{aligned} \tag{3.33i}$$

$$\begin{aligned}
 &= 0 \\
 &\quad \text{with } p = p' = N.
 \end{aligned} \tag{3.33j}$$

The following nomenclature was used here:

$e_{\text{MRN}_{i,j,k}}^T$ = Effort supplied to the thermal port of element $\text{MRN}_{i,j,k}$; this is the nodal average temperature. Units: K.

$Y_{i,j,k}^{L,pp'}(t, e_{\text{MRN}_{i,j,k}}^T)$ = Row p , column p' element of $\mathbf{Y}_{i,j,k}^L(t, e_{\text{MRN}_{i,j,k}}^T)$.

$\bar{\Sigma}_{tg,i,j,k}(t, e_{\text{MRN}_{i,j,k}}^T)$, $\bar{\Sigma}_{sgg',i,j,k}(t, e_{\text{MRN}_{i,j,k}}^T)$, $\overline{\chi \nu_p \Sigma}_{fgg',i,j,k}(t, e_{\text{MRN}_{i,j,k}}^T)$,

$\overline{\nu_{d,m} \Sigma}_{fg',i,j,k}(t, e_{\text{MRN}_{i,j,k}}^T)$, $\overline{\kappa \Sigma}_{fg',i,j,k}(t, e_{\text{MRN}_{i,j,k}}^T)$

= Potentially time- and temperature-dependent nodal properties. In many models, most of these will actually be constants.

A group g internodal resistance is modeled as a 1-junction and MR-element pair. These resistances only govern diffusion, and none of the local reaction rates. Figure 3.10 illustrates this bond graph representation.

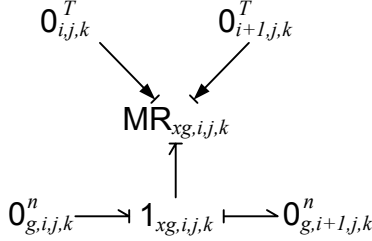


Figure 3.10: Group g (i, j, k) to ($i + 1, j, k$) Internodal Resistance Bond Graph Representation

The 0-junctions here are from the representation in Figure 3.9. The new elements are:

- $1_{xg,i,j,k}$: 1-junction between nodes (i, j, k) and ($i + 1, j, k$), used to evaluate the difference in group g fluxes between the two nodes, supply this difference to the resistor, and enforce the net current, computed by the resistor, between the two nodes.
- $MR_{xg,i,j,k}$: Modulated 1-port resistive element, used to evaluate the net group g current between nodes (i, j, k) and ($i + 1, j, k$). This element is modulated the two nodes' average temperature, which are used to modify the corresponding group diffusion coefficients. The coefficients may also be functions of time, to account for control element movement.

The modulated internodal resistance is given by:

$$R_{g,i,j,k}(t, e_{0_{i,j,k}^T}, e_{0_{i+1,j,k}^T}) = \frac{1}{A_{x,j,k}} \left[\frac{\Delta x_i/2}{\bar{D}_{xg,i,j,k}(t, e_{0_{i,j,k}^T})} + \frac{\Delta x_{i+1}/2}{\bar{D}_{xg,i+1,j,k}(t, e_{0_{i+1,j,k}^T})} \right], \quad (3.34)$$

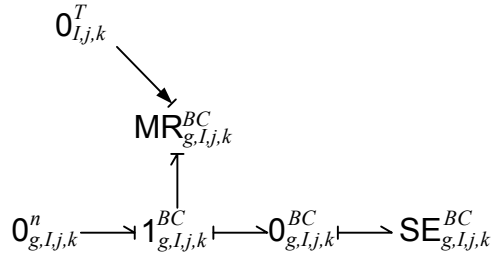
in which:

- $R_{g,i,j,k}(t, e_{0_{i,j,k}^T}, e_{0_{i+1,j,k}^T})$ = The resistance of the $MR_{xg,i,j,k}$ internodal resistor. Units: cm^{-2} .
- $e_{0_{i,j,k}^T}$ = Effort delivered to $MR_{xg,i,j,k}$ from $0_{i,j,k}^T$ via a modulating signal bond. This is the node (i, j, k) average temperature.
- $\bar{D}_{xg,i,j,k}(t, e_{0_{i,j,k}^T})$ = Node (i, j, k) group g x -directional diffusion coefficient, possibly time- and temperature-dependent. Compared to other group properties, the diffusion coefficient's dependence on temperature is usually weaker, and so this modulation is sometimes dropped.

The y - and z -directed internodal diffusion terms can be constructed trivially from Figure 3.10.

As was discussed above, reflective BCs are modeled by simply omitting the internodal resistance block. Zero flux BC is represented in Figure 3.11. The elements here are:

- $1_{g,I,j,k}^{BC}$: 1-junction which supplies the difference between the boundary node (I, j, k) group g flux, and the (zero) boundary flux, to $MR_{g,I,j,k}^{BC}$.
- $MR_{g,I,j,k}^{BC}$: Modulated resistor, which computes the group g leakage into the zero-flux boundary from node (I, j, k) . Like other internodal and boundary resistors in this representation, it is modulated by the boundary node's temperature.
- $0_{g,I,j,k}^{BC}$: 0-junction which conveys the zero flux to the BC resistor and the leaking current to the sink. Here, it is a trivial 2-port junction, but with more complex (combined) BCs, it may be used to balance currents.
- $SE_{g,I,j,k}^{BC}$: Group g boundary source of effort, which sets the boundary flux. Normally, it is zero; for a more complex BC, it may be a TSE element instead.


 Figure 3.11: Group g Boundary Node (I, j, k) Dirichlet BC Bond Graph Representation

The equations are:

$$e_{SE_{g,I,j,k}^{BC}}^{BC} = 0, \quad (3.35a)$$

$$R_{g,I,j,k}^{BC}(t, e_{0_{I,j,k}^T}) = \frac{1}{A_{x,j,k}} \left[\frac{\Delta x_I/2}{\overline{D}_{xg,I,j,k}(t, e_{0_{I,j,k}^T})} \right], \quad (3.35b)$$

in which:

$e_{SE_{g,I,j,k}^{BC}}^{BC}$ = Effort enforced by $SE_{g,I,j,k}^{BC}$. This is the group g boundary (zero) flux. Units: neutrons/cm² s.

$R_{g,I,j,k}^{BC}(t, e_{0_{I,j,k}^T})$ = Node (I, j, k) group g boundary resistor's resistance. It is modulated by $e_{0_{I,j,k}^T}$, which is the node's average temperature.

As with the internodal resistances, Figure 3.11 can be trivially extended to other dimensions.

It should be noted that the representation given here is the most general possible case for the physics considered: scattering between all (g, g') pairs is accounted for, as are all potential group delayed and prompt neutron sources. In practice, particularly with $G > 2$, this is not necessary: neutrons cannot be born in the thermal group, and outside of the thermal upscattering, upscattering can frequently be neglected. For these reasons, the admittance matrix $\mathbf{Y}_{i,j,k}^L(t, e_{MRN_{i,j,k}}^T)$

is usually significantly more sparse than shown here.

Additionally, it should be noted that by an implicitly made assumption, fissionable materials are present in the represented nodes. In practice, many of the boundary nodes tend to be homogenized reflectors (structural materials and coolant, but no fuel), and therefore cannot contain precursor nuclei or fission heat rates. When representing such nodes, it makes sense to reduce the number of unknowns in the node, by dropping the elements and bonds that represent the precursors, and, depending on the feedback model, sometimes the temperature as well (without phase change, coolants are much less sensitive to temperature changes than fuel Doppler broadening is).

In 2D problems, a buckling term may be added to the absorption cross section; it does not change the representation, besides this modification.

This concludes the summary of the bond graph representation of multidimensional, multigroup neutron diffusion, which is the predominant model for spatial kinetics full core analysis of nuclear reactors. This representation is successfully tested, in detail, in chapters 6 and 7.

Lastly, bond graph representations for the P_n and S_n discretizations are briefly discussed.

3.4 Spherical Harmonics and Discrete Ordinates Methods

There do not appear to be any examples in literature of the use of bond graphs for discretized radiative transport analysis of neutrons or other elementary particles. For completeness, bond graph representations of basic, uncoupled FV-discretized spherical harmonics and discrete ordinates discretizations of the NTE and DNPEs are discussed here; this section may be skipped without loss of continuity.

Equations (2.64) and (2.66) in subsection 2.1.1.6 give the FV-discretized spherical harmonics equations with no thermal feedback. After dropping the elements' time dependencies (for brevity; it could easily be added back by modulating the resistive elements), and indexing the precursors with simple m indices (instead of (m, j_f)), the equations will be represented here as-is.

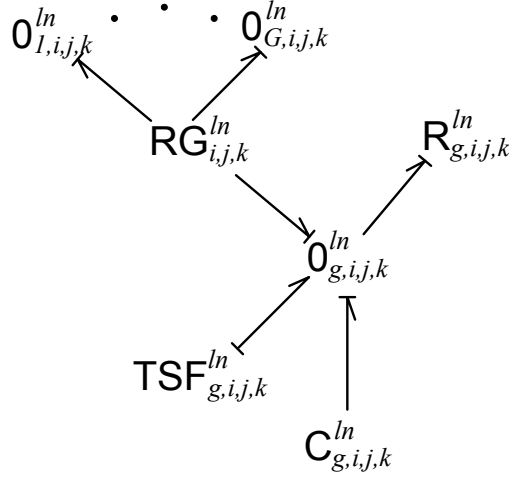
Both equations are statements of conservation of quantity: a neutron number moment, and a number of precursors, in a given volume (i, j, k) . The individual contributions to the net rates of change of the conserved quantities are all proportional to the group flux moments or the precursor numbers. From this, it seems natural to treat the neutron number moments as displacements, the group flux moments as efforts and the moment contribution rates (individual terms in the discretized equations) as flows. The precursors are treated in the same way as in section 3.3: the precursor densities are displacements and efforts, the corresponding capacitors' moduli are unity, and the precursor decay and generation rates are flows.

With this convention, we can begin to represent Eq. (2.64). We first consider the local reactions that affect all (l, n) pairs.

For a given moment (l, n) , Eq. (2.65) relates the flux and neutron number moments; this is a capacitor's constituent expression. The net rate of change of the neutron number moment is a balance of flows, which is enforced by a 0-junction. The group-to-group scattering relates all of the groups of the (l, n) moment in the node, and therefore can be enforced by a multiport resistive element (RG, since it has to have G ports). Total reaction rate is another rate set by a dedicated single-port resistor. The external source is a known time-dependent rate contribution, and so can be represented by a TSF element. Together, these local, general (i.e., non-fission) reaction rates are represented by Figure 3.12.

The elements here are:

- $O_{g,i,j,k}^{ln}$: 0-junction which computes the net rate of change of $N_{g,i,j,k}^{ln}(t)$ and conveys it to the flux moment capacitor. It outputs the nodal average group flux moment to all of its other bonds.
- $C_{g,i,j,k}^{ln}$: Group g flux moment (l, n) capacitor.
- $\text{TSF}_{g,i,j,k}^{ln}$: Time-dependent source of flow, which is responsible for the external source term.
- $R_{g,i,j,k}^{ln}$: Total interaction resistor.
- $\text{RG}_{i,j,k}^{ln}$: Group-to-group scattering multiport resistor, with G ports. Only one of these exists in every node for every (l, n) pair; it is shared between groups.


 Figure 3.12: P_n General Local Reactions Bond Graph Representation

The single-port elements' constituent expressions are:

$$C_{g,i,j,k}^{ln} = \frac{V_{ng}}{\Delta V_{i,j,k}}, \quad (3.36a)$$

$$f_{\text{TSF}_{g,i,j,k}^{ln}}(t) = S_{\text{exg},i,j,k}^{ln}(t), \quad (3.36b)$$

$$R_{g,i,j,k}^{ln} = \frac{1}{\Delta V_{i,j,k} \bar{\Sigma}_{tg,i,j,k}}, \quad (3.36c)$$

in which:

$C_{g,i,j,k}^{ln}$ = Capacitance of the flux capacitor $C_{g,i,j,k}^{ln}$. Units: $1/\text{cm}^2 \text{ s}$.

$f_{\text{TSF}_{g,i,j,k}^{ln}}(t)$ = Flow output by $\text{TSF}_{g,i,j,k}^{ln}$. Units: neutrons/s.

$R_{g,i,j,k}^{ln}$ = Resistance of the total interaction resistive element $R_{g,i,j,k}^{ln}$. Units: $1/\text{cm}^2$.

The group-to-group scattering multiport resistive element $\text{RG}_{i,j,k}^{ln}$ relates a vector of group flux

moments (ordered from $g = 1$ to G) to a vector of inscattering rates, by multiplying the group flux moment vector by a constant coefficient admittance matrix $\mathbf{Y}_{i,j,k}^{ln}$, whose row g , column g' element is given by:

$$Y_{i,j,k,g,g'}^{ln} = \Delta V_{i,j,k} \bar{\Sigma}_{s^+gg',i,j,k}^l, \quad (3.37)$$

in which:

$\mathbf{Y}_{i,j,k}^{ln}$ = Constant admittance matrix of $\text{RG}_{i,j,k}^{ln}$.

$Y_{i,j,k,g,g'}^{ln}$ = Row g , column g' element of the $\text{RG}_{i,j,k}^{ln}$ element admittance matrix $\mathbf{Y}_{i,j,k}^{ln}$. Units: cm^2 .

Next, we find a representation for the terms only present in the $l = n = 0$ equation. These are the prompt and delayed neutron source terms, and, similarly to the scattering matrix above, they can be represented by a combined multiport resistive element with $N = G + M$ ports. Precursors are modeled the same way as in section 3.3. The resulting bond graph representation is shown in Figure 3.13.

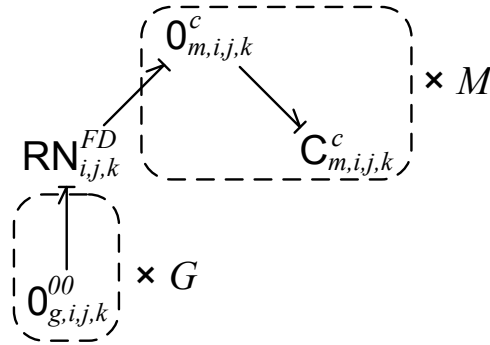


Figure 3.13: P_n Prompt and Delayed Neutron Sources Bond Graph Representation

The elements here are:

$\text{RN}_{i,j,k}^{FD}$: Multiport resistive element that models the prompt and delayed neutron generation rates, and precursor production and decay rates. It has $N = G + M$ ports, which are ordered as follows: group fluxes ($g = 1, \dots, G$), then precursors ($m = 1, \dots, M$). Scalar group fluxes and precursor densities are supplied to it, so it is very similar to the $\text{MRN}_{i,j,k}$ element in section 3.3, although total reaction rates and scattering are modeled elsewhere in the spherical harmonics representation.

$0_{m,i,j,k}^c$: This element is exactly the same as in section 3.3.

$C_{m,i,j,k}^c$: This element is exactly the same as in section 3.3.

The constituent expression of $\text{RN}_{i,j,k}^{FD}$, like that of $\text{RG}_{i,j,k}^{ln}$, is a constant admittance matrix $\mathbf{Y}_{i,j,k}^{FD}$. Using the port notation from section 3.3, its row p , column p' element expression becomes:

$$Y_{i,j,k}^{FD,pp'} = -\Delta V_{i,j,k} \overline{\chi \nu_p \Sigma_{fgg,i,j,k}} \quad \text{with } p = p' \in \mathcal{G}, \quad g = g' = g(p), \quad (3.38a)$$

$$= -\Delta V_{i,j,k} \chi_{dg,m'} \lambda_{m'} \quad \text{with } p \in \mathcal{G}, \quad p' \in \mathcal{M}, \quad g = g(p), \quad m' = m(p'), \quad (3.38b)$$

$$= -\lambda_m \quad \text{with } p = p' \in \mathcal{M}, \quad m = m' = m(p), \quad (3.38c)$$

$$= \overline{\nu_{d,m} \Sigma_{fg',i,j,k}} \quad \text{with } p \in \mathcal{M}, \quad p' \in \mathcal{G}, \quad m = m(p), \quad g' = g(p'), \quad (3.38d)$$

$$= 0 \quad \text{otherwise.} \quad (3.38e)$$

The following notation was used here:

$\mathbf{Y}_{i,j,k}^{FD}$ = Constant admittance matrix of $\text{RN}_{i,j,k}^{FD}$.

$Y_{i,j,k}^{FD,pp'}$ = Row p , column p' element of $\mathbf{Y}_{i,j,k}^{FD}$.

Lastly, the streaming term has to be represented. It is important to note that here, unlike in the other representations in this chapter, while neutron number moments are conserved quantities, a flux moment (l, n) may affect the rate of change of another moment without directly adjusting $N_{g,i,j,k}^{ln}(t)$. Therefore, active bonds will be used, to convey information without directly conveying neutrons.

A single streaming term, \mathcal{I}_l^n , will be represented. Using a transformer to flip the sign on the effort from $0_{g,i,j,k}^{l+1,n}$, a 1-junction can compute the difference in flux moments (efforts) that can then be supplied to a resistor, which evaluates the \mathcal{I}_l^n term's contribution to $dN_{g,i,j,k}^{ln}(t)/dt$. Note, that while \mathcal{I}_l^n cannot be zero, other streaming term coefficients can be; the corresponding block of elements and bonds should only be present if the term is nonzero, otherwise an infinite resistance will result, which can crash the bond graph processing code. The bond graph representation of the \mathcal{I}_l^n streaming term is show in Figure 3.14.

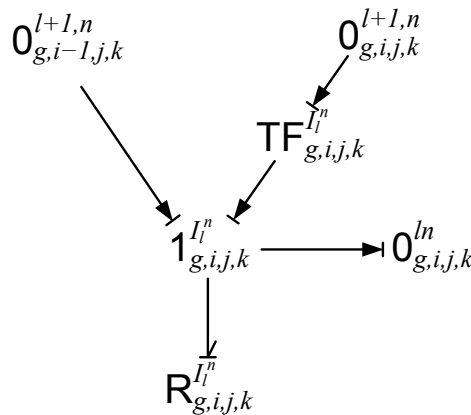


Figure 3.14: $P_n \mathcal{I}_l^n$ Streaming Term Bond Graph Representation

The elements here are:

- $1_{g,i,j,k}^{\mathcal{I}_l^n}$: 1-junction that computes the difference in flux moments supplied to the resistor, to evaluate these moments' contributions to the \mathcal{I}_l^n streaming term. Only one bond connected to it is full, the rest are active.
- $\text{TF}_{g,i,j,k}^{\mathcal{I}_l^n}$: Transformer element for flipping the sign on the effort supplied to the $1_{g,i,j,k}^{\mathcal{I}_l^n}$ -junction from $0_{g,i,j,k}^{l+1,n}$.
- $R_{g,i,j,k}^{\mathcal{I}_l^n}$: Resistor that evaluates the contribution to $dN_{g,i,j,k}^{ln}(t)/dt$ from the \mathcal{I}_l^n streaming term.

The elements' constituent expressions are:

$$\mu_{\text{TF}_{g,i,j,k}^{\mathcal{I}_l^n}} = -1, \quad (3.39a)$$

$$R_{g,i,j,k}^{\mathcal{I}_l^n} = \frac{1}{A_x \mathcal{I}_l^n}, \quad (3.39b)$$

in which:

$\mu_{\text{TF}_{g,i,j,k}^{\mathcal{I}_l^n}}$ = Modulus of the $\text{TF}_{g,i,j,k}^{\mathcal{I}_l^n}$ element. Dimensionless.

$R_{g,i,j,k}^{\mathcal{I}_l^n}$ = Resistance of the $R_{g,i,j,k}^{\mathcal{I}_l^n}$ element. $1/\text{cm}^2$.

This concludes the bond graph representation of the finite volume multigroup spherical harmonics discretization of the NTE, and the associated DNPEs. Unlike the other representations presented above, this one was only given here for completeness, and as a potential future resource, and was not tested in this work.

The discrete ordinates representation is principally the same as the spherical harmonics one: capacitors set the fluxes on 0-junctions, flow sources model the external neutron sources, and scattering and neutron generation is modeled by RN-elements. The total interaction term is modeled by a 1-port resistor. The streaming term is modeled topologically differently, which is discussed below.

Assuming $\mu_d > 0$, the x -directional streaming term is given by (from Eq. (2.76), subsection 2.1.1.7):

$$\left[\frac{1}{\mu_d A_x} \right]^{-1} \left[\bar{\psi}_{g,i-1,j,k}^d(t) - \bar{\psi}_{g,i,j,k}^d(t) \right], \quad (3.40)$$

which clearly looks very similarly to the advection term in section 3.2; this makes sense, because finite volume discretization with upwind fluxes was used to discretize both equations. The representation used in Figure 3.6 can therefore be repeated here, identically. The new x -directional streaming term representation from the S_n equation is shown in Figure 3.15.

$$\mathbf{0}_{g,i-1,j,k}^d \longrightarrow \mathbf{R}2_{g,i,j,k}^d \longrightarrow \mathbf{0}_{g,i,j,k}^d$$

Figure 3.15: S_n x -directional Streaming Term with $\mu_d > 0$ Bond Graph Representation

The elements here are:

- $0_{g,i,j,k}^d$: 0-junction responsible for computing the net rate of change of the number of neutrons traveling in the $\hat{\Omega}_d$ direction. The effort it outputs is $\bar{\psi}_{g,i,j,k}^d(t)$.
- $R2_{g,i,j,k}^d$: The 2-port resistor responsible for evaluating the rate of streaming from node $(i-1, j, k)$ to (i, j, k) along the direction $\hat{\Omega}_d$.

The flow set by the $R2_{g,i,j,k}^d$ is given by:

$$f_{R2_{g,i,j,k}^d}^{in/out} \left(e_{g,i-1,j,k}^d \right) = \mu_d A_x e_{g,i-1,j,k}^d, \quad (3.41)$$

in which:

- $e_{g,i,j,k}^d$ = Effort set by $0_{g,i,j,k}^d$. Units: neutrons/cm² s.
- $f_{R2_{g,i,j,k}^d}^{in/out} \left(e_{g,i-1,j,k}^d \right)$ = Flow into/out of $R2_{g,i,j,k}^d$, set by this 2-port resistor as a function of $e_{g,i-1,j,k}^d$. This flow is output onto both ports of the $R2_{g,i,j,k}^d$ element. Units: neutrons/s.

This concludes the summary of the bond graph representation techniques developed as part of this project. They do not cover all possible sets of physics and discretizations that can be used to model a reactor multiphysics problem, but they are sufficient for a number of benchmark problems, like the ones solved in chapters 5–7. After the bond graph representation of a problem is developed, the representation must be processed using a bond graph processing code. Automated bond graph processing algorithms are discussed in chapter 4.

Chapter 4

Automated Bond Graph Processing

“Bond graph processing” refers to steps 4–7 of the bond graph process, in which a state derivative vector is constructed from the bond graph representation and integrated.

The bond graph system, besides the topology, contains the constituent expressions, which, for realistic problems, may involve data table lookup functions, and other materials that are hard to abstract for general multiphysics. From this issue arise two different approaches to the implementation of a bond graph processing code, described in section 4.1.

Steps 4–5, and depending on context, sometimes 6, of the bond graph process are the “sorting steps.” They involve the conversion from the bond graph system to the state derivative vector, and are the least studied part of the bond graph formalism. Multiple techniques had to be developed, as part of this research, for the sorting steps; they are described in section 4.2.

Bond graphs are normally used for providing a state derivative vector, to be used for fully coupled transient simulation, but such models are often initialized by steady state ICs. Methods for obtaining these are briefly discussed in section 4.3.

Lastly, section 4.4 briefly discusses the time integrators used in this work for fully coupled time integration.

4.1 Implementation Methods

The proof-of-concept bond graph processing code `BGSolver` v1.01 used an ASCII-file format to describe the bond graph systems supplied to it, called “.BGSD” (Bond Graph System Descriptor) [84]. This format, in the form presented in Ref. [84], did not provide any means of supplying large data sets (e.g., material properties) along with the .BGSD file. However, even if it did, `BGSolver` v1.01 was still a separate MATLAB code, which accepted some inputs, processed the system, and provided some output. It is an example of the Separate Code implementation method, discussed in subsection 4.1.1. The bond graph processing codes listed in subsection 2.3.3 all fall in this category.

Many of the MSFs discussed in subsection 2.2.3 do not function the same way; they are not true “codes” in the strictest sense of the word. Instead, they can be thought of as object-oriented C++ libraries, which provide the tools for a code developer, assuming the code developer wants to accept the meshing, discretization and other restrictions placed on them by the MSF library. A bond graph processing code can be thought of as being implemented the same way: as a bond graph processing object-oriented C++ library. This is the other implementation method: the

Object-Oriented Library method, discussed in subsection 4.1.2. Until this work, there did not exist any bond graph processing libraries implemented this way.

4.1.1 Separate Bond Graph Processing Code

If a bond graph processing code is a completely standalone software, it must have a standardized input file format, or a graphical user interface (GUI) for specifying the bond graph system to process. Both approaches are limiting: in a GUI, it may be feasible to manually place 50–100 bond graph elements on diagram, and connect them, but it would be prohibitively time-consuming to place several thousand elements this way. A standardized ASCII input file may avoid this issue, but it has other problems: that of size (an ASCII file is a much less efficient way of using storage space than storing data in binary form), and more importantly, accompanying data specification is still, generally, not possible, unless the code accepts data in certain specified formats as well. Doing so, however, would likely restrict the code’s generality (data for different physics is generally stored and read in different ways), and thus go against the original goal of a general physics, high-performance bond graph processing code.

If, instead, a bond graph processing code is implemented as a toolbox in MATLAB, or a comparable computational environment, supplying data along with larger BGSs becomes easier. In MATLAB, the `.mat` file format allows the user to store individual arrays in a binary, MATLAB-readable file; if the `.BGSD` file is accompanied with such file, and the `.mat` file is loaded into memory prior to processing the `.BGSD` file, the nonlinear constituent expressions in the `.BGSD` file may reference arrays in the `.mat` file. Additionally, if the bond graph system is generated with a script, or a function, one can bypass the `.BGSD` file step entirely, and convey the bond graph system directly to the bond graph processing code from memory, after formatting it as an array of structs, or a similar array-type format. The accompanying data can also simply remain in memory, or be loaded from a `.mat` file.

`BGSolver` v1.01 did not allow for direct BGS specification, and only worked with `.BGSD` files. However, the new versions of `BGSolver`, described below (subsection 4.2.1), do work with struct-type BGS specifications, and allow external `.mat` files to be loaded; this greatly upscaled the size of the problem that could be addressed.

4.1.2 Bond Graph Processing via Object-Oriented Libraries

If a bond graph processing “code” is implemented as an object-oriented library, its functions should function by taking inputs directly as memory pointers. Such library may define multiple element classes, and the creation of a bond graph system would now involve the construction of an array of element class instances, using the class specifications from the library. Data is then supplied trivially: a class instance may simply take, as an optional parameter, one or more pointers to data objects in memory, and use the data objects’ own reading methods for efficiently accessing this data. The bond connectivity map can be a large $N_b \times 6$ array, where:

N_b = Number of bonds in the system.

6 = The array is $N_b \times 6$, because for each bond, the two connected elements, their ports, the bond’s causality, and the bond’s type, may be specified. The bond’s type and causality can be implemented as enumerated data types.

To implement the elements' constituent expressions in element class instances, either pointers to existing functions (with some additional parameters, which specify how to use these functions) may be supplied, or certain standardized (“off-the-shelf”) functions may already be implemented in the element classes. The user may be allowed to create their own versions of the element class types, by inheriting from the existing element classes, or the abstract “element” class itself. Such versions (i.e., a 2-port resistor used specifically to represent electrical current and resistive heating, called RS in Ref. [86]) may be used for more efficient subsequent specific-physics code development.

One of the new bond graph processing codes, described in subsection 4.2.2, was implemented as a C++ bond graph processing library.

Regardless of the implementation method, the bond graph processing code must, by definition, process the bond graph representation. Techniques for doing so are described in the following section.

4.2 Processing Algorithms

While the bulk of bond graph research has focused on bond graph representation, a substantial amount of research has also been conducted in bond graph automation theory.

Some very early work by Rosenberg, and later, Moultrie, focused on developing automated bond graph system augmentation and state derivative vector formulation techniques by building large (junction, resistive, source) “*structure matrices*”, and assembling the state derivative vector from them [102–104]. The approach became the foundation for ENPORT-6, an important early bond graph processing code, but it was limited to linear, time-invariant (LTI) elements, and so is completely unusable for reactor multiphysics.

An earlier work by Birkett and Roe analyzed the mathematical foundations of bond graphs, primarily attempting to find tools from other branches of mathematics (graph and matroid theory) that may be useful for the use of bond graphs [105–108]. More recent work by Lamb, Woodall and Asher focused on proving a number of results about sortability, uniqueness and causality assignment procedures for bond graphs, as well as theorizing about bond graph equivalences that may be used to reduce a bond graph system prior to processing it [109–111].

Finally, more recently, when symbolic equation solution became more feasible, many bond graph texts resorted to simply calling for symbolic solution of the algebraic equations generated in step 4 of the bond graph process [112]. This approach is not very scalable, because symbolic solving is a very inefficient, poorly-scaling procedure, but it is capable of addressing all other desired features of the bond graph representations developed in Ref. [84] and chapter 3, and its use was warranted for the initial testing of the reactor bond graph representations. Automated bond graph processing via symbolic sorting is discussed in subsection 4.2.1.

As expected, while symbolically-sorted algorithms did resolve and correctly simulate the representations developed in chapter 3, they could not do so efficiently: the symbolic sorting was a major bottleneck. To address this issue, a new sorting algorithm, which combines the performance of early LTI approaches with the flexibility of the symbolic sorting was developed. It was named “*Matrix-Based Sorting*,” and is described in subsection 4.2.2.

4.2.1 Symbolic Sorting

As implemented in this work, a symbolically-sorted bond graph processing code works through the following steps:

1. It loops through the elements of the model, formulating the algebraic equations for each one, according to Table 2.3. It eventually arrives at a residual function:

$$\vec{\mathbf{F}}(t, \vec{\mathbf{b}}, \vec{\mathbf{x}}) = \vec{\mathbf{0}} \in \mathbb{R}^{N_{bv}}, \quad (4.1)$$

in which:

N_{bv} = Number of bond variables in the system.

$\vec{\mathbf{b}}$ = Vector of all bond variables in the problem.

$\vec{\mathbf{x}}$ = State vector.

$\vec{\mathbf{F}}(t, \vec{\mathbf{b}}, \vec{\mathbf{x}})$ = N_{bv} -dimensional residual function, which, when set equal to zero, constitutes the output of step 4 of the bond graph process.

2. A symbolic solver is used on $\vec{\mathbf{F}}(t, \vec{\mathbf{b}}, \vec{\mathbf{x}})$, which, if successful, yields the functional form of $\vec{\mathbf{b}}(t, \vec{\mathbf{x}})$. This solution, assuming a correctly functioning algebraic symbolic solver and a fully causal bond graph with correctly formulated $\vec{\mathbf{F}}(t, \vec{\mathbf{b}}, \vec{\mathbf{x}})$, is guaranteed to exist and be unique. $\vec{\mathbf{b}}(t, \vec{\mathbf{x}})$ constitutes the output of step 5 of the bond graph process.
3. The bond variables which, by Table 2.3, correspond to the state derivative vector elements (the flows delivered to capacitors and the efforts delivered to inertial elements), are extracted, and their symbolic expressions are formed into a new vector $\dot{\vec{\mathbf{x}}}(t, \vec{\mathbf{x}})$:

$\dot{\vec{\mathbf{x}}}(t, \vec{\mathbf{x}})$ = State derivative vector, which consists of the bond variables delivered to the storage elements, in terms of time and the state vector.

$\dot{\vec{\mathbf{x}}}(t, \vec{\mathbf{x}})$ constitutes the output of step 6 of the bond graph process.

4. $\dot{\vec{\mathbf{x}}}(t, \vec{\mathbf{x}})$ can be integrated in time, which constitutes state 7 of the bond graph process. Some methods for doing so are discussed in section 4.4.

This algorithm is fundamentally unchanged from the one presented in Ref. [84].

Two new important element types, RN and MRN, had to be added to the list of elements that `BGSolver` had to process, in order to successfully simulate the representations in chapter 3. A new version of `BGSolver`, version 1.03, was developed to do so. It made the following functional additions:

1. Overall performance improvement. By frequently vectorizing and optimizing the MATLAB code, and by implementing parts of the code in a compiled `.mex` (MATLAB Executable) format, `BGSolver` v1.03 resulted in being significantly (up to 2 orders of magnitude) faster than `BGSolver` v1.01. The performance difference was particularly sensible in state derivative evaluation, which is what the time integrators operate through.

2. RN and MRN element support, instead of the limited R2 and MR2 elements, was added. Having arbitrary multiport resistors allows the local coupling of more than two PDEs, which was necessary for the multiphysics discussed in chapter 3.
3. Jacobian sparsity pattern construction was added. This allowed support for several implicit time integrators, that without it had to assume the Jacobian was a fully dense matrix, which made them prohibitively expensive even for small problems.
4. Physics level specification for storage elements was added, to support split operator time integration. One of the objectives of this work was to prove or reject the superiority of superlinear fully coupled time integrators, and doing so by simulating the same problem using otherwise identical codes was the chosen approach. This split operator approach is optional; by default, `BGSolver` v1.03 still uses a fully coupled time integrator.
5. Processing restart capability. `BGSolver` v1.03 is able to save the results of sorting, and restart it multiple times, with varying ICs, time integrator type, and other parameters. Because, as discussed above, sorting is a very costly procedure, this capability greatly accelerated the work involved in integrating the same problem using an array of time integrators and physics splitting options.
6. State variable non-negativity and absolute tolerance specification. These parameters may be used by some (usually adaptive) time integrators.

Like all other bond graph processing codes developed for this work, `BGSolver` v1.03 is able to process only fully causal bond graphs. It supports all of the elements in Table 2.3, with constant coefficient, symbolic or numeric (MATLAB function) expressions. It supports a `.BGSD` v1.03, or a struct-based input, and allows for `.mat` data files to be associated with its inputs.

`BGSolver` v1.03 was released as open source software [113]. Reference [114] serves as a short introduction to it.

`BGSolver` v1.03 was tested with the bond graph representations developed in sections 3.1, 3.2 and 3.3. It was able to successfully process and integrate all of them, with up to about 500 unknowns (about 5000 bond variables), but as expected, symbolic sorting, despite the optimizations described above, remained a major bottleneck, preventing the processing of fine-mesh spatial kinetics problems. The reliance on MATLAB’s symbolic engine was also a restriction in itself: it was not possible to readily convert the code to another, compiled and higher-performance language, such as C++, because this engine is not currently available as a library in other languages, and developing a new symbolic engine would be a massive undertaking.

To address these two issues, the Matrix-Based Sorting algorithm (MBS) was developed, described in the following subsection.

4.2.2 Matrix-Based Sorting

The matrix-based sorting algorithm aims to arrive at a state derivative vector by only building and multiplying large sparse matrices — a procedure that can be easily parallelized, implemented in a compiled language, and that is expected to be significantly faster than symbolic sorting. The MBS algorithm developed here is a first iteration of it, which was developed with the intent of being high-performance over high-flexibility. For this reason, the algorithm has several restrictions:

1. The algorithm only supports the following element types: SE, SF, I, C, R, 1, 0, TSE, TSF, MR, RN, MRN. Symbolic expression types, modulated storage elements, and bond- and state-modulated sources are not supported. State modulation of the resistive elements is also no longer supported.
2. All non-junction elements may only be connected to junctions: for example, a storage element cannot be directly connected to a multiport resistor. In representations where this is required, a trivial 2-port junction may be placed between the resistor and the storage element; this is where some of the trivial 2-port junctions in section 3.3 came from.
3. Signal and active bonds are supported, but may only originate at junctions or at multiport resistive elements.

This reduction in flexibility was intentional: they were required for the version of MBS algorithm presented below. It does not in any way affect the representations in chapter 3.

The following nomenclature is used in this subsection:

N_{bv} = Number of bond variables, including signal bonds.

N_x = Number of state variables.

N_{el} = Number of elements.

N_{nl} = Number of numeric layers in the system. Numeric layers are defined below.

i = Bond variable ID.

l = Numeric layer ID of a bond variable. Numeric layers are defined below.

b_i = Bond variable with bond variable ID i .

\vec{l} = Vector of length N_{bv} , with l_i being the numeric layer ID of b_i . Defined below.

\vec{i}_x = Vector of BV IDs which correspond to the state derivative BVs.

\vec{i}_l^F = Vector of all layer l feed BV IDs in the system. Defined below.

\vec{i}_l^J = Vector of all layer l junction BV IDs in the system. Defined below.

\vec{i}_l^C = Vector of all combined layer l BV IDs in the system. Defined below.

$\vec{\mathbf{b}}_l^F$ = Numeric layer l feed BV vector. Defined below.

$\vec{\mathbf{b}}_l^J$ = Numeric layer l junction BV vector. Defined below.

$\vec{\mathbf{b}}_l^C$ = Numeric layer l combined BV vector. Defined below.

$\vec{\mathbf{b}}_l^R$ = Numeric layer l resistive input vector. Defined below.

$\mathbf{M}_{i,j}$ = Row i , column j element of matrix \mathbf{M} .

Additionally, recall from section 2.3, that statements like “element j ” or “BV i ” mean “element with element ID j ” (all elements are indexed) and “bond variable with bond variable ID i .”

Two terms, coined for this algorithm, are defined:

- “*Feed variables*” are defined as bond variables delivered to the junction structure from source, storage and resistive elements.

- “*Expanded vectors*” here mean the following: a vector of a group of BVs is expanded (or “in expanded format”) if it is of length N_{bv} , but all of the entries that do not correspond to this group of BVs are zero. For example, if there are a total of 4 bond variables in a system, and a group of BVs \mathcal{G} corresponds to $i = \{1, 3\}$, the corresponding BV vector in expanded format is:

$$\vec{\mathbf{b}}_{\mathcal{G}} = \begin{bmatrix} b_1 \\ 0 \\ b_3 \\ 0 \end{bmatrix}, \quad (4.2)$$

in which:

$\vec{\mathbf{b}}_{\mathcal{G}}$ = Expanded vector of BVs corresponding to group \mathcal{G} .

- “*Expanded matrices*” are similar: when, for example, a matrix relates a group of BVs in group \mathcal{G} to a group of BVs in group \mathcal{H} (like the junction structure matrices, described below, do), it is convenient to place them in the expanded format: the expanded vector of group \mathcal{G} BVs, when multiplied by the matrix in the expanded format, produces the expanded vector of \mathcal{H} BVs. This is very effective for indexing, and, assuming sparse array structures are used, does not significantly increase the storage space required. However, note, that it is still more beneficial, at least in MATLAB, to treat the expanded vectors as full vectors: this will take less storage space than constructing a sparse vector, due to how the sparse array objects are arranged in memory.

The algorithm begins by forming element connectivity matrices (ECMs). An element connectivity matrix is a sparse $N_{el} \times N_{el}$ matrix, defined as follows:

- \mathbf{ECM}^t = Element connectivity matrix of type t . t may be effort e , flow f or bond b variables. All three matrices, as well as the transpose of \mathbf{ECM}^b , are built.
- $\mathbf{ECM}_{j_{to}, j_{from}}^t$ = BV ID i , if bond variable i of type t is delivered from element j_{from} to element j_{to} . 0 otherwise.

The ECMs may be used as follows: if we need to identify the BVs output by a group of elements with element IDs \vec{j} , we simply obtain all of the nonzeros in the columns indexed by \vec{j} of the \mathbf{ECM}^b matrix. The transpose can be used similarly to identify the BV IDs of the BVs delivered to a group of elements. The transpose is used here, instead of rowwise nonzero identification operation, because of the column-major manner in which MATLAB stores its matrices. \mathbf{ECM}^e and \mathbf{ECM}^f are used when constructing \mathbf{ECM}^b .

The ECMs are first used to construct three other matrices: the junction structure matrix (JSM), the junction structure dependency matrix (JSDM) and the resistor input/output dependency matrix. They are defined as follows:

- JSM** = The JSM is a sparse $N_{bv} \times N_{bv}$ double precision matrix, in expanded format. The product of the JSM with a vector of feed variables in expanded format (the non-feed elements of the vector may be zero, they are not multiplied) produces a vector of all non-feed bond variables in the system. The notion of the JSM was first introduced by Rosenberg, and he gives procedures to construct it [102]; here it is converted into the expanded format. Note, that this only works because all junctions' equations are linear and constant coefficient expressions.
- JSDM** = The boolean sparsity pattern of the JSM. Some sources (e.g., Ref. [115]) call this the “graph” of the JSM.
- RIODM** = The resistor input/output dependency matrix. $\mathbf{RIODM}_{i_{out}, i_{in}} = 1$ if BV i_{out} is output by a resistor to which BV i_{in} is delivered, and 0 otherwise.

Next we consider the notion of the numeric layer, first used in Ref. [84], but modified here for MBS. It works by assigning a layer ID to every bond variable in the problem. The idea is: a bond variable with layer ID $l = 1$ can be evaluated by knowing only the state vector and time; bond variables with layer ID $l > 1$ can be evaluated by knowing the state vector, time, and all BVs on the layers below l . The numeric layers are used to evaluate the variables sequentially, in blocks.

The layer IDs are assigned as follows:

1. The columns of \mathbf{ECM}^b that correspond to source and storage elements are first looked at, and their nonzeros recorded as \vec{i}_1^F , the layer 1 feed BVs. Layer 1 feed BVs that can be evaluated based on t and \vec{x} only (by the above assumptions, the storage and source elements' expressions cannot depend on anything else). \vec{i}_1^F are also the only BVs on layer 1, so they may be referred to as simply “layer 1 BVs” instead.
2. If a BV can be evaluated by multiplying the JSM by the vector produced in step 1, it is considered to be on layer 2. Because it is output by the junctions, it is referred to as a “layer 2 junction BV,” and the vector of all such BV IDs is denoted \vec{i}_2^J . This can be checked by examining the rows of **JSDM**: the rows in which all nonzero entries are in the columns indexed by \vec{i}_1^F are the ones that correspond to layer 2 junction BVs. The index vector of these rows is therefore \vec{i}_2^J . These layer 2 junction BVs are the BVs delivered to the feed elements (sources, storage and resistive elements) by the junction structure (junction output BVs), and the ones delivered from one junction element to another (interjunction BVs). At this point the layer 1 feed BV IDs \vec{i}_1^F and the layer 2 junction BV IDs \vec{i}_2^J can be combined into a “combined layer 2 BV ID vector” \vec{i}_2^C . This vector combines all of the known BVs after evaluating the state and storage element outputs, and feeding them through the junction structure.
3. If a BV can be evaluated by supplying the layer 2 combined BV vector to the resistive elements, and having them output it, this BV is considered to be a layer 2 (resistive) feed BV. To check whether or not a BV is a resistive BV, \mathbf{ECM}^b is once again used; the nonzero elements in the columns which belong to the resistive elements are the resistive feed BVs. To check if a resistive feed BV is on layer 2, its row in **RIODM** must be examined. If the row's nonzero elements are all in the columns of **RIODM** indexed by \vec{i}_2^C , this resistive feed BV is on layer 2; otherwise, it is on a higher layer, because additional junction BVs are required. Using this procedure for all resistive feed BVs, the layer 2 resistive feed BV ID vector \vec{i}_2^F is constructed.

4. At this point, the layer 1 feed, layer 2 junction, layer 2 feed and combined layer 2 BV IDs are known. Steps 2 and 3 can be repeated for layer 3, 4, and so forth, until all BVs are assigned a numeric layer. The combined BV ID vectors, in particular, grow with every layer, until they contain all BV IDs in the system.

In practice, there are usually only 2–3 numeric layers in a discretized P(I)DE problem, unless an unusually wide spatial stencil is used. When finished, N_{nl} may be assigned; it is the highest l in the problem.

After all BVs have been assigned a layer ID, and all \vec{i}_l^F , \vec{i}_l^J and \vec{i}_l^C vectors have been constructed, it can be convenient to build an \vec{l} vector of length N_{bv} , where its element l_i is the layer ID of BV i .

In the following procedure, for convenience, the BV vectors are kept in expanded format. Additional steps will later be identified to trim the vectors to their minimum required lengths.

The ECMs are first used to build the source and storage structure function:

$\vec{F}_{SS}(t, \vec{x}) =$ A vector function, which outputs all of the capacitors' efforts, all of the inertial elements' flows, and all of the bond variables output by the sources, in the order specified by \vec{i}_{SS} . For now, we assume the output to be in the expanded format.

By the definition above:

$$\vec{b}_1^F = \vec{F}_{SS}(t, \vec{x}). \quad (4.3)$$

Once the layer IDs have been assigned, the procedure can loop through layers, building supporting matrices and vector functions for the state derivative evaluation, as follows :

1. Nothing is done for layer 1: $\vec{F}_{SS}(t, \vec{x})$ already exists.
2. A matrix in expanded format is built, which consists of the part of the JSM that outputs the layer 2 junction BV vector \vec{b}_2^C , which is indexed by \vec{i}_2^C . Note, that the nonzeros in this output are not only the layer 2 junction BVs, but also the layer 1 feed BVs \vec{b}_1^F . We denote this matrix \mathbf{JSM}^2 .
3. Another matrix is built, this time not in expanded format (but with N_{bv} columns). It is intended to multiply the layer 2 combined BV vector \vec{b}_2^C , and output only the BVs required by the resistors to produce the layer 2 resistive feed BVs, i.e., the vector \vec{b}_2^R . We denote this matrix \mathbf{RISM}^2 , the layer 2 resistive input structure matrix. So far, we have the following:

$$\vec{b}_1^F = \vec{F}_{SS}(t, \vec{x}), \quad (4.4a)$$

$$\vec{b}_2^C = \mathbf{JSM}^2 \cdot \vec{b}_1^F, \quad (4.4b)$$

$$\vec{b}_2^R = \mathbf{RISM}^2 \cdot \vec{b}_2^C = \mathbf{RISM}^2 \cdot \mathbf{JSM}^2 \cdot \vec{F}_{SS}(t, \vec{x}). \quad (4.4c)$$

4. A vector function is built that works as follows: it takes time and \vec{b}_2^R as input, and outputs, in expanded format, the layer 2 resistive feed bond variables. We denote this function $\vec{F}_R^2(t, \vec{b}_2^R)$. We now have, additionally:

$$\vec{b}_2^F = \vec{b}_2^C + \vec{F}_R^2(t, \vec{b}_2^R). \quad (4.5)$$

5. We continue looping through layers until each layer's \mathbf{JSM}^l , \mathbf{RISM}^l and $\vec{F}_R^l(t, \vec{\mathbf{b}}_l^R)$ have been constructed. Eventually, the top layer's junction vector $\vec{\mathbf{b}}_{N_{nl}}^C$, if evaluated (note, we are just constructing functions and matrices, and not evaluating anything, yet) should contain all of the bond variables. A final extraction matrix \mathbf{FEM}^x , which extracts the state derivative BVs, is constructed as an $N_x \times N_{bv}$ matrix, with a 1 in every row, in the column that corresponds to a state derivative BV.

Assuming, for example, a 3-layer system (most of them are), the output therefore is given by:

$$\dot{\vec{\mathbf{x}}}(t, \vec{\mathbf{x}}) = \mathbf{FEM}^x \cdot \mathbf{JSM}^3 \cdot \vec{\mathbf{b}}_2^F, \quad (4.6a)$$

$$\vec{\mathbf{b}}_2^F = \vec{\mathbf{b}}_2^C + \vec{F}_R^2(t, \vec{\mathbf{b}}_2^R), \quad (4.6b)$$

$$\vec{\mathbf{b}}_2^R = \mathbf{RISM}^2 \cdot \vec{\mathbf{b}}_2^C, \quad (4.6c)$$

$$\vec{\mathbf{b}}_2^C = \mathbf{JSM}^2 \cdot \vec{F}_{SS}(t, \vec{\mathbf{x}}). \quad (4.6d)$$

The state derivative evaluation is therefore a multistage process, with two matrix multiplications, a potentially nonlinear vector function evaluation, and a sum of two vectors, on every layer, except for the first or the last.

Again, expanded vectors and matrices were used here; their use significantly simplified the notation. In practice, not every bond variable needs to be evaluated to evaluate the state derivative vector, and so all of the sorting objects described above may be trimmed, dropping the unneeded bond variables from the expanded format. Briefly, this trimming procedure requires the following (assuming the highest state derivative BV's l is N_{nl} , and using \mathbf{JSDM} and \mathbf{RIODM} to check for dependencies):

1. Starting with the top layer, identify the parts of $\vec{\mathbf{b}}_{N_{nl}}^J$ (i.e., the junction BVs just evaluated) that are not state derivative variables. Mark them as unneeded.
2. Proceeding to the layer below, identify the feed BVs computed solely to evaluate the unneeded combined BVs on the upper layer(s). Mark these feed BVs as also unneeded.
3. At the same layer, identify the combined BVs computed solely to evaluate the unneeded feed BVs, or the junction BVs on the upper layer(s). Mark these combined BVs as unneeded.
4. Repeat steps 2 and 3 for every lower layer, until coming to layer 1. Using the resulting vector of unneeded BVs, trim and rearrange \mathbf{RISM}^l , \mathbf{JSM}^l and $\vec{F}_R^l(t, \vec{\mathbf{b}}_l^R)$ to only input and output the needed BVs.

In principle, the two steps can be combined: it's possible to, from the start, identify which BVs are needed for $\dot{\vec{\mathbf{x}}}(t, \vec{\mathbf{x}})$ evaluation and build the sorting objects accordingly. The algorithm, as given here, is far clearer, but it may be somewhat less efficient; the more efficient version without the expanded vectors was implemented in the `BGSolver` v2.0 and `Larch` codes, mentioned below.

This concludes the summary of the MBS algorithm. Its performance is question of software engineering: the functions used to construct, multiply and perform columnwise/rowwise indexing operations on the large sparse matrices have to be fast, otherwise the algorithm, despite being

fully numeric, will not perform well. Fortunately, both MATLAB and C++ possess the tools to do so.

The MBS algorithm was implemented in two versions. First, a new version of `BGSolver`, still MATLAB, but no longer symbolic sorting-based, was developed using MBS. This version was titled `BGSolver v2.0`. Next, to achieve greater scalability, the algorithm was implemented in a C++ bond graph processing library, titled `Larch` (after a tree). The two codes were successfully tested using fine mesh spatial kinetics problems in chapters 6 and 7, respectively.

Sorting was the major bottleneck associated with the automation of bond graph processing, solved by the MBS algorithm. Other relatively minor issues include steady state problem solutions, and time integration, discussed in the following sections.

4.3 Use of Bond Graphs for Steady State Problems

The bond graph process outputs, and potentially integrates, $\dot{\vec{x}}(t, \vec{x})$. Finding the steady state of this state derivative is fairly simple if the problem has a forcing function, and nontrivial if it's not. The two cases are discussed in the following subsections.

4.3.1 Problems with Forcing Functions

At steady state, the following is true, by definition:

$$\dot{\vec{x}}(0, \vec{x}^0) = \vec{0}. \quad (4.7)$$

If the discretized problem has a nonzero forcing function, like an external neutron source, then Eq. (4.7) can be solved using a nonlinear solver (e.g., Newton's method) for \vec{x}^0 , which is the steady state of the system. Newton's method will not necessarily converge, particularly if property discontinuities or undefined regions are present; to get it to converge, it can be dampened. Section 4.4 discusses a method for doing so.

If the discretized problem, however, has no forcing function, then, as was discussed in subsection 2.1.1.3, it is an eigenproblem. Eigenproblems cannot be solved by solving Eq. (4.7) (a solution does not, generally, exist); instead it must be addressed as an eigenproblem. A simple, somewhat brute force method for doing so, is discussed in subsection 4.3.2.

4.3.2 Eigenvalue Problems

If a problem is expected to behave as an eigenproblem for its steady state search (i.e., no forcing functions are present), then an eigenvalue parameter must be specified. It may be an additional dummy capacitor connected to a 0-junction, with a unity modulus, and the junction disconnected from anything else. The effort on this bond is then the equal to the capacitor's displacement (the eigenvalue), and it may modulate all of the resistive elements where the eigenvalue would normally be used (in a neutron problem, the prompt neutron and delayed neutron precursor production resistors).

This introduces an unknown — the eigenvalue. The rate of change of this parameter is then zero (the junction is disconnected from anything else, so its flow is zero), and it does not close the system. Instead, another closing function, like the total reactor power, or an initial power density, may be specified. This equation's residual (something like $P(0, \vec{x}^0) - P^0$) can then be set

to replace the term in $\dot{\vec{x}}$ that corresponds to the dummy flow on the eigenvalue's capacitor, and the resulting (adjusted) state derivative vector now becomes a solvable vector residual (assuming a steady state for such a system exists, which it may not).

BGSolver v1.03 was developed without this capability, and its roundabout search for eigenvalue is described chapters 5 and 6. **BGSolver** v2.0 and **Larch** were developed with this capability, and so they are able to, directly, find the problem's eigenvalue and eigenfunction, prior to integrating it in time.

Time integration methods used by all 3 codes developed in this work are described in the following section.

4.4 Time Integration

Multiphysics problems can be expected to be poorly-scaled, and stiff. Even the point kinetics equations alone are considered very stiff, due to the wide range of half-lives on the precursor families.

Multistep methods are conventionally recommended for such ODEs. The codes developed in this work support three backward differentiation formula, multistep time integration methods: the 1st, 2nd and 3rd-order BDF methods, given by Eqs. (4.8a), (4.8b) and (4.8c), respectively [116]:

$$\vec{x}^{n+1} = \vec{x}^n + \Delta t \dot{\vec{x}}^{n+1}, \quad (4.8a)$$

$$\vec{x}^{n+1} = -\frac{1}{3}\vec{x}^{n-1} + \frac{4}{3}\vec{x}^n - \frac{2}{3}\Delta t \dot{\vec{x}}^{n+1}, \quad (4.8b)$$

$$\vec{x}^{n+1} = \frac{2}{11}\vec{x}^{n-2} - \frac{9}{11}\vec{x}^{n-1} + \frac{18}{11}\vec{x}^n + \frac{6}{11}\Delta t \dot{\vec{x}}^{n+1}. \quad (4.8c)$$

Here the notation is:

Δt = Time step. Fixed time steps are assumed. Units: s.

\vec{x}^n = State vector evaluated at time step n .

$\dot{\vec{x}}^{n+1}$ = State derivative vector evaluated at time step $n + 1$.

These multistep methods are initialized by assuming the pre-initial states to be the same as the initial condition; this strictly only works for initially steady state problems, which RIAs are.

To take a single BDF time step, a nonlinear solver is required. Newton's method, or its variations (e.g., Jacobian-Free Newton Krylov methods [23]), are normally used. Jacobians of the state derivative vector are therefore required; Trilinos has built-in finite difference methods for Jacobian estimation; MATLAB can use a built-in, undocumented procedure called `numjac`, which is based on an algorithm by Salane [117]. Jacobian sparsity pattern, which **BGSolver** v1.03 and later versions (as well as **Larch**) build, greatly accelerates this step.

To account for property discontinuities, and oversteps between Newton iterations, it may be useful to sometimes seed regular Newton's method with Picard iterations, which do not update the Jacobian between iterations [96]. These steps, as well as regular Newton's method steps, may also be dampened. Overall, a Picard iteration-initiated, potentially-dampened nonlinear Newton-Raphson solver was developed, with the following parameters:

N_P = Number of Picard iterations to seed the nonlinear solver with. In regular Newton's method, $N_P = 0$.

λ_P = Picard iteration damping factor. In regular Newton's method, $\lambda_P = 1.0$.

λ_{NR} = Newton-Raphson iterations (which do update the Jacobian) damping factor. Regular Newton-Raphson method uses $\lambda_{NR} = 1.0$.

This nonlinear solver was implemented in `BGSolver v1.03` and `BGSolver v2.0`. In practice, regular Newton's method, assuming a direct solver was used to solve the Newton iterations, was sufficient for most problems' steady state solutions and transient time stepping.

This concludes the summary of the automated bond graph processing tools developed for this project. The bond graph representation techniques from chapter 3, and the automated bond graph processing tools from this chapter, were extensively tested using benchmark problems. The first, simplest benchmark problem was a PWR PKE problem with feedback, presented in chapter 5.

Chapter 5

Neutron Point Kinetics with Feedback Problem

To test the newly-developed bond graph processing code, `BGSolver` v1.03, a coupled reactor kinetics problem was required. A neutron point kinetics benchmark problem, based on a typical PWR RELAP5-3D deck, was constructed, with the following objectives:

1. To verify that `BGSolver` v1.03 functions as intended, including its support of the RN element. `BGSolver` v1.01, developed as part of my S.M. thesis, was only tested on very small problems of about 30 unknowns. `BGSolver` v1.03 must work on problems with several hundred unknowns to be able to adequately process spatial kinetics problems.
2. To verify that the bond graph representations of neutron point kinetics with feedback, as well as the systems-level thermal hydraulics (sections 3.1 and 3.2, respectively), correctly represents the underlying physics (subsections 2.1.5.3 and 2.1.4.3, respectively).
3. To study the time convergence properties of superlinear fully coupled time integrators, and to compare them with split operator time integrators.
4. To compare the resulting solution with a RELAP5-3D solution.

Most of the results presented in this chapter were published in Ref. [114].

Section 5.1 defines the benchmark problem's underlying physics, geometry, material properties and initial conditions. Sections 5.2 and 5.3 discuss the steady state and transient solutions obtained using `BGSolver` v1.03 and RELAP5-3D v2.42. The findings are summarized in section 5.4.

5.1 Benchmark Problem Definition

The benchmark problem solved here is a modified version of the typical PWR core from RELAP5-3D sample decks collection (Ref. [52, `typpwr.i` deck distributed with the code]). It consists of a set of fuel rods (fuel, gap and clad materials) which represent the entire core, and are axially divided into 6 axial levels. The rods are cooled by a single effective channel, the properties of which represent the core average thermohydraulic properties. The bulk flow is single-phase, with a fixed inlet mass flow rate and core average operating pressure.

The reactor power is generated in both the fuel and the coolant directly, and the power distribution is modeled by fixed power fractions, which add up to 1. The fixed power shape justifies the use of a point kinetics neutron model. There are 6 delayed neutron precursor families, and all point kinetics parameters, except for the reactivity, are constant. Thermal feedback is described by the (simplest) spatially distributed separable linear reactivity feedback model. Only the fuel and moderator temperature feedback is considered; the effects of moderator density are assumed to be fully covered by the moderator temperature feedback, same as in `typpwr.i`. The physics modeled in the benchmark, and their bond graph representations, are described in sections 3.1 and 3.2.

The core geometry is described by Tables 5.1 and 5.3. The source fractions and feedback coefficients are given in Table 5.1, and the precursor family properties in Table 5.2. Inlet parameters, including the average core pressure at which the water properties are evaluated, are also given in Table 5.3.

Table 5.1: PWR PKE Benchmark Problem Axial Parameters

i	Δz_i [m]	N_{pi}	S_i	γ_i	α_{fi} [K ⁻¹]	α_{wi} [K ⁻¹]
1	0.7639	36 552	0.141211	0.003769	-5.35743×10^{-6}	-9.5355×10^{-5}
2	0.6655	36 552	0.204599	0.005461	-8.73639×10^{-6}	-15.5376×10^{-5}
3	0.6655	36 552	0.208339	0.005561	-7.80390×10^{-6}	-13.8879×10^{-5}
4	0.6655	36 552	0.206780	0.005520	-7.29846×10^{-6}	-12.9870×10^{-5}
5	0.6655	36 552	0.174414	0.004656	-7.80741×10^{-6}	-13.8879×10^{-5}
6	0.6508	36 552	0.038668	0.001032	-2.46519×10^{-6}	-4.3875×10^{-5}

Table 5.2: PWR PKE Benchmark Problem Precursor Family Properties

m	λ_m [s ⁻¹]	β_m
1	0.0127	0.0002470
2	0.1150	0.0012220
3	1.4000	0.0008320
4	0.0317	0.0013845
5	0.3110	0.0026455
6	3.8700	0.0001690

The benchmark problem is solved using `BGSolver` v1.03 and RELAP5-3D v2.42. In the `BGSolver` v1.03 model, the IAPWS-IF97 water properties are used [48]. The RELAP5-3D model uses the IFC-67 water properties [47]. The IAPWS-IF97 properties are an update on the IFC-67 properties.

The Weisman heat transfer correlation was used for the heat transfer correlation in the `BGSolver` v1.03 model [118]. It is defined by a circular tube Nusselt number correlation with a bundle correction factor (here, properties are assumed to be only water temperature-dependent, because a fixed core average pressure is used):

$$\text{Nu} = \psi_{bcf} \text{Nu}_{ct}, \quad (5.1)$$

Table 5.3: PWR PKE Benchmark Problem Additional Parameters

Parameter	Value
r_f [mm]	4.267
Δr_g [mm]	0.6096
Δr_c [mm]	0.57
A_w [m ²]	4.9617
D_{hw} [m]	0.0135
P/D	1.1
Λ [s]	2.18855×10^{-5}
\dot{m} [kg/s]	18 395.15
\bar{p}_{core} [bar]	156.5478
$T_{w,0}$ [K]	550

with the circular tube Nusselt number given by:

$$\text{Nu}_{ct} = 0.023\text{Re}^{0.8}\text{Pr}^{0.333}, \quad (5.2)$$

and the following bundle correction factor:

$$\psi_{bcf} = 1.826P/D - 1.0430. \quad (5.3)$$

Nusselt, Reynolds and Prandtl numbers are defined by:

$$\text{Nu} = \frac{hD_{hw}}{k_w(T_w)}, \quad (5.4a)$$

$$\text{Re} = \frac{\rho_w(T_w) V_w(T_w) D_{hw}}{\mu_w(T_w)}, \quad (5.4b)$$

$$\text{Pr} = \frac{c_{pw}(T_w) \mu_w(T_w)}{k_w(T_w)}. \quad (5.4c)$$

The convection coefficient h is obtained from Eq. (5.4a), using the Nusselt number from Eq. (5.1).

Because the `BGSolver` v1.03 model assumes a constant, uniform mass flow rate, the flow velocity can be evaluated as a function of water temperature, as discussed in section 3.2.

The following notation was used here:

Nu = Nusselt number. Dimensionless.

Re = Reynolds number. Dimensionless.

Pr = Prandtl number. Dimensionless.

Nu_{ct} = Nusselt number for a circular tube. Dimensionless.

ψ_{bcf} = Square bundle correction factor. Dimensionless.

P/D = Bundle pitch-to-diameter ratio. Dimensionless.

D_{hw} = Channel hydraulic diameter. Units: m.

The RELAP5-3D v2.42 model does not specify the HTC explicitly, and is instead computed by RELAP5-3D's heat transfer module. The steady state error between the two HTCs is under 2%. Pressure may vary in the RELAP5-3D model, while in the BGSolver model, the pressure is held constant; because the core stays pressurized, the two configurations are nearly identical.

Gap is approximated as a solid material with a low thermal conductivity. A constant gap thermal conductivity of 1.5258 W/m K and a constant gap volumetric heat capacity of 4.3595 J/m³ K are assumed. The fuel and clad material properties are given in Figure 5.1.

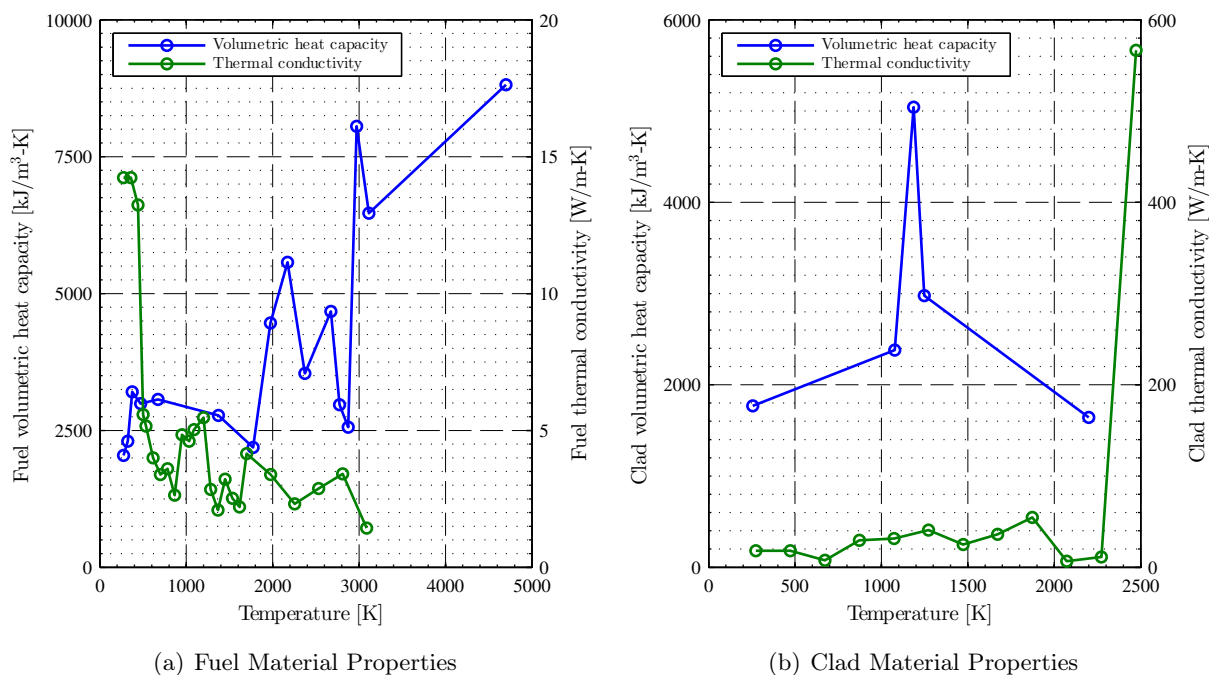


Figure 5.1: PWR PKE Benchmark Problem Material Thermal Properties

These properties were taken directly out of `typ_pwr.i`; they are much less smooth than material properties normally are, but for this problem, this is an advantage: a lack of smoothness in material properties is expected to stress the time integrator.

The reactor's nominal full power is 3600 MW. The initiating perturbation consists of a ramp insertion of 2\$ of reactivity in 1 s:

$$\rho_{ex}(t) = \begin{cases} 0 & \text{if } t \leq 0 \text{ s,} \\ (2\$/\text{s}) \cdot t & \text{if } 0 \text{ s} < t < 1 \text{ s,} \\ 2\$ & \text{if } 1 \text{ s} \leq t. \end{cases} \quad (5.5)$$

Two sets of initial conditions are used: (1) the “hot zero power” (HZP) condition, and (2) the “very hot zero power” (VHZP) condition. The HZP IC is obtained by finding the steady state with the steady reactor power of 1 W. The VHZP IC is obtained by first finding the steady state at nominal transient power of 3600 MW, followed by setting the initial transient power to 1 W and setting the initial precursor family powers in equilibrium with the 1 W initial power, using

Eq. (5.6):

$$\tilde{C}_m^0 = \left(\frac{\beta_m}{\lambda_m \Lambda} \right) P^0. \quad (5.6)$$

The following notation is used:

P^0 = Initial reactor power. Units: W.

\tilde{C}_m^0 = Initial precursor family m power. Units: W.

The HZP IC essentially represents a steady shutdown core with hot inlet water. The VHZP IC is not a steady state: because the 1 W power is negligible for a full core, the fuel temperature will not stay steady, and will be dropping due to the coolant moving at full velocity through the core. The VHZP IC is studied here to stress the code: because it emphasizes the effect of cooling on the short transient, it is generally more difficult for a code to simulate, compared to the HZP IC. The VHZP IC is less physically realistic.

The benchmark problem, as specified here, is taken directly out of Ref. [52, `typwr.i` deck], with several modifications (to make the problem more realistic):

1. $N_{pi} = 36\,552$ for all axial segments in the benchmark; in the deck, the scaling factors vary.
2. Clad thickness $\Delta r_c = 0.57$ mm is used in the benchmark; in the deck, the clad is 0.1524 mm instead.
3. $\beta = 0.0065$ (the DNF for ^{235}U) is used in the benchmark; in the deck, $\beta/\Lambda = 297\text{ s}^{-1}$ is used. Keeping this β/Λ gives $\Lambda = 2.18855 \times 10^{-5}$ s.
4. $N_{f_t} = 10$ is used in the benchmark solution, with one gap and one clad radial cell. In the deck, a finite difference discretization with 17 points in the radial mesh is used.
5. $\dot{m} = 18\,395.15$ kg/s, $\bar{p}_{core} = 156.5478$ bar and $T_{w,0} = 550$ K are used in the benchmark; here \bar{p}_{core} is the core average operating pressure at which the coolant properties are evaluated as functions of temperature. These parameters are not available in the deck explicitly, because the deck models the full primary system, and so these parameters are not generally kept constant. These specific parameters were chosen as the steady state conditions from the deck.
6. Weisman correlation is used to evaluate the convective HTC in the benchmark [118]; in the deck the, the HTC is not specified explicitly, and is instead computed by RELAP5-3D's heat transfer module.
7. IAPWS-IF97 water properties are used in the benchmark [48]; RELAP5-3D v2.42, by default, uses the IFC-67 water properties [47].

The transient is integrated up to 3 seconds.

Three quantities are of interest: reactor reactivity, reactor power, and hottest axial segment's average fuel temperature. The convergence of peak reactivity, peak power, and hottest axial segment's final average fuel temperature is looked at.

5.2 Steady State Search

The benchmark problem is, by the specification, divided into 6 axial segments, and the fuel elements are concentrically divided into a fuel, gap and clad. In UO_2 fuel, apparently used by the typical PWR, fuel thermal conduction resistance almost always dominates the gap, clad and convective thermal resistances, due to its radius and low thermal conductivity (typical for a ceramic). For these reasons, it is necessary to divide the fuel into N_{f_l} concentric shells, to then ensure that the effective thermal resistance between the average fuel temperature and the bulk coolant is adequately modeled. Figure 5.2 illustrates the convergence of \bar{T}_{f4}^0 , the initial average fuel temperature of axial segment 4, as N_{f_l} increases. The fluctuations, which are clearly reducing in magnitude, arise from property discontinuities.

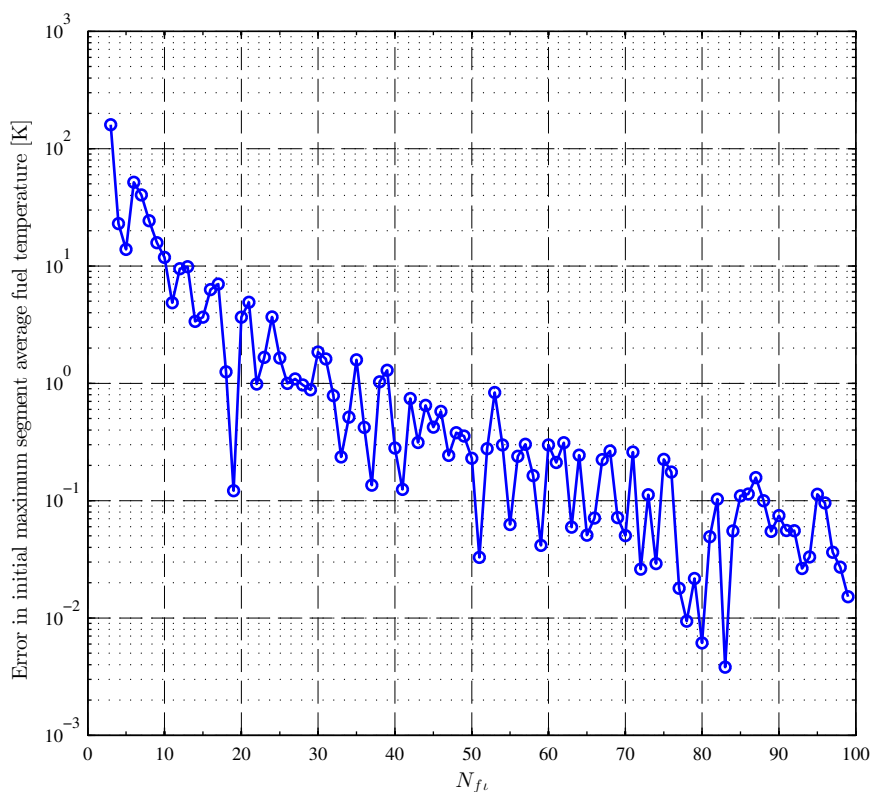


Figure 5.2: PWR PKE Initial Fuel Temperature Spatial Convergence Plot

The following nomenclature is used:

\bar{T}_{fi}^0 = Initial average fuel temperature of axial segment i . Units: K.

As shown in section 3.1, the steady state problem here is an eigenvalue problem, with the bias reactivity ρ_b acting to set the initial reactivity to zero, and by doing so, set the system to an initial steady state. As was discussed in subsection 4.3.2, `BGSolver` v1.03 was developed without a dedicated steady state solver: only a time integrator is present. Because of this, a steady state

solver was manually implemented: a residual function $P - P^0$ was defined, which replaced the $d\rho_b/dt$ element of the state derivative vector. ρ_b still remained one of the unknowns. The resulting system was subsequently solved using the Picard iteration-initiated Newton-Raphson nonlinear solver discussed in section 4.4. $N_P = 25$, $\lambda_P = 0.25$ and $\lambda_{NR} = 1.0$ was used; depending on $N_{f\ell}$, the solver converged in 25 Picard and 5 to 10 Newton-Raphson iterations.

RELAP5-3D steady state solver works by running a long transient with a modified heat diffusion component, which eventually arrives at a steady state; it explicitly computes ρ_b based on the specified initial reactivity (zero) and thermohydraulic state. The transients are then restarts of the resulting solution.

By inspection of Figure 5.2, we can see that at $N_{f\ell} = 10$, the error drops to below 15 K, and appears to enter a relatively stable convergence regime (a solution with $N_{f\ell} = 100$ was used as the reference solution). For this reason, $N_{f\ell} = 10$ was chosen as the radial mesh to use for the benchmark problem, which corresponds to $10 + 1 + 1$ “solid” regions (10 fuel, 1 gap and 1 clad). The RELAP5-3D deck used to solve the same problem, which uses a finite difference solution for the heat conduction equation in structures, used 13 radial grid points: 10 in the fuel, 1 between fuel and gap, 1 between gap and clad, and 1 on the clad outer surface.

The HZP and VHZP ICs were obtained from $P^0 = 1$ W and $P^0 = 3600$ MW steady state solutions, as described above. They were then used for time integration of the transient, described in the following section.

5.3 Results

MATLAB’s `ode15s` adaptive time integrator, and BDF3 with a range of time steps from $\Delta t = 0.5^9 \times 5$ ms (9.765625 μ s) to 5 ms were used for the reference solutions to the transient of the bond graph model. BDF3 with $\Delta t = 9.765625$ μ s was used as the reference solution. RELAP5-3D time integrator is, in theory, adaptive, but a maximum time step can be specified; solutions with specified time steps of $\Delta t = 0.5$ ms, 1 ms and 5 ms were obtained.

In earlier versions of RELAP5-3D, a bug was present in the point kinetics module, pointed out by Davis [119]. This bug would be relevant for this benchmark problem. For this work, RELAP5-3D v2.42 was used, in which the bug was corrected.

The obtained transients are shown in the following subsection.

5.3.1 Transient Results

Figures 5.3–5.5 show the reactor reactivity, power and hottest axial segment temperature transient solutions, obtained using both `BGSolver` v1.03 with adaptive and fixed step time integrators, and using RELAP5-3D.

Both the HZP and the VHZP transients look as expected: 1) reactivity rises linearly with negligible thermal feedback, following $\rho_{ex}(t)$, 2) once the reactor gets prompt critical, power shoots up, which enables rapid thermal feedback, 3) reactivity rapidly drops to subprompt levels, and 4) the reactor stabilizes shortly after the reactivity insertion ends. The BDF3, `ode15s` and RELAP5-3D (with a fine step) results match rather closely; the disagreements are likely due to the differences in water properties and the HTCs discussed in section 5.1. Additionally, the simplified nature of the bond graph thermal hydraulic model may account for some of the discrepancy. Note, that if constant properties are used, at steady state, the disagreements disappear entirely.

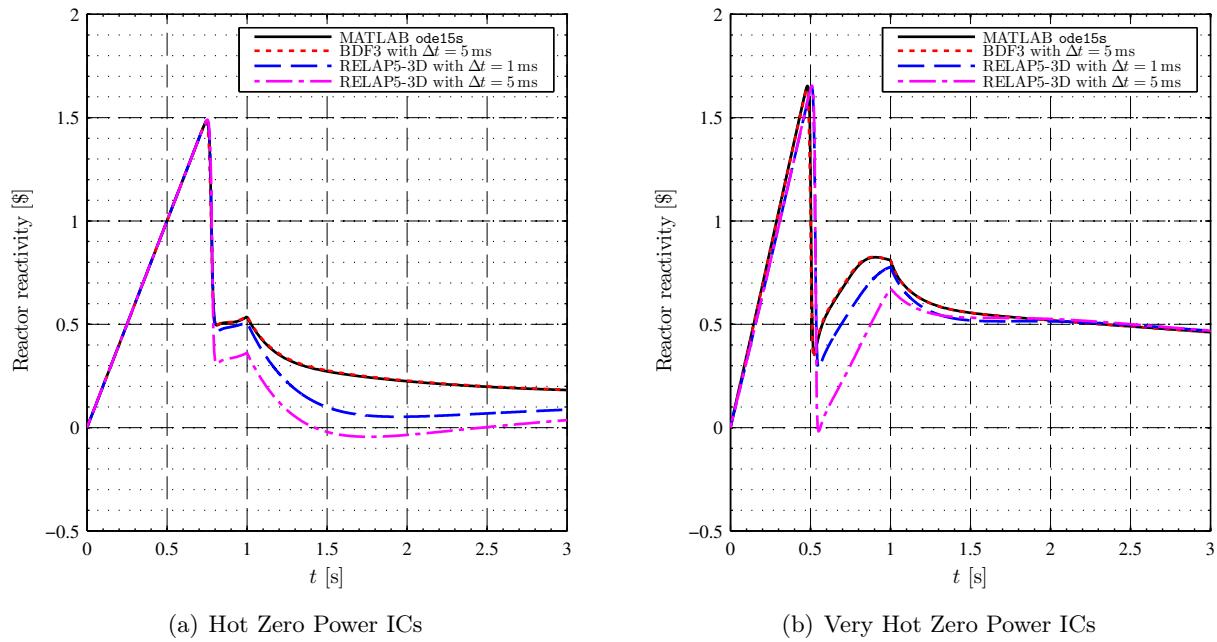


Figure 5.3: PWR PKE Benchmark Problem Reactivity Transient Solutions

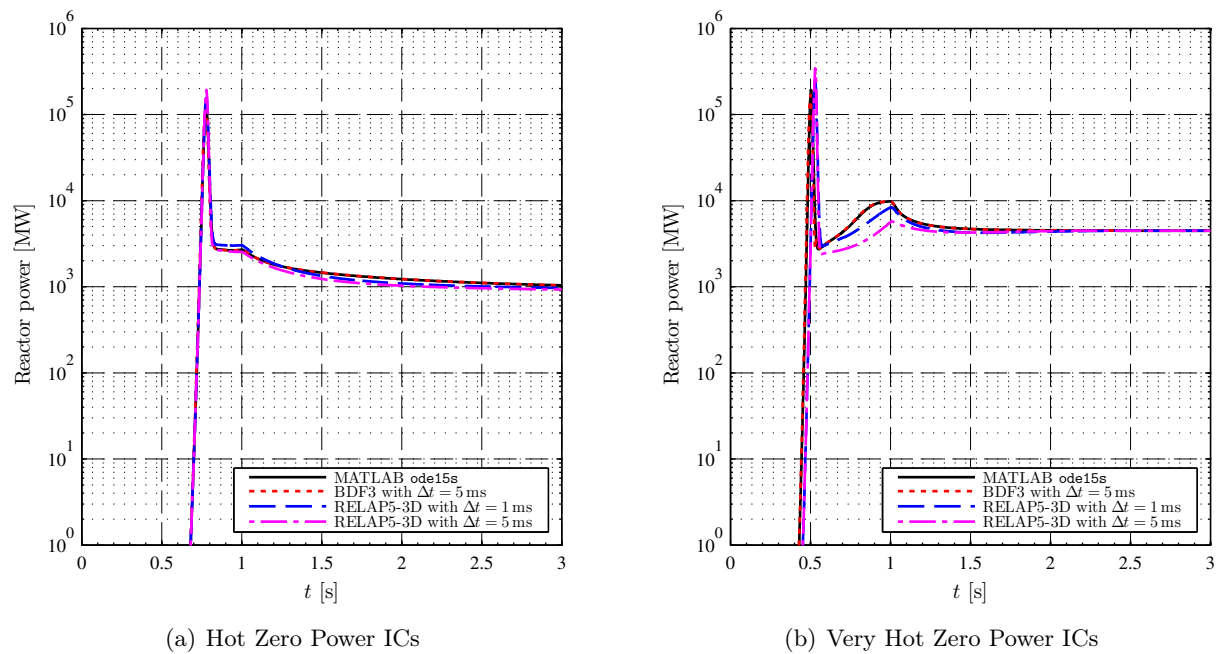


Figure 5.4: PWR PKE Benchmark Problem Reactor Power Transient Solutions

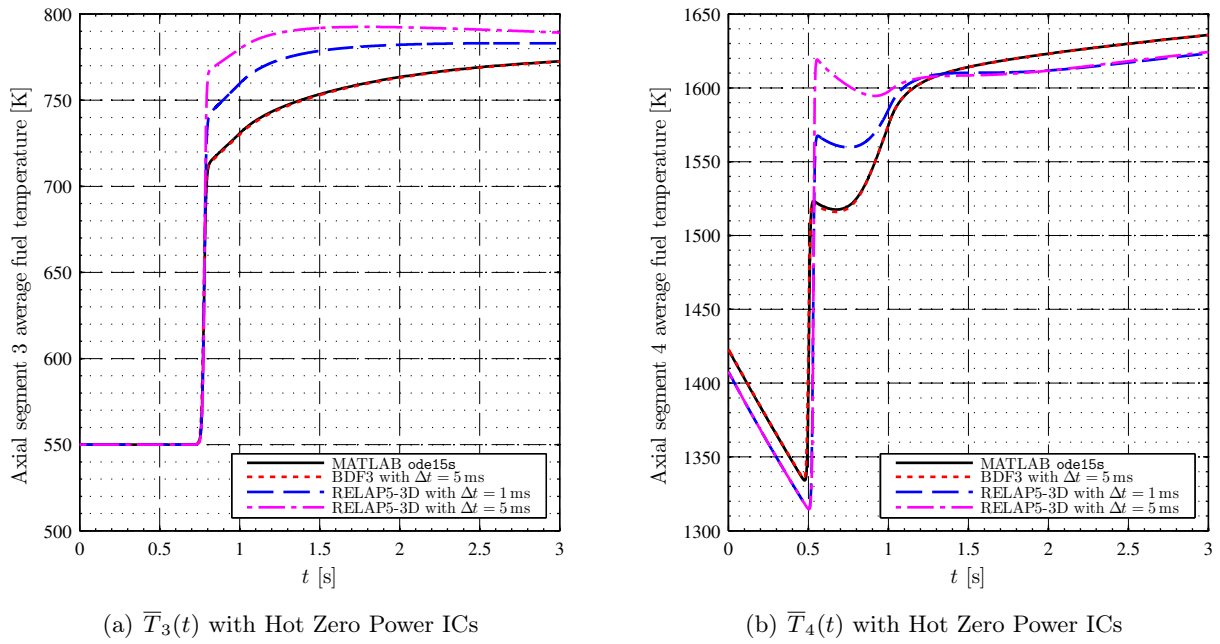


Figure 5.5: PWR PKE Benchmark Problem Hottest Axial Segment Temperature Transient Solutions

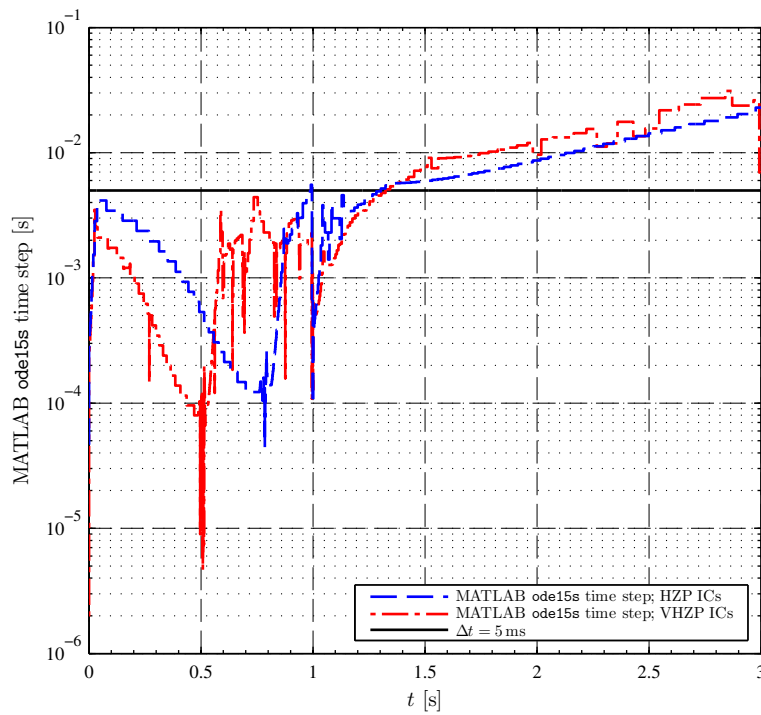


Figure 5.6: PWR PKE MATLAB ode15s Adaptive Time Integrator Required Time Steps

The fine 1 ms time step used by RELAP5-3D is shown as the time step necessary to converge the hottest segment average fuel temperature to within 10 K at all points in time, with the 0.5 ms RELAP5-3D solution as the reference.

The full power steady state \bar{T}_{f4} predicted by RELAP5-3D v2.42 and BGSolver v1.03 are approximately 15 K apart, which also confirms the difference in thermohydraulic properties as the reason for the disagreements between RELAP5-3D v2.42 and BGSolver v1.03 observed throughout the transient.

A fixed time step of $\Delta t = 5$ ms was used by BDF3; at this point, the solution is clearly converged, as indicated by the comparison with ode15s. A significantly finer maximum time step of 1 ms is required for RELAP5-3D v2.42 to converge; as the figures show, a coarser time step resulted in a significant overshoot of the post-peak drop in reactivity, for both transients. This is likely due to the fact that RELAP5-3D is, fundamentally, an operator splitting-based code, while BGSolver is fully coupled, when using the BDF3 time integrator. As discussed in subsection 2.2.1, split operator codes generally have lower order of convergence in time, and therefore require finer time steps.

The time step size used by ode15s is shown in Figure 5.6. The figure clearly illustrates the potential benefit of an adaptive time integrator with a fully coupled system: while the integrator refines the time step below 5 ms during sharp transitional periods (when the reactor is super-prompt critical and when reactivity insertion stops), for a lot of the transient, it uses a significantly coarser time step.

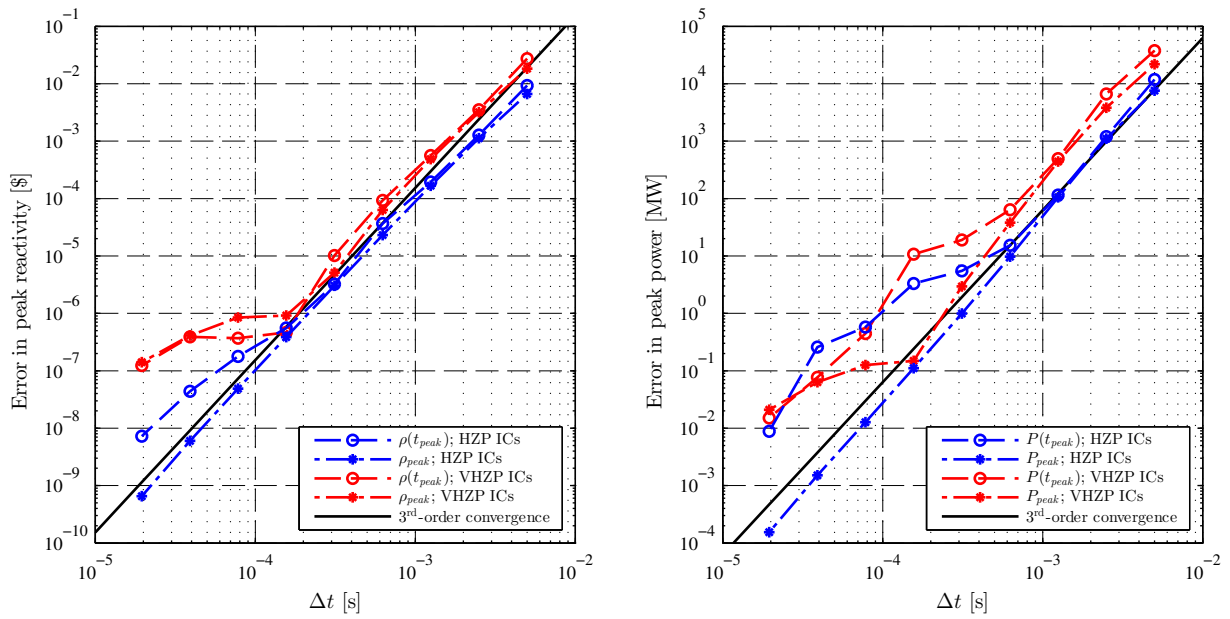
This behavior was not observed for RELAP5-3D: the code used the maximum specified time step, despite possessing an adaptive time stepping capability. The reason for this is, most likely, the following: RELAP5-3D's adaptive time stepper is set up to adjust the time step in response to time discretization errors in a single set of physics. However, because the code uses operator splitting, it cannot, without integrating both sets of physics multiple times, check for errors due to coupling, and therefore only couples at fixed time steps. No phase change occurs in this problem (which is what RELAP5-3D's time integrator is most sensitive to), and all of the error due to the split operator is because of an unconverged thermal feedback. With sufficiently fine time steps of 1 ms, the error disappears, but this is a much finer time step than the 5 ms than what a fully coupled fixed step time integrator, like BDF3, required.

The observed convergence of the BDF3 time integrator is studied in greater detail in the next subsection.

5.3.2 Convergence Results

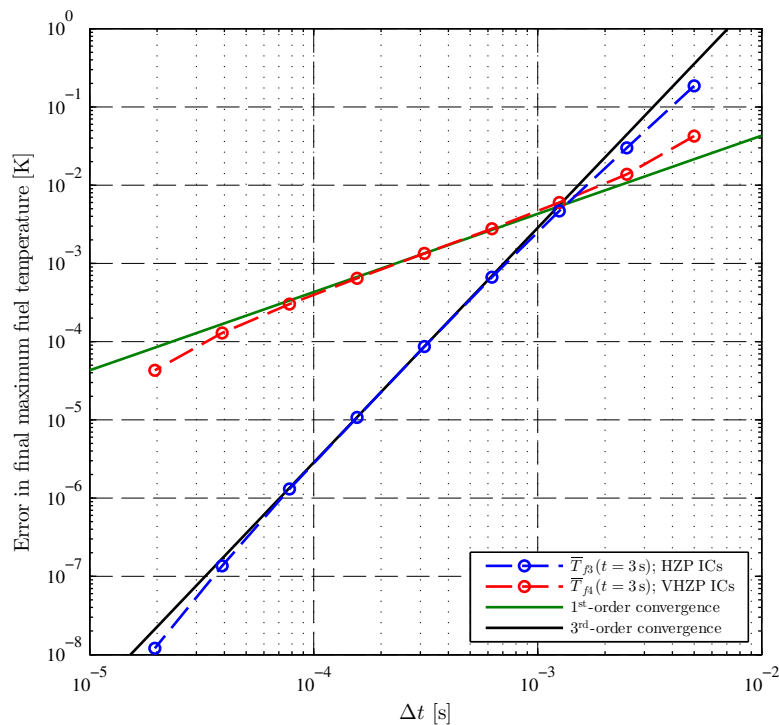
The convergence of three quantities of interest was studied, peak power, peak reactivity, and final average fuel temperature of the hottest axial segment. The convergence of the peaks was analyzed in two ways: the convergence of the quantity of interest at the time of true peak t_{peak} , and the convergence of the peak observed for each time step. The true peak quantity (power, reactivity) was estimated via cubic interpolation of the peak of the computed points, and the time at which it occurred was denoted t_{peak} :

t_{peak} = Time of the true peak of the quantity of interest (power, reactivity), obtained via cubic interpolation of the computed points. Units: s.



(a) Peak Reactivity Convergence Plot

(b) Peak Reactor Power Convergence Plot



(c) Final Hottest Axial Segment Temperature Convergence Plot

Figure 5.7: PWR PKE Benchmark Problem Temporal Convergence Plots

The BDF3 solution with $\Delta t = 0.5^9 \times 5 \text{ ms}$ ($9.765625 \mu\text{s}$) was used as the “exact” (reference) solution. Figure 5.7 illustrates the BDF3 time integrator convergence for a range of time steps.

The three figures clearly show that BDF3 successfully achieved 3rd-order convergence in time, even for the peak quantities at true peak times, which tend to be the more sensitive parameters. The 3rd-order convergence is present even at the relatively coarse time step of 5 ms. The one observed exception is the convergence of $\bar{T}_{f4}(t = 3\text{ s})$ with VHZP ICs, which only shows linear convergence. Possible explanations for this are: (a) VHZP is not a steady state IC, while the BDF3 time integrator is initialized with pre-initial states (see section 4.4), which implies an initial steady state, and (b) as Figure 5.5 shows, the VHZP-initiated transient occurs at higher temperatures, which, by Figure 5.1(a), corresponds to less smooth properties, which may lead to a reduction in convergence order.

Nevertheless, the hottest axial segment final average fuel temperature is clearly converged at 5 ms, which fully justifies the use of a fully coupled time integrator on this type of problem, even when superlinear convergence cannot be observed due to property and initiating perturbation discontinuities.

Excluding the sorting steps, the BDF3 time integration with 5 ms took 24 s on an Intel i7-2600 3.4 GHz machine with Windows 7 x64 and MATLAB R2013a x64. The `ode15s` time integration of the same problem took 2.2 s, and the RELAP5-3D v2.42 runs for the same problem on the same machine took 2.1 s with a 5 ms maximum time step. As was illustrated in subsection 5.3.1, RELAP5-3D is clearly not converged at this time step; with a time step of 1 ms, required for convergence, RELAP5-3D took approximately 12.4 s. This illustrates that while the fixed-step BDF3 time integrator (implemented in MATLAB) clearly loses in performance to the more efficient RELAP5-3D, the two are more comparable (within a factor of two) when a finer, as required, time step is used by RELAP5-3D. Both approaches are clearly beaten by an adaptive time integrator `ode15s`, which is, again, implemented in MATLAB, and not a compiled, higher-performance language.

Despite this being a small problem (86 unknowns), the sorting steps (steps 3 and 4 of the bond graph process) took approximately 30 s, which is more than the time integration itself. This clearly indicates that `BGSolver` v1.03, because of the symbolic sorting, cannot be scaled up to realistic full core reactor problems. However, it was hypothesized that the code would be able to handle problems on the order of several hundred unknowns, which is sufficient for a coarse mesh 2D multigroup diffusion problem with feedback. The use of `BGSolver` v1.03 to solve such problem is shown in chapter 6.

The findings of this chapter are summarized in the following section.

5.4 Summary

A symbolic sorting-based bond graph processing code, `BGSolver` v1.03, was used to solve a pressurized water reactor point kinetics rod ejection benchmark problem, with a single channel flow. The bond graph representation techniques used to represent the point kinetics equations and the linear, constant coefficient thermal feedback, were confirmed to be correctly derived and accurate. The bond graph solution was successfully compared to a RELAP5-3D solution of the same problem.

The benchmark problem was solved using the simplified PWR core model and both adaptive and fixed step fully coupled time integrators. The adaptive time integrator was able to respond

to the sharp changes in properties, while taking coarse time steps during the smoother parts of the transient. The fixed time step solution demonstrated the desired 3rd-order convergence for most quantities of interest, and was converged within 1 K for the coarse time step of 5 ms, during which the split operator RELAP5-3D v2.42 solution was clearly not converged.

The PWR PKE problem is a first successful test for the bond graph formalism as applied to reactor multiphysics, but larger, more realistic problems must also be addressed. Chapter 6 presents the solution of a benchmark 2D BWR control blade drop problem using the bond graph formalism.

Chapter 6

Two-dimensional BWR Control Blade Drop Problem

The success of the nonlinear point kinetics bond graph representation verified that `BGSolver` v1.03 was working as intended, and that full coupling is beneficial even at the time scales of reactivity-initiated accidents. However, the PWR PKE problem is quite small, and so it, alone, is not a sufficient test of the bond graph formalism and fully coupled time integrators for full core problems. A spatial kinetics problem must therefore be modeled, with the following objectives:

1. To verify that the bond graph representation techniques developed for multidimensional, multigroup neutron diffusion with delayed neutron precursors (section 3.3) adequately represent these physics.
2. To study how bond graph processing codes, particularly the sorting algorithms, perform for medium-size problems (on the order of 500 to 50 000 state variables). In particular, check if the presence of multiport resistors is an issue for the sorting algorithms.
3. To study whether full coupling maintains superlinear convergence, or at least the benefits from superlinear time integrators, when used on a more realistic reactor problem.

The Laboratorium für Reaktorregelung und Anlagensicherung (LRA) BWR control blade drop problem was chosen as the benchmark problem to study in more detail [36, Problem 14]. This problem has similar challenging time scales to the PWR PKE problem, reactor power in it varies over about 10 orders of magnitude, and, because a peripheral control blade is dropped, the power profile shifts throughout the transient. The speed of the transient ensures that full coupling effects, if potentially beneficial, will be experienced, and the spatial shape variation justifies using a fine spatial mesh, which is required for testing the algorithm's efficiency, as described above.

Both 2D and 3D versions of this problem exist; in this chapter, the 2D version is considered. The considerably more difficult 3D version is analyzed in chapter 7. Many of the coarse mesh-based results described in section 6.2 were published in Ref. [120].

Section 6.1 defines the benchmark problem. Sections 6.2 and 6.3 present the bond graph-based solutions to the coarse and fine mesh discretizations of the problem. Section 6.4 discusses the execution speeds of the bond graph processing codes used for these analyses, and section 6.5 summarizes the findings of the two studies.

6.1 Benchmark Problem Definition

The physics comprising the 2D LRA BWR control blade drop problem are: (a) two-group, 2D neutron diffusion over homogenized nodes, (b) two precursor families, and (c) homogenized, adiabatic thermal feedback. Equations (6.1)–(6.3) model these physics.

Equations (6.1) model two-group neutron diffusion, and are a specific form of Eq. (2.96) in a given homogenized region:

$$\begin{aligned} \frac{\partial}{\partial t} n_1(t, \vec{r}) &= \nabla \cdot D_{1,r} \nabla \phi_1(t, \vec{r}) - \left(\Sigma_{a1,r}(T) + \Sigma_{s21,r} + B_z^2 D_{1,r} \right) \phi_1(t, \vec{r}) + \\ &+ \frac{\nu(1-\beta)}{k_{eff}^0} \sum_{g'=1}^2 \Sigma_{fg',r} \phi_{g'}(t, \vec{r}) + \sum_{m=1}^2 \lambda_m c_m(t, \vec{r}) + S_{ex1}(t, \vec{r}), \end{aligned} \quad (6.1a)$$

$$\frac{\partial}{\partial t} n_2(t, \vec{r}) = \nabla \cdot D_{2,r} \nabla \phi_2(t, \vec{r}) + \Sigma_{s21,r} \phi_1(t, \vec{r}) - \left(\Sigma_{a2,r}(t) + B_z^2 D_{2,r} \right) \phi_2(t, \vec{r}). \quad (6.1b)$$

Equation (6.2) models the system of 2 delayed neutron precursor families, and is a form of Eq. (2.101) with a principal fissionable nuclide and $M = 2$:

$$\frac{\partial}{\partial t} c_m(t, \vec{r}) = \frac{\beta_m \nu}{k_{eff}^0} \sum_{g'=1}^2 \Sigma_{fg',r} \phi_{g'}(t, \vec{r}) - \lambda_m c_m(t, \vec{r}) \quad \forall m \in \{1, 2\}. \quad (6.2)$$

Equation (6.3a) is a form of Eq. (2.170), with $\gamma = 0$ and constant energy per fission. Equation (6.3b) is an adiabatic form of Eq. (2.139), with heat diffusion dropped and the other terms divided by a constant volumetric heat capacity:

$$\dot{u}_{v,ex}(t, \vec{r}) = \kappa \sum_{g'=1}^2 \Sigma_{fg',r} \phi_{g'}(t, \vec{r}), \quad (6.3a)$$

$$\frac{\partial}{\partial t} T(t, \vec{r}) = \alpha \dot{u}_{v,ex}(t, \vec{r}) = \alpha \sum_{g'=1}^2 \Sigma_{fg',r} \phi_{g'}(t, \vec{r}). \quad (6.3b)$$

Temperature and volumetric thermal energy density are related through Eq. (6.4), which is a form of Eq. (2.138) with a constant volumetric heat capacity defined by Eq. (6.5):

$$T(u_v) = \frac{1}{c_v} u_v(t, \vec{r}), \quad (6.4)$$

$$c_v = \frac{\kappa}{\alpha}. \quad (6.5)$$

The following nomenclature is used:

- r = A given homogenized region in which \vec{r} is.
- $D_{g,r}, \Sigma_{s21,r}, \Sigma_{fg,r}$ = Constant homogenized region-specific group neutron properties.
- $\Sigma_{a1,r}(T)$ = Fast absorption cross section, affected by thermal feedback, which is described by Eq. (6.6). Units: cm^{-1} .

$\Sigma_{a2,r}(t)$	= Thermal absorption cross section, time-dependent to model the control blade movement (initiating perturbation). Units: cm^{-1} .
B_z^2	= Axial geometric buckling, which accounts for axial leakage. Units: cm^{-2} .
$T, T(t, \vec{r}), T(u_v)$	= Temperature profile. Units: K.
ν	= Average total number of neutrons born per fission, here assumed constant. Dimensionless.
c_v	= Volumetric heat capacity, here assumed constant. Units: $\text{kJ}/\text{m}^3 \text{K}$.
κ	= Energy generated per fission, here assumed constant. Units: MeV.
α	= Temperature conversion factor. Units: K cm^3 .

Temperature dependence of the fast absorption cross section is modeled by:

$$\Sigma_{a1,r}(T) = \Sigma_{a1,r}^0 \left[1 + \gamma_{fb} \left(\sqrt{T} - \sqrt{T^0} \right) \right], \quad (6.6)$$

in which:

$\Sigma_{a1,r}^0$	= Fast absorption cross section of region r at reference temperature, a constant. Units: cm^{-1} .
γ_{fb}	= Feedback constant. Units: $\text{K}^{-1/2}$.
T^0	= Reference temperature. Units: K.

The initial region properties are given in Table 6.1.

Figure 6.1 defines the reactor horizontal layout and boundary conditions. Here, regions 1–4 are the fuel, and region 5 is the water reflector. Region R indicates the subregion of region 3 from which the blade is dropped during the initiating perturbation.

Additional reactor-wide parameters are given in Table 6.2.

For the 2D problem, the initiating perturbation (control blade drop) is modeled by Eq. (6.7).

$$\Sigma_{a2,R}(t) = \begin{cases} \Sigma_{a2,3}^0 \cdot \left[1 - (0.0606184 \text{ s}^{-1} \cdot t) \right] & \text{if } t \leq 2 \text{ s}, \\ \Sigma_{a2,3}^0 \cdot 0.8787631 & \text{if } t > 2 \text{ s}, \end{cases} \quad (6.7)$$

in which:

$\Sigma_{a2,3}^0$	= Thermal absorption cross section in region 3, which, initially, is identical to thermal absorption cross section in region R. Units: cm^{-1} .
-------------------	---

Table 6.1: LRA BWR Benchmark Problem Initial Region Properties

Region	Material	g	$D_{g,r}$ [cm]	$\Sigma_{a,g,r}$ [cm ⁻¹] ^a	$\nu\Sigma_{f,g,r}$ [cm ⁻¹]	$\Sigma_{s21,r}$ [cm ⁻¹]
1	Fuel 1,	1	1.255	0.008252	0.004602	0.02533
	blade in	2	0.211	0.1003	0.1091	
2	Fuel 1,	1	1.268	0.007181	0.004609	0.02767
	blade out	2	0.1902	0.07047	0.08675	
3, R ^b	Fuel 2,	1	1.259	0.008002	0.004663	0.02617
	blade in	2	0.2091	0.08344	0.1021	
4	Fuel 2,	1	1.259	0.008002	0.004663	0.02617
	blade out	2	0.2091	0.073324	0.1021	
5	Reflector	1	1.257	0.0006034	0	0.04754
		2	0.1592	0.01911	0	

^a All $\Sigma_{a1,r}$ are given at $T = T^0 = 300$ K.

^b Region R is the subregion of region 3, from which the blade is dropped during the initiating perturbation. They are initially identical.

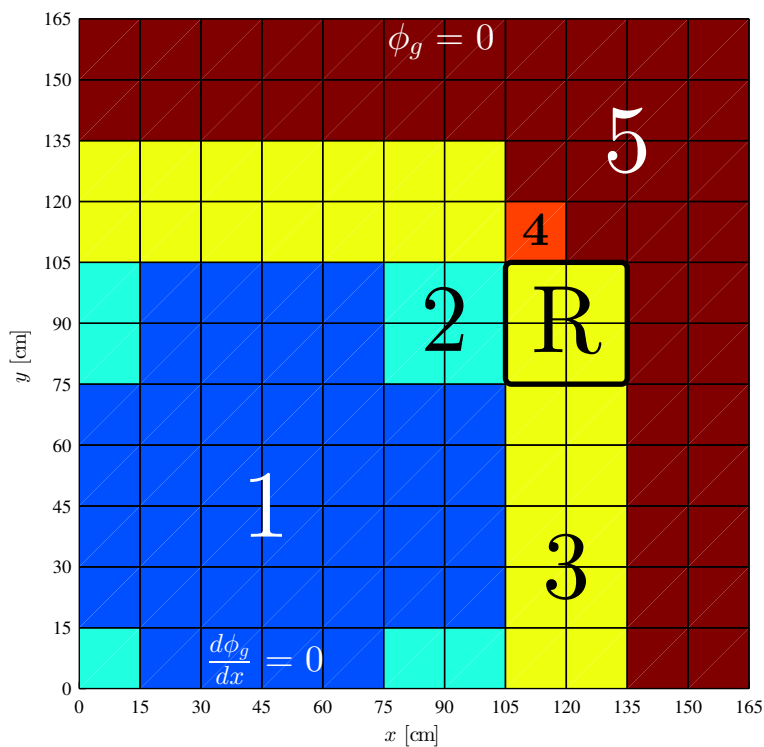


Figure 6.1: 2D LRA BWR Benchmark Problem Specification (from Ref. [36, Problem 14])

Table 6.2: LRA BWR Benchmark Problem Additional Reactor-wide Parameters

Parameter	g or m	Value
V_{ng} [cm/s]	1	3.0×10^7
	2	3.0×10^5
β_m	1	0.0054
	2	0.001087
λ_m [s ⁻¹] ^a	1	0.0654
	2	1.35
B_z^2 [cm ⁻²]		1.0×10^{-4}
ν [n/fission]		2.43
α [K cm ³]		3.83×10^{-11}
γ_{fb} [K ^{-1/2}] ^a		3.034×10^{-3}
κ [J/fission]		3.204×10^{-11}
V_{cb} [cm/s] ^b		150

^a Reference [36, Problem 14] lists $\lambda_1 = 0.00654 \text{ s}^{-1}$ and $\gamma_{fb} = 2.034 \times 10^{-3} \text{ K}^{-1/2}$. Sutton and Aviles (Ref. [44]) suggest that the existing published solutions (e.g., Ref. [121]), including the ones cited in Ref. [36, Problem 14], have instead been run with the properties as given in this table. As shown in section 6.3, Sutton and Aviles' suggestion appears to be correct, and so these properties are used in this text, unless stated otherwise.

^b V_{cb} is the control blade drop velocity, which is used to initiate the perturbation in the 3D LRA BWR problem (chapter 7).

Core average power density and temperature are the two quantities of interest, defined by Eqs. (6.8) and (6.9), respectively. Initial average power is given in Eq. (6.10); initial temperature is flat T^0 throughout the core.

$$\bar{P}_{core}(t) = \frac{1}{V_{core}} \iiint_{V_{core}} dV \dot{u}_{v,ex}(t, \vec{r}), \quad (6.8)$$

$$\bar{T}_{core}(t) = \frac{1}{V_{core}} \iiint_{V_{core}} dV T(t, \vec{r}), \quad (6.9)$$

$$\bar{P}_{core}^0 = 1.0 \times 10^{-6} \text{ W/cm}^3, \quad (6.10)$$

where:

$\bar{P}_{core}(t)$ = Core average power density. Units: W/cm³.

$\bar{T}_{core}(t)$ = Core average temperature. Units: K.

V_{core} = Core volume. Here, “core” refers only to the 78 fuel assemblies (see Figure 6.1), and does not include the reflector. Units: cm² for the 2D problem, cm³ for the 3D problem.

\bar{P}_{core}^0 = Initial core average power density. Units: W/cm³.

The initial group flux profiles are found by solving the eigenproblem and scaled to yield the above core average power density. The initial precursor densities are computed by assuming them to be in equilibrium with the eigenfunction fluxes, and substituting the eigenfunction fluxes into

a steady-state version of Eq. (2.101). Initial temperature profile is $T(0, \vec{r}) = T^0$ throughout the core. Symmetric BCs (Neumann, zero current) are on the $x = 0$ and $y = 0$ planes, and zero flux Dirichlet BCs are on the $x = 165$ cm and $y = 165$ cm planes.

Because this is an adiabatic problem, the two quantities of interest are the peak average core power density and the final average core temperature. The problem was studied with two different versions of `BGSolver` (both discussed in chapter 4: version 1.03 for the coarse mesh model, and the higher performance version 2.0 for the fine mesh model).

6.2 Coarse Mesh Model

For all 2D LRA BWR solutions, structured, square meshes were used, because they naturally fit the highly structured problem geometry. The coarsest possible mesh that doesn't require additional property averaging is a $15 \text{ cm} \times 15 \text{ cm}$ square, which is exactly the size of one fuel assembly. This is the coarse mesh discretization; such mesh cannot adequately estimate thermal leakage, and so is very inaccurate. However, the reactivity-initiated accidents with such mesh still exhibit the appropriate time scales, and so it was deemed sufficient to be used to test `BGSolver` v1.03 for a larger problem than the PWR PKE problem.

Because the coarse mesh could not adequately model thermal leakage, the amount of reactivity inserted by the blade drop was too significantly high, and so the initiating perturbation was changed, reducing the absorption cross section drop by 25%:

$$\Sigma_{a2,R}(t) = \begin{cases} \Sigma_{a2,3}^0 \cdot \left[1 - 0.75 \times (0.0606184 \text{ s}^{-1} \cdot t) \right] & \text{if } t \leq 2 \text{ s}, \\ \Sigma_{a2,3}^0 \cdot \left[1 - 0.75 \times (0.0606184 \text{ s}^{-1} \cdot 2 \text{ s}) \right] & \text{if } t > 2 \text{ s}. \end{cases} \quad (6.11)$$

Additionally, the original λ_1 and γ_{fb} specified in Ref. [36, Problem 14] (0.00654 s^{-1} and $2.034 \times 10^{-3} \text{ K}^{-1/2}$, respectively) were used for the coarse mesh model, instead of the ones specified in Table 6.2; this difference in parameters contributes to the differences between the coarse mesh and fine mesh solutions, but the mesh itself is the more significant cause of the differences.

Finite volume spatial discretization, derived in subsection 2.1.2.2 and represented with bond graphs in section 3.3, was used to discretize the physics in the problem over the coarse mesh.

6.2.1 Steady State Search

As discussed in subsection 4.3.2, `BGSolver` v1.03 was developed without a dedicated steady state solver: only a time integrator is present. For this reason, a long asymptotic subcritical transient was used as the steady state solver for the coarse mesh problem.

Consider Eqs. (2.128), the PKEs with constant parameters, under the following assumptions: 1) constant external source $S_{ex} > 0$, 2) negligible variation in reactivity ρ , 3) a sub-prompt reactivity $\rho < \beta$, 4) one effective delayed neutron family ($M = 1$), and 5) the prompt jump approximation, which neglects dA/dt . Under these assumptions, and the initial condition $A^0 = A(0)$, the solution for $A(t)$ becomes [122]:

$$A(t) = \begin{cases} A^0 + \left(\frac{\lambda \Lambda}{\beta} S_{ex} \right) t & \text{if } \rho = 0, \\ A^0 \exp\left(\frac{\lambda \rho}{\beta - \rho} t \right) + \left(\frac{\Lambda}{\rho} S_{ex} \right) \left[\exp\left(\frac{\lambda \rho}{\beta - \rho} t \right) - 1 \right] & \text{if } \rho \neq 0, \end{cases} \quad (6.12)$$

where:

A^0 = Initial value of flux amplitude function. Units (assuming dimensionless neutron importance function): neutrons.

Equation (6.12) indicates that sufficiently long after the prompt jump and after all of the delayed neutron families achieve equilibrium with the flux, the flux in the reactor will either: (a) asymptotically achieve a steady state if $\rho < 0$; (b) continue growing linearly in time if $\rho = 0$, or (c) exponentially grow and run away if $\rho > 0$.

As discussed in subsection 2.1.3, k_{eff}^0 modifies ρ and Λ . The objective of a criticality search is to find the eigenvalue and eigenfunction which make the reactor critical ($\rho = 0$). Under the above assumptions, the flux does not achieve a steady state with $\rho = 0$, but does if $\rho < 0$. As the reactor's reactivity approaches criticality while staying subcritical, the asymptotic climb to steady state slows down, until it results in a linear growth for a perfectly critical reactor.

Using these facts, a near-steady state solver can be constructed, which relies on finding a k_{eff}^0 that yields a very long asymptotic climb for $A(t)$, therefore resulting in a slightly subcritical reactor. The degree of subcriticality is controlled by the minimum length of time t_{ss} it takes for the reactor to achieve a steady state; a shorter t_{ss} results in a more subcritical reactor. The magnitude of S_{ex} can be chosen to control the initial power density, as long as it is sufficiently small not to result in sensitive heat and cause thermal feedback over the period t_{ss} .

S_{ex} is derived from $S_{ex1}(t, \vec{r})$ using Eq. (2.124). The shape of $S_{ex1}(t, \vec{r})$ can be chosen arbitrarily; however, it affects the initial power profile. It must also be constant, as per assumption 1) above. A flat fast source was used in the fuel nodes. Equations (6.13)–(6.16) summarize the resulting coarse mesh steady state solver's parameters:

$$S_{ex1}(t, \vec{r}) = 44.4 \text{ neutrons/cm}^3 \text{ s}, \quad (6.13)$$

$$t_{ss} = 10\,000 \text{ s}, \quad (6.14)$$

$$\phi_g^0(\vec{r}) = \phi(0, \vec{r}) = 0 \quad \forall g \in \{1, 2\}, \quad (6.15)$$

$$c_m^0(\vec{r}) = c_m(0, \vec{r}) = 0 \quad \forall m \in \{1, 2\}, \quad (6.16)$$

with the following nomenclature:

t_{ss} = Time over which the long transient is integrated. Units: s.

$\phi_g^0(\vec{r})$ = Scalar group flux g profile at the beginning of the long transient. Note, that this is different from the initial scalar flux used for the control blade drop transient analysis. Units: neutrons/cm² s.

$c_m^0(\vec{r})$ = Precursor family m density profile at the beginning of the long transient. Note, that this is different from the initial precursor density profile used for the control blade drop transient analysis. Units: precursors/cm³.

MATLAB's `ode15s` time integrator is used to integrate the long transient for a given k_{eff}^0 . Running this integrator multiple times for different values of k_{eff}^0 , and using a binary search algorithm, yielded a k_{eff}^0 with which the reactor was weakly subcritical, but sufficiently close to criticality to take most of the 10 000s to achieve apparent steady state. Together with this

eigenvalue, the flux and precursor density profiles at the end of the 10 000 s long transient were used as the initial conditions for the control blade drop transient.

This steady state solution procedure was used on two different reactor states: one with the control blade fully inserted ($\Sigma_{a2,R} = \Sigma_{a2,R}(0\text{ s}) = 0.08344\text{ cm}^{-1}$), and one with the control blade fully withdrawn ($\Sigma_{a2,R} = \Sigma_{a2,R}(2\text{ s}) = 0.075853\text{ cm}^{-1}$). The resulting eigenvalues were 1.0045 and 1.0290, respectively. By Eq. (2.123), these eigenvalues correspond to a control blade reactivity worth of 3.76\$. As will be shown in section 6.3, both the eigenvalues and the control blade worth are significantly overestimated; this is expected, because the coarse mesh underpredicts the thermal leakage, and therefore overestimates the reactor criticality.

Figure 6.2 shows the initial power density profile obtained by the coarse mesh model. It is flatter than the true initial critical power distribution (section 6.3), but still exhibits a power peaking factor of 2.373, which is expected to grow after the blade is dropped.

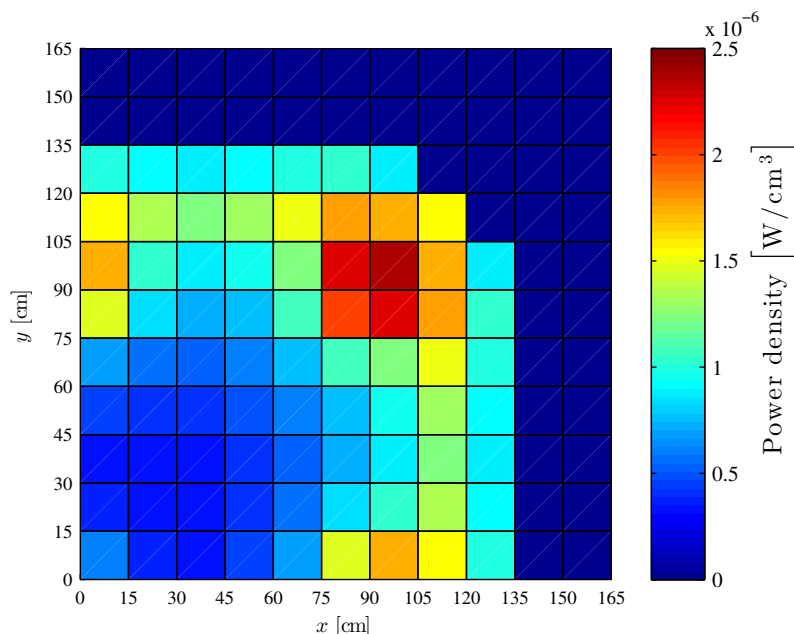


Figure 6.2: 2D LRA BWR Coarse Mesh Initial Power Density
(normalized to $\bar{P}_{core}^0 = 1.0 \times 10^{-6}\text{ W/cm}^3$)

The coarse mesh transient results, based on the initial eigenvalue of 1.0045 and the state vector from the end of the long subcritical transient, are presented in the following subsection.

6.2.2 Transient Results

As was discussed in section 4.4, `BGSolver` v1.03 was developed to be used with both split operator and fully coupled time integrators. BDF1, BDF2 and BDF3 time integrators, both fully coupled and with a split operator, were used to solve the coarse mesh 2D LRA blade drop transient. Figures 6.3 and 6.4 show the computed solutions for a set of time integrators and time step sizes. Fully coupled BDF3 with $\Delta t = 0.1\text{ ms}$ was used as the reference solution.

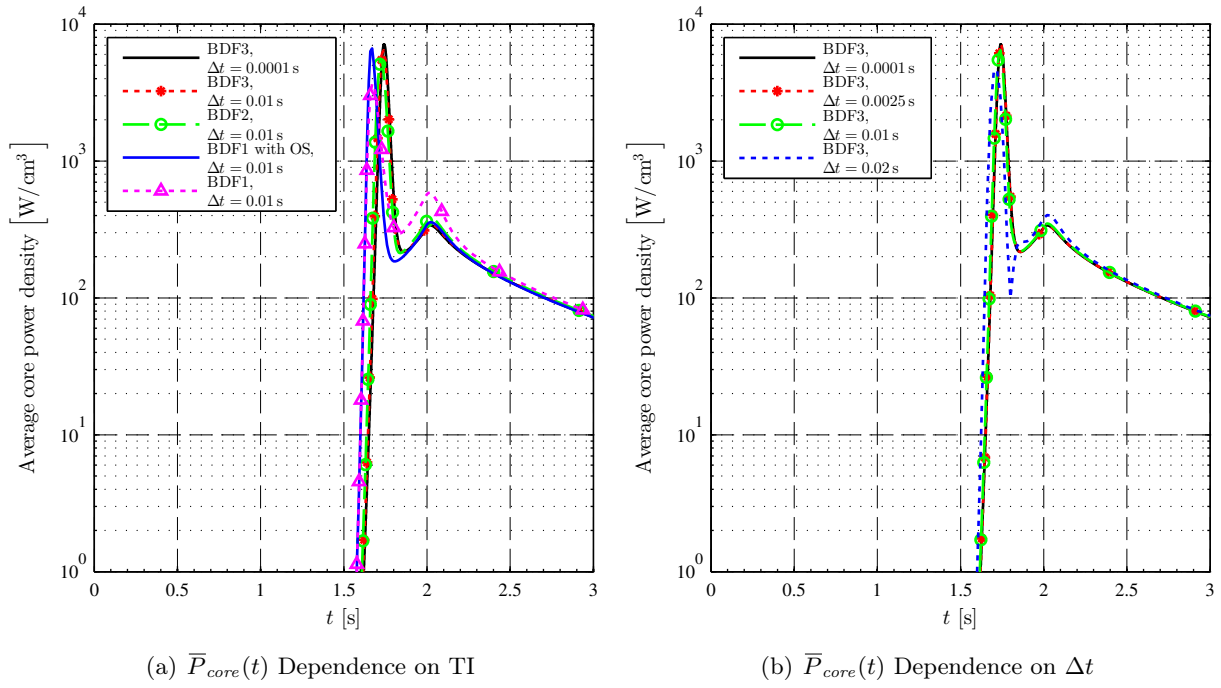


Figure 6.3: 2D LRA BWR Coarse Mesh Average Core Power Density Transients

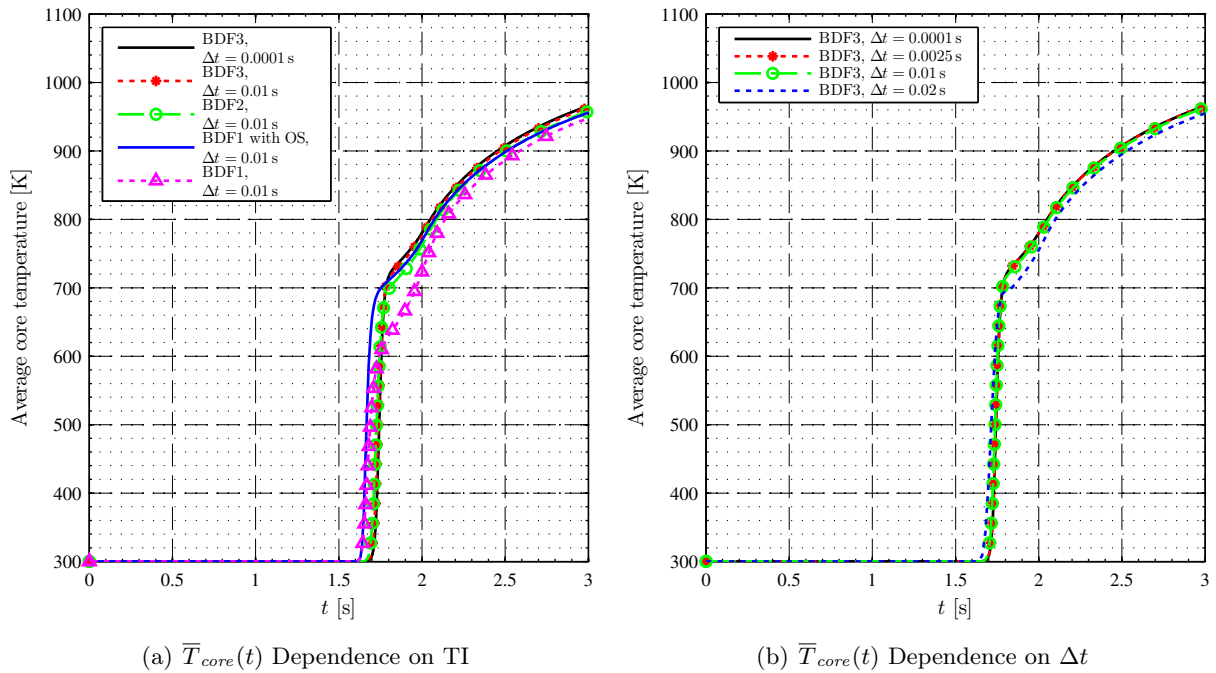


Figure 6.4: 2D LRA BWR Coarse Mesh Average Core Temperature Transients

These solutions are physically reasonable for an adiabatic control blade drop problem: a sharp rise in power, followed by temperature rise and negative reactivity insertion due to thermal feedback, followed by a drop in power, slower subsequent rise until the end of the reactivity insertion, and then a slow power decay due to the constantly increasing temperature. As shown in section 6.3, this behavior qualitatively matches the true solution, with the differences being due to the modifications discussed above, and the coarse mesh.

As expected, the higher order methods provide the more accurate solutions. Methods with coarser time steps generally predict a faster than reference rise to peak power, and also tend to underpredict the peak. High order methods tend to predict the timing of the peak better than lower order methods, and, for this reason, stay more accurate than lower order (both fully coupled and split operator) methods for all attempted time step sizes. This is illustrated by the order of convergence studies demonstrated in the following subsection.

6.2.3 Convergence Results

Convergence of two quantities of interest is studied: the peak average core power, and the final average core temperature. Figure 6.5 illustrates the fully coupled and split operator time integrators' convergence for a range of time steps.

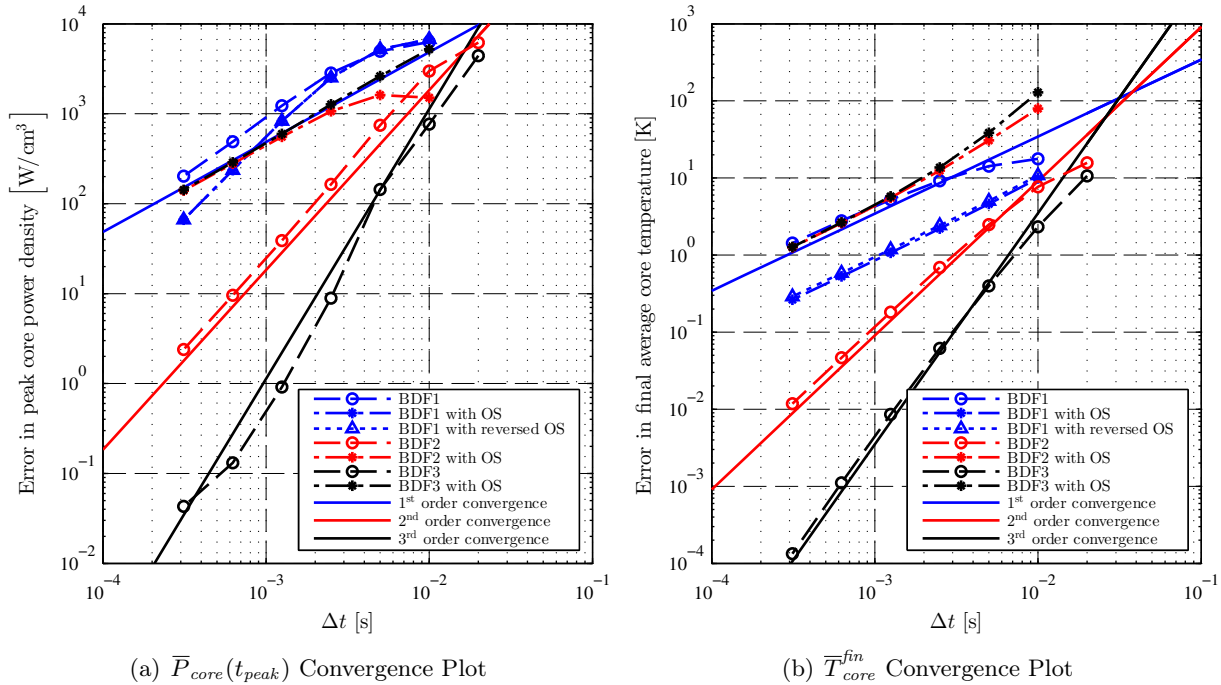


Figure 6.5: 2D LRA BWR Coarse Mesh Temporal Convergence Plots

Fully coupled BDF3 with $\Delta t = 0.1$ ms was used as the reference solution. The true peak power was estimated via cubic interpolation of the peak of the computed points, and the time at which it occurred was denoted t_{peak} :

t_{peak} = Time of the true peak average core power, obtained via cubic interpolation of the computed points. Units: s.

The $\bar{P}_{core}(t)$ transients were then evaluated at $t = t_{peak}$, using polynomial interpolations of appropriate degree (1 for BDF1, 2 for BDF2 and 3 for BDF3, for both split operator and fully coupled TIs). Two versions of split operator BDF1 were used, with the order of physics evaluation switched; the default order is the integration of fluxes and precursor densities' first, followed by temperature.

The findings of this study repeat the findings in chapter 5:

1. Split operator TIs are limited to first order, regardless of the order of the TI used for individual physics.
2. Superlinear fully coupled TIs, overall, show their theoretical order of convergence, even for the peak average core power, which tends to be the more sensitive parameter.
3. Superlinear fully coupled TIs consistently perform better than the split operator TIs; for a given accuracy, about a factor of 8 advantage in time step size (or more) can be gained by using BDF3 over BDF1 with OS.

Additional findings of this specific study are:

4. To get an error of 1 K in \bar{T}_{core}^{fin} , approximately an 8 ms time step is required for the BDF3 TI with FC. Comparatively, a 1 ms time step is required for a BDF1 with OS TI. This is almost an order of magnitude advantage, which is significant.
5. To get an error of about 71.1 W/cm³ in $\bar{P}_{core}(t_{peak})$ (about 1% error), again, approximately an 8 ms time step is required for the BDF3 TI with FC. Once again, the BDF1 with OS TIs require a 1 ms time step.
6. Reversing the order of the BDF1 OS triangular coupling did not produce an appreciable difference in accuracy or in order of convergence.
7. Due to the coarseness of the mesh, the solutions obtained were not accurate. However, they were qualitatively correct, exhibiting the expected transient shapes in all of the quantities analyzed. This verified that the bond graph representation developed in section 3.3 was, most likely, correct, although this remained to be verified by the fine mesh study (section 6.3).
8. As expected, the symbolic sorting employed by `BGSolver` v1.03 was the major bottleneck, with the sorting taking longer than most short time step solutions. This finding highlighted the need for the matrix-based sorting algorithm development, which was implemented in `BGSolver` v2.0 and used for subsequent studies.

Overall, this coarse mesh study confirmed that the bond graph-based approach to coarse mesh 2D LRA problem, and by extension, to coarse mesh multigroup diffusion, was implemented correctly, and maintained superlinear convergence expected from the fully coupled time integrators. It also highlighted the disadvantage that split operator time integrators face when integrating reactivity-initiated accidents.

To confirm the approach's validity, and to ensure that the approach is sufficiently fast for problems with realistic meshes, the fine mesh LRA BWR 2D model was constructed and used. It is described in the following section.

6.3 Fine Mesh Model

To refine the coarse mesh model in section 6.2, the coarse $15\text{ cm} \times 15\text{ cm}$ square mesh elements were each subdivided into $N \times N$ constituent square mesh elements, with a side of Δh , where:

N = The number of fine square mesh elements that, arranged side-by-side, fit into a single $15\text{ cm} \times 15\text{ cm}$ coarse square mesh element. There are therefore N^2 fine mesh elements in a single $15\text{ cm} \times 15\text{ cm}$ fuel assembly-sized square. Dimensionless.

Δh = Length of the side of a single fine square mesh element, given by Eq. (6.17). Units: cm.

$$\Delta h = \frac{15\text{ cm}}{N}. \quad (6.17)$$

Because the fine mesh model, assuming a sufficiently large N , is expected to converge to the correct solution, unlike for the coarse mesh model, no modifications were made to the benchmark specification given in section 6.1. The equations are the same as specified at the beginning of this chapter, and so the bond graph representation from section 3.3 may again be used.

The N required to spatially converge the 2D LRA BWR problem is identified in the following subsection.

6.3.1 Steady State Search

As discussed in subsection 4.3.2, `BGSolver` v2.0 was developed with a built-in eigenproblem solving capability. The solver, here, is configured as follows:

1. An eigenvalue storage element is added. This storage element is marked as an "eigenvalue storage element" to exclude its flow from the state derivative vector. This element moderates the MR5 elements, by supplying k_{eff}^0 to them, which they use to divide the prompt neutron and precursor production components of the corresponding flows.
2. An algebraic function $f_{pp}(\vec{\mathbf{x}})$ is defined:

$$f_{pp}(t, \vec{\mathbf{x}}) = \bar{P}(t, \vec{\mathbf{x}}) - \bar{P}_{core}^0, \quad (6.18)$$

where

$\bar{P}(t, \vec{\mathbf{x}})$ = Average core power, defined as the sum of the power flows (outputs from the MR5 elements to the thermal 0-junctions) divided by V_{core} . Units: W/cm^3 .

$f_{pp}(\vec{\mathbf{x}})$ = Average core power residual. Units: W/cm^3 .

3. The thermal state variables are set to known values: the thermal capacitors' displacements

are set to total thermal energies which correspond to T^0 . They are excluded from the state vector, and the corresponding thermal capacitors' flows are excluded from the state derivative vector.

4. The system is sorted and the state derivative vector is formed. k_{eff}^0 , the displacement on the eigenvalue capacitor, is part of the state vector, but its time derivative is not part of the state derivative vector.
5. The state derivative vector is appended with $f_{pp}(\vec{\mathbf{x}})$.
6. Time is set to 0s (or, alternatively, to 2s to model the system with the control blade dropped), and the appended state derivative is set to zero and algebraically solved, using a nonlinear solver. Regular Newton's method was used here; the versions of nonlinear solvers supported by `BGSolver` v2.0 were discussed in section 4.4.
7. The resulting solution is the set of flux and precursor displacements which correspond to the flux and precursor eigenfunctions normalized by the given \bar{P}_{core}^0 , together with k_{eff}^0 , the last unknown state variable. This solution, together with the above specified (known) thermal displacements, can be used as the initial conditions for the transient.

To ensure that the spatial mesh can adequately model the transient, a spatial convergence study was conducted, with two goals: 1) in the withdrawn control blade configuration, k_{eff}^0 must be converged within 15 pcm, and 2) in the inserted control blade configuration, all $\bar{P}_{asm,l}^0$ must be converged within 2%. These values were chosen because they appear to be, approximately, the range within which the benchmark solutions fall [44]. The following nomenclature was used:

$$\bar{P}_{asm,l}^0 = \text{Fuel assembly } l \text{ power eigenfunction, averaged over the assembly. } l \in [1, \dots, 78]. \text{ Units: } \text{W/cm}^3.$$

Figure 6.6 illustrates the spatial convergence of k_{eff}^0 and $\bar{P}_{asm,l}^0$, obtained using the above steady state solver parameters. The reference solutions are obtained with $N = 50$, which corresponds to $\Delta h = 0.3$ cm.

These plots illustrate that the spatial convergence conditions are met with $N = 10$, which corresponds to $\Delta h = 1.5$ cm. They also confirm that the finite volume discretization used in section 3.3 is of 2nd order in space. $N = 3$ appears to give an unexpectedly accurate solution; this is, mostly likely, due to the fact that the solver is not yet spatially asymptotic for such coarse spatial steps.

Figure 6.7 shows the initial power density profile obtained with $N = 10$. Qualitatively, this solution matches Figure 6.2, but unlike it, this solution is spatially converged. With $N = 10$, the eigenvalues are 0.99628 and 1.01519 for the inserted and dropped blade configuration, respectively. By Eq. (2.123), these correspond to a control blade worth of 2.876\$, which is significantly less than what was predicted by the coarse mesh model in subsection 6.2.1. The reference solution in Ref. [36, Problem 14] gives the eigenvalues of 0.99631 and 1.01531; the errors are, therefore, 3 and 12 pcm, respectively.

Subsequent transient analysis is conducted with $N = 10$, discussed in the following subsection.

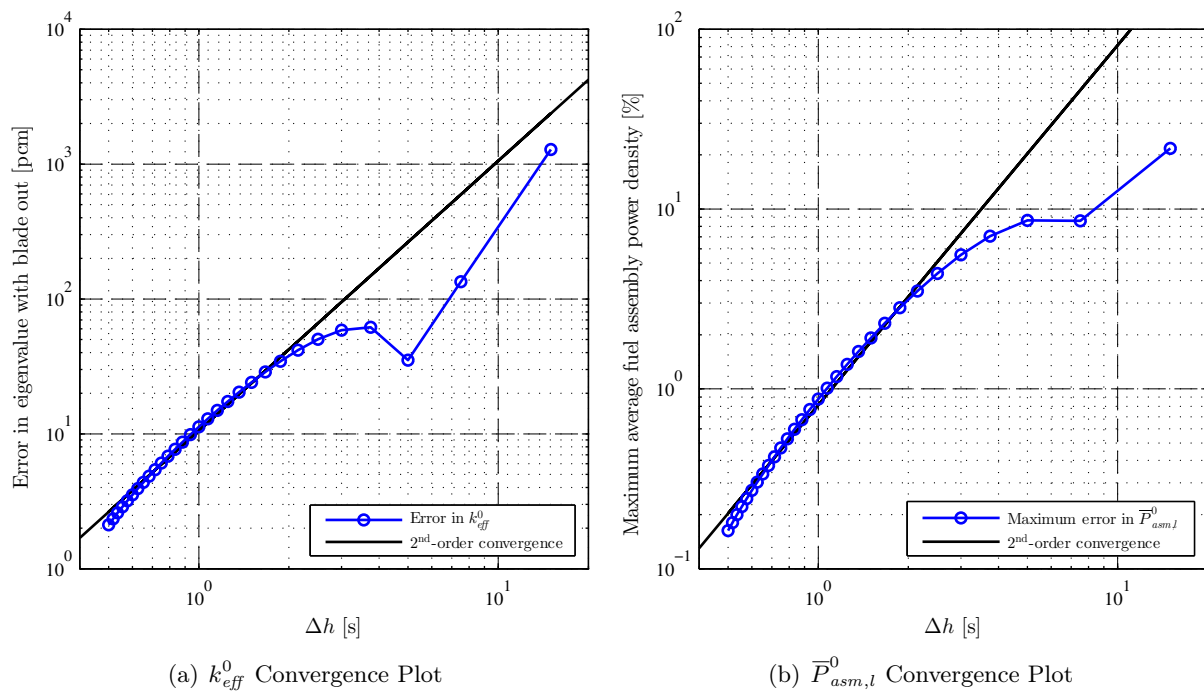


Figure 6.6: 2D LRA BWR Steady State Spatial Convergence Plots

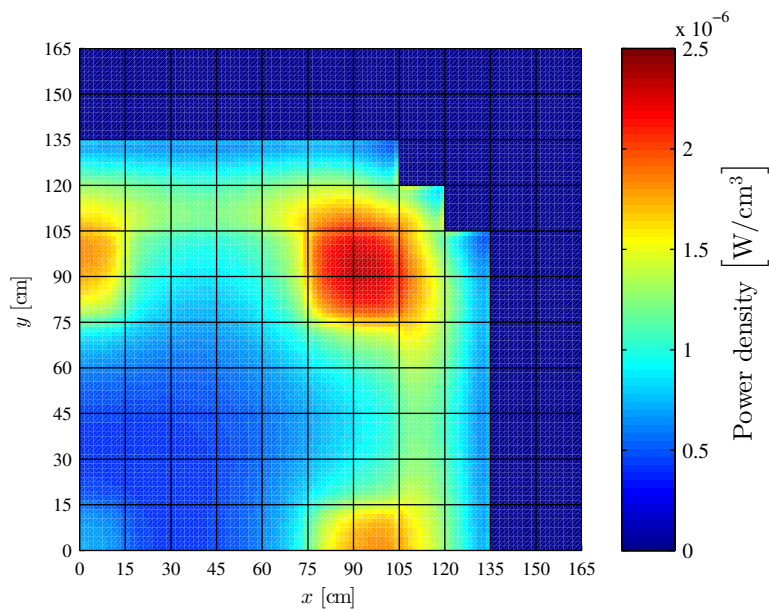


Figure 6.7: 2D LRA BWR Fine Mesh Initial Power Density (normalized to $\bar{P}_{core}^0 = 1.0 \times 10^{-6} W/cm^3$, $\Delta h = 1.5$ cm)

6.3.2 Transient Results

EGSolver v2.0 was set up without split operator integration capabilities, and intended to work only with the BDF3 and MATLAB's adaptive `ode15s` time integrators. The intent of the fine mesh transient analysis was therefore only to verify that the code is converged with a coarse time step, and that the quantities of interest are approximately equal, within a tolerance, to the reference solutions. Figure 6.8 shows the obtained solutions with $\Delta h = 1.5$ cm.

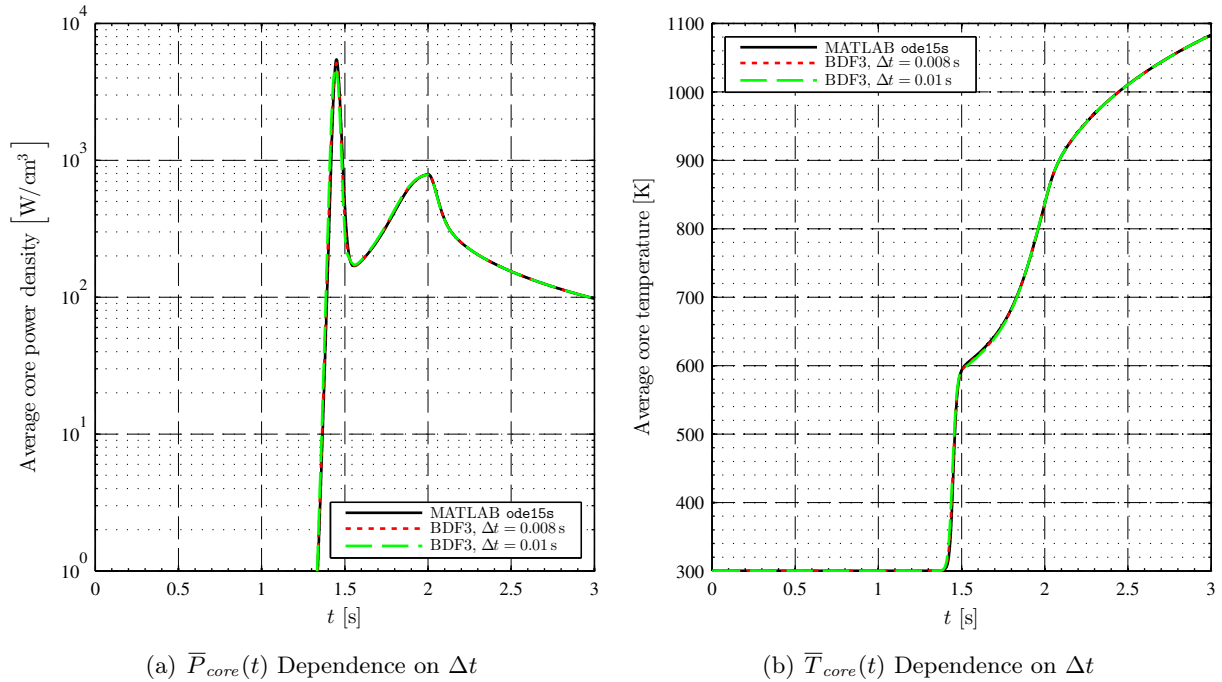


Figure 6.8: 2D LRA BWR Fine Mesh Transient Solutions
(with $\Delta h = 1.5$ cm)

These transient solutions are both physically reasonable, and correspond closely with the reference solutions (under 1% error). Notably, $\Delta t = 8$ ms appears to be closely following the solution even near the power peak, and is therefore, most likely, sufficient for the transient in question. Split operator methods, both in the coarse mesh study above, and in a reference solution that deliberately optimized time steps (Ref. [121]), required $\Delta t = 1$ ms time steps near the peak. This is nearly an order of magnitude improvement in required time step size, which is significant.

6.3.3 Convergence Results

Here, only the peak average core power density is studied, because it is the far more sensitive parameter. Fully coupled BDF3 with $\Delta t = 0.1$ ms was once again used as the reference solution, but less time steps were tried, as this was only a verification study. The true peak power was once again estimated via cubic interpolation. Figure 6.9 illustrates the observed convergence.

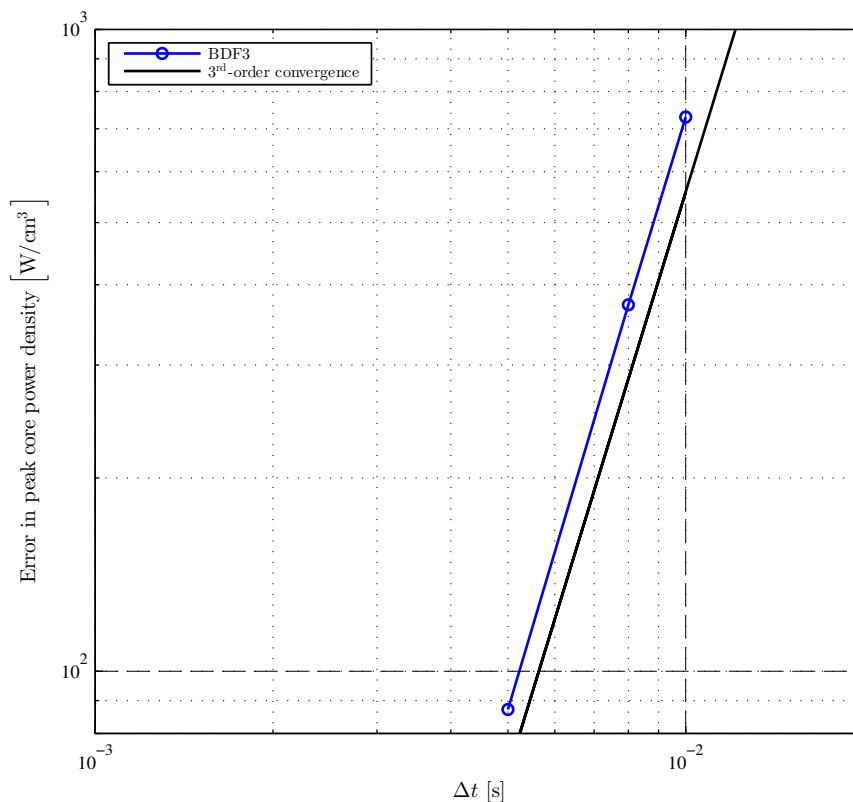


Figure 6.9: 2D LRA BWR Fine Mesh $\bar{P}_{core}(t_{peak})$ Temporal Convergence Plot (with $\Delta h = 1.5$ cm)

The findings of this study fully repeat the findings of the coarse mesh study in section 6.2: superlinear 3rd-order convergence is present and $\Delta t = 8$ ms is once again sufficient for BDF3 with FC. BGSolver v2.0, with $N = 10$ (47601 unknowns), performed the time integration, approximately as fast as BGSolver v1.03 with $N = 1$, while the sorting took about 1 s. The codes' performance times are discussed in more detail in the following subsection.

6.4 Code Performance

The code's performance is summarized in Table 6.3.

These performance times indicate that:

1. Matrix-based sorting is no longer the bottleneck, and majority of the wall clock time is spent on the actual time integration. It also significantly speeds up the state derivative evaluation.
2. With matrix-based sorting, state derivative evaluation time appears to scale linearly with the number of unknowns, which is very important: it means that iterative, Jacobian-free nonlinear solvers (e.g., JFNK [23]) are expected to scale linearly as well, which is necessary for high scalability.

Table 6.3: BGSolver v1.03 and v2.0 Performance on the 2D LRA BWR Benchmark

Sorting method ^a	Δh [cm]	$N_{\vec{x}}$ ^b	Sorting time [s]	Single \vec{x}	N_t for BDF3 ^c	Integration time [s]	
				evaluation time [s]		(BDF3)	(ode15s)
Symbolic	15.0	478	373.0	0.080	600	302.0	
					1200	531.0	
Matrix-based	15.0	477	0.8	0.003	600	2.3	1.1
Matrix-based	1.5	47 601	2.5	0.011	300	604.5	56.3
					600	1329.9	
					30 000	40 725.2	

^a BGSolver v1.03 used for symbolic sorting, BGSolver v2.0 used for matrix-based sorting.

^b Number of unknowns in the state vector, with 5 unknowns per fuel mesh element, 2 unknowns per reflector mesh element, and an additional eigenvalue unknown. For symbolic sorting, the external source is also an additional unknown.

^c Number of equal fixed time steps for BDF3 time integrator. ode15s is adaptive.

Note: The runs were conducted on an Intel i7-2600 3.4 GHz machine with Windows 7 x64 and MATLAB R2013a x64.

- BGSolver v2.0, with a matrix-based sorter, is a reasonable bond graph processing code to use on 2D reactivity-initiated multigroup full core transients. The new bottleneck now appears to be the nonlinear solvers employed by the time integrators, and the overall memory management performance, which cannot be improved without switching from MATLAB to a higher-performance, fully compiled language.

The findings of the 2D LRA BWR control blade drop problem study with BGSolver v1.03 and BGSolver v2.0 are summarized in the next chapter.

6.5 Summary

Two bond graph processing codes, BGSolver v1.03 and v2.0, using symbolic and matrix-based sorting, respectively, were used to solve the 2D LRA BWR control blade drop problem. The bond graph representation techniques used to represent the multidimensional, multigroup neutron diffusion with precursors and adiabatic thermal feedback, were confirmed to be correctly derived and accurate. The matrix-based sorting algorithm proved to be sufficiently fast to completely eliminate the sorting bottleneck that symbolic sorting created in bond graph processing, and therefore justified the use of bond graphs for fully coupled simulation of the problem in question.

Time steps on the order of 5 ms to 8 ms were deemed to be sufficient to converge the transient within 2% of peak power and less than 1 K of final average core temperature, respectively. The desired 3rd-order superlinear convergence was observed. In comparison, split operator time integrators required approximately 1 ms time steps for the same convergence accuracy. This, once again, confirmed that fully coupled time integration is fully justified for use on reactivity-initiated accidents, because of the nearly 1 order of magnitude improvement it yielded for required time steps.

While `BGSolver` v2.0 is a good potential code for bond graph-based algorithm development and prototyping, a higher-performance bond graph processing code is necessary to address larger, more realistic benchmarks. `Larch`, a C++ Trilinos-based bond graph processing library was developed for this purpose; it uses the same processing algorithms as `BGSolver` v2.0, and its underlying philosophy is described in subsection 4.1.2. In the next chapter, the 3D LRA BWR problem is analyzed using `Larch`.

Chapter 7

Three-dimensional BWR Control Blade Drop Problem

`BGSolver` v2.0, a MATLAB-based bond graph processing code with matrix-based sorting, was successfully used to model the fine-mesh 2D LRA BWR control blade drop problem. Real reactor core reactivity-initiated accidents are three-dimensional problems, and `BGSolver` v2.0, due to its reliance on MATLAB, is limited in the size of problems it can run; it is therefore unfit for all but the most basic 3D reactivity-initiated problems.

This is not, fundamentally, a bond graph-related limitation, therefore the formalism can be used in a higher-performance code, which can be used to process 3-dimensional full core reactor problems. `Larch`, a C++ Trilinos-based bond graph processing library was developed for this purpose [115]; it uses the same algorithms as `BGSolver` v2.0.

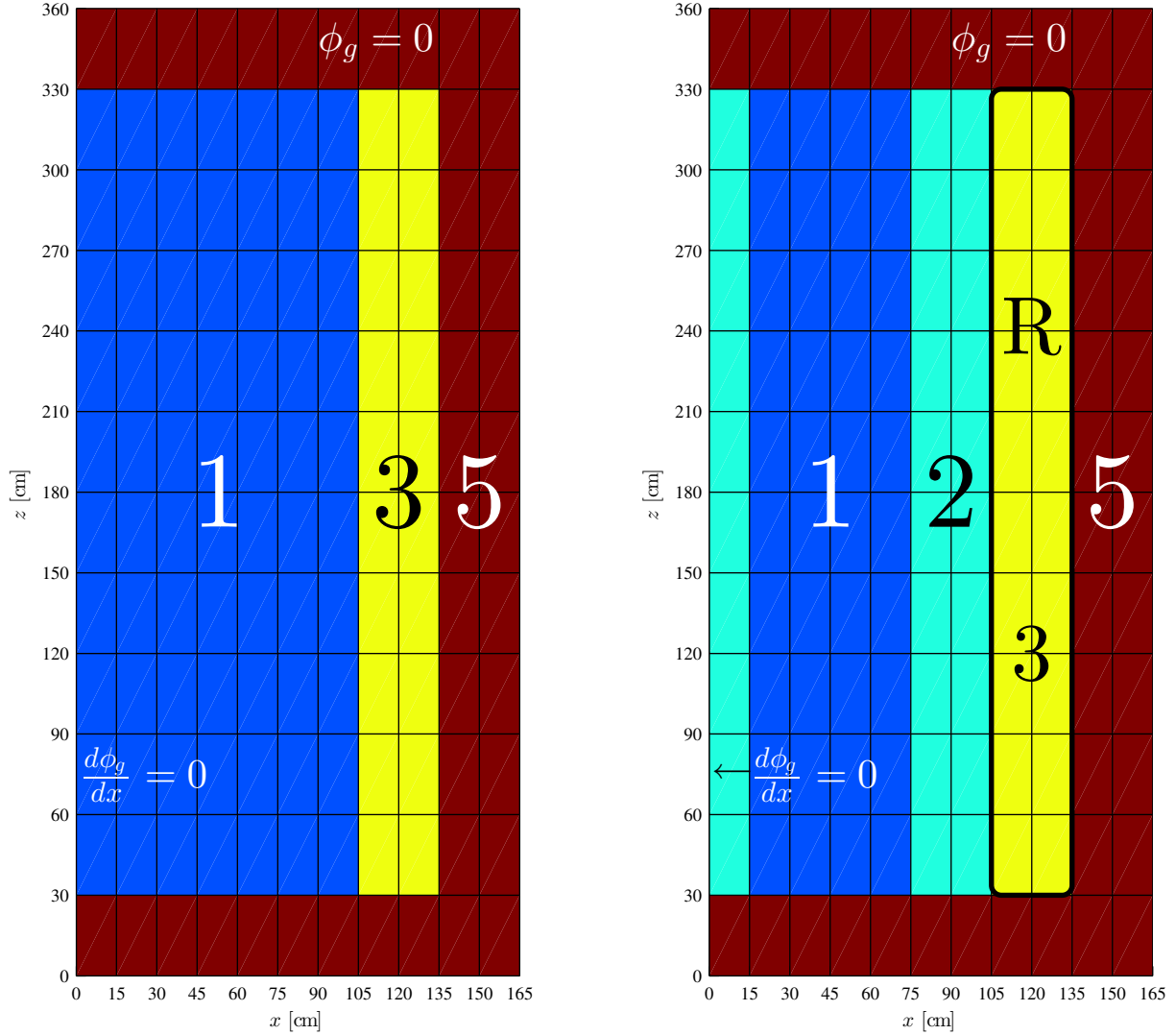
To test `Larch`, and with it, the use of bond graph formalism for realistic 3D reactor problems, a 3D version of the LRA BWR control blade drop problem was chosen. It is defined in section 7.1. Sections 7.2 and 7.3 discuss the steady state search and transient results for this problem, respectively. Section 7.4 summarizes the findings of the 3D problem study.

7.1 Benchmark Problem Definition

The 3D LRA BWR control blade drop problem is an axial (z -axis) extrusion of the two-dimensional geometry described in section 6.1, with a 30 cm reflector (region 5) above and below the resulting core. No axial variation in nominal core properties is present. Figure 7.1 illustrates the vertical slices through the core at $y = 45$ cm and $y = 90$ cm.

Figure 7.1(b) illustrates that the control blade 30 cm \times 30 cm region is now modeled as extending through the entire core, over the same 4 region 3 fuel assemblies as in the 2D LRA BWR problem. The control blade movement is now modeled differently. The blade is assumed to move downward with a fixed axial velocity of 150 cm/s (thereby clearing the core in 2 s):

$$z_{cb}(t) = \begin{cases} 330 \text{ cm} - (150 \text{ cm/s}) \cdot t & \text{if } t \leq 2 \text{ s,} \\ 30 \text{ cm} & \text{if } t > 2 \text{ s.} \end{cases} \quad (7.1)$$


 (a) Axial Layout at $y = 45$ cm

 (b) Axial Layout at $y = 90$ cm

Figure 7.1: 3D LRA BWR Benchmark Problem Axial Specifications

An affected axial node's thermal absorption cross section is then modeled by:

$$\Sigma_{a2,R,k}(t) = \begin{cases} \Sigma_{a2,3,k}^0 & \text{if } z_{bot,k} + \Delta z < z_{cb}(t), \\ \Sigma_{a2,3,k}^0 \cdot \left[1 - (0.0606184 \text{ s}^{-1} \cdot 2 \text{ s}) \times \right. \\ \quad \left. \times \left(1 - \frac{z_{cb}(t) - z_{bot,k}}{\Delta z} \right) \right] & \text{if } z_{bot,k} < z_{cb}(t) \leq z_{bot,k} + \Delta z, \\ \Sigma_{a2,3,k}^0 \cdot 0.8787631 & \text{if } z_{cb}(t) \leq z_{bot,k}. \end{cases} \quad (7.2)$$

The following notation was used:

$z_{cb}(t)$	= Height of the control blade tip from the bottom of the reactor. Units: cm.
$\Sigma_{a2,R,k}(t)$	= Macroscopic thermal absorption cross section of an axial node k in region R (through which the control blade moves). The index k specifies the axial position of the node. Units: cm^{-1} .
$\Sigma_{a2,3,k}^0$	= Nominal macroscopic thermal absorption cross section of an axial node k in region 3, which is initially identical to thermal absorption cross section in region R. Units: cm^{-1} .
$z_{bot,k}$	= Height of the bottom of axial node k from the bottom of the reactor. Units: cm.
Δz	= Axial dimension of a single node. Units: cm.

It is important to note that many reference solutions in literature (e.g., Sutton and Aviles [44]), when referring to the “3D LRA problem,” solve a true full core problem, and not the quarter-core addressed here. Such configuration leads to a more significant radial tilt after the blade drop, and is clearly a larger (by a factor of 4) problem with a different solution. Care should be taken to ensure that the correct configuration is being looked at when comparing 3D LRA BWR benchmark solutions.

The physics of the problem are exactly the same as in the 2D LRA BWR problem, with one modification: axial buckling term $B_z^2 D_{g,r} \phi_g(t, \vec{r})$ is not present, because \vec{r} is now 3-dimensional. The same bond graph representation, developed in section 3.3, is used here. The only difference is the dimensionality of the neutron currents and reaction rates, because volumes and nodal surface areas now have units of cm^3 and cm^2 , respectively.

The same initial conditions as in chapter 6 are used here.

7.2 Steady State Search

The coarsest possible mesh element that does not require additional property averaging is $15 \text{ cm} \times 15 \text{ cm}$ in the xy -plane, and 30 cm in the z -direction. There is more property variation in the x - and y -directions than in the z -direction, and, together with the shorter effective core length ($2 \times 135 \text{ cm} = 270 \text{ cm}$ radially vs 300 cm axially), this leads to the problem being much more sensitive to radial, than axial, discretization.

For this reason, an intermediate spatial mesh discretization of $5 \text{ cm} \times 5 \text{ cm} \times 30 \text{ cm}$ was used. As the spatial convergence study in subsection 6.3.1 showed, while the problem is not completely converged with such horizontal mesh ($\Delta h = 5 \text{ cm}$), the error in k_{eff} for a withdrawn blade is, at most, on the order of 30 pcm for the 2D problem. As will be discussed below, the error for the 3D problem is comparable: 39 pcm and 29 pcm errors for the inserted and withdrawn blade configurations, respectively, compared to the reference solutions by Smith [121]. This mesh, therefore, cannot be considered fully spatially converged, but it is sufficiently close to test the 3D bond graph representation of the problem, its processing through `Larch`, and the performance of fully coupled time integrators on 3D spatial kinetics problems with feedback.

`Larch` eigenproblem solving capability is similar to `BGSolver` v2.0. Modulated elements’ constituent expressions can be functions of explicitly specified parameters, such as eigenvalues (or other parameters, such as depletion stage or material type). An eigenvalue solve is executed by specifying a residual function for the core average power density, and by specifying the parameter that is the eigenvalue (here it is the only parameter present, but in other problems, additional non-

eigenvalue parameters may be present). Nominal thermal energy densities (corresponding to T^0 are also specified. A nonlinear solver then solves for the flux and precursor density eigenfunctions and k_{eff}^0 .

With this spatial discretization, using the built-in eigenproblem solving capability of `Larch`, $k_{eff}^0 = 0.99688$ was obtained, compared to 0.99648 in Ref. [121]. The initial power profiles at the vertical slices through the core at $y = 45$ cm and $y = 90$ cm are given in Figure 7.2.

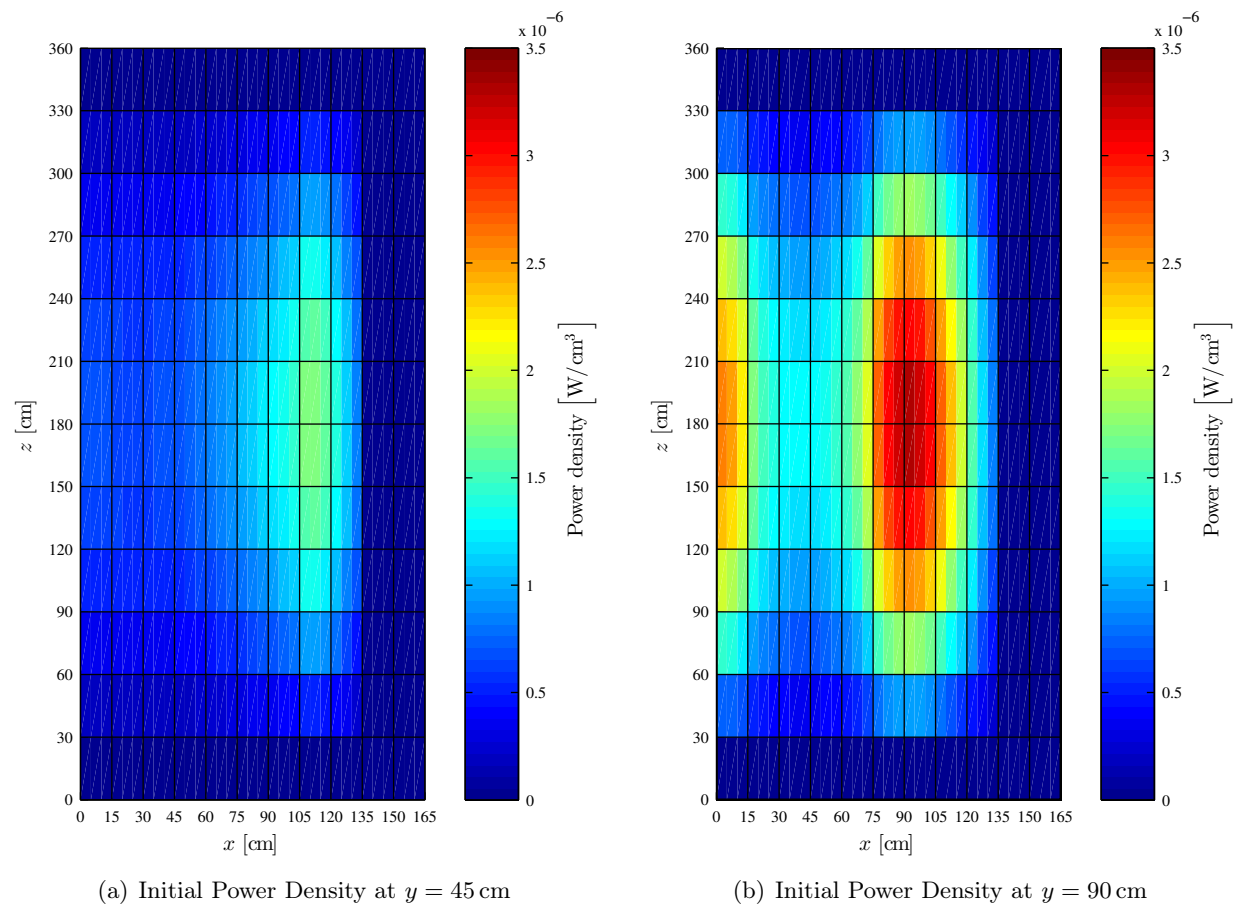


Figure 7.2: 3D LRA BWR Initial Power Density
 (normalized to $\bar{P}_{core}^0 = 1.0 \times 10^{-6} \text{ W/cm}^3$, $\Delta h = 5$ cm, $\Delta z = 30$ cm)

The results of the transient analysis, based on this initial condition, are given in section 7.3.

7.3 Results

Like `BGSolver` v2.0, `Larch` was set up without split operator integration capabilities. Its time integrator is a simple BDF3 with a fixed time step, documented in section 4.4. The time integrator relies on the Trilinos package `Nox` to solve the nonlinear algebraic problem at every time step using a form of Newton's method; the linear systems' solutions at every time step, here, were implemented using the Amesos package with a direct linear solver. BDF3 solution with $\Delta t = 0.1$ ms was used as the reference solution.

Figure 7.3 shows the transient solutions to the 3D LRA BWR control blade drop problem.

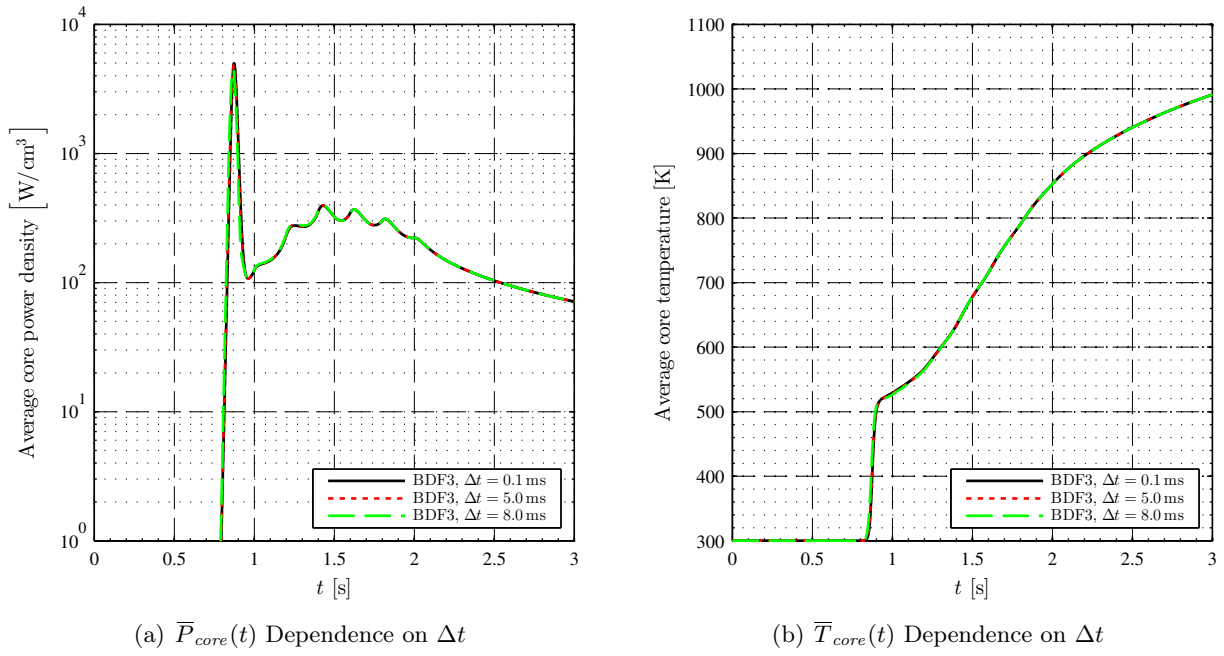


Figure 7.3: 3D LRA BWR Transient Solutions
(with $\Delta h = 5$ cm, $\Delta z = 30$ cm)

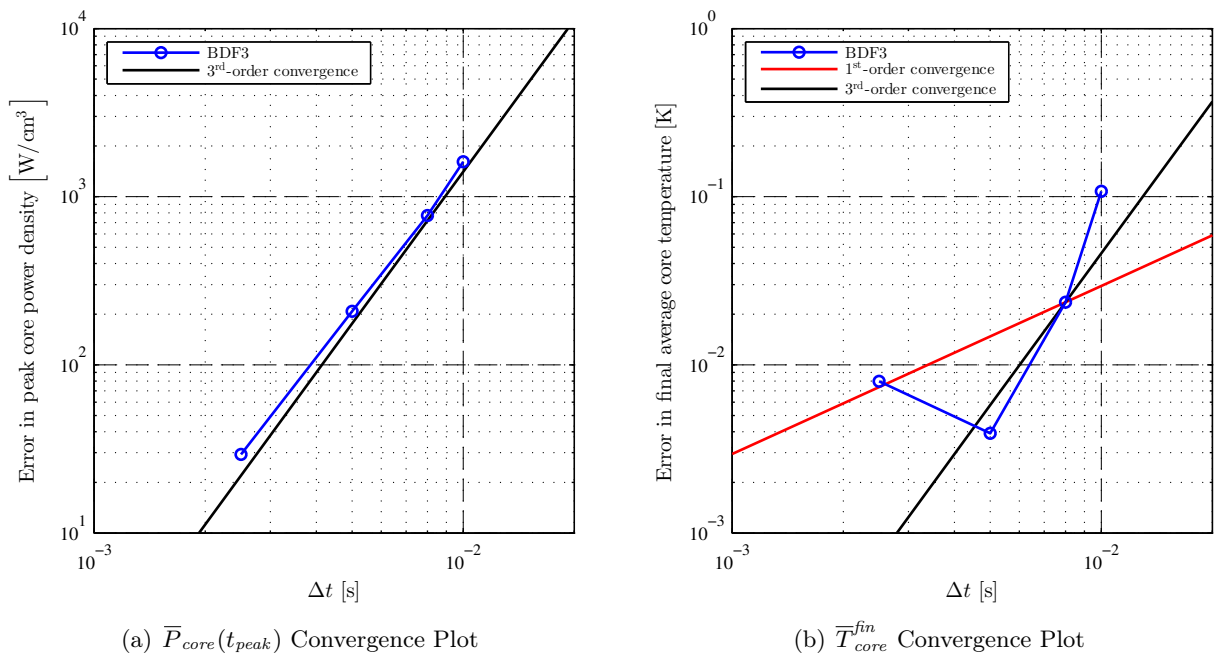


Figure 7.4: 3D LRA BWR Temporal Convergence Plots
(with $\Delta h = 5$ cm, $\Delta z = 30$ cm)

The transient solutions appear to be almost completely converged at a coarse time step of 5 ms. Due to the relatively coarse mesh used, there is a 12 % error in peak power and 3 % error in final core average fuel temperature, relative to the reference solutions in Ref. [121]. These errors are due to the coarseness of the spatial mesh.

The oscillations in peak power between the first peak and the end of the control blade drop (approximately $t = 0.874$ s to $t = 2.0$ s) are nonphysical, but are present in all solutions with this type of spatial homogenization, due to a phenomenon called “blade cusping” (Ref. [121] refers to it as “rod cusping”). When the blade tip crosses over a node boundary, the rate of reactivity insertion briefly reduces, before rising linearly again. This phenomenon can only be present in a 3D problem; it can be addressed by using spatial discretization methods that allow for spatial variation in homogenized properties across a node.

Figure 7.4 illustrates the time convergence of the quantities of interest, obtained using the BDF3 fully coupled time integrator.

Peak core average power density clearly demonstrates the desired 3rd-order convergence. However, the final core average temperature results, while converged (0.1 K is a negligibly small error), do not appear to converge superlinearly: the overall order of convergence is closer to linear. This is, most likely, due to the fact that, like in the PWR PKE VHZP final temperature convergence case (subsection 5.3.2), the time-dependent initiating perturbation is not smooth. This is also the case in the 2D LRA BWR control blade drop problem, but there, the two 1st-order discontinuities (discontinuities in property time derivatives) are only present at $t = 0$ s and $t = 2$ s. Here, however, a discontinuity in properties is present every time the blade crosses a node’s boundary, which occurs every 0.2 s. This results in the observed lack of reliable superlinear convergence. Again, this is unlikely to be an issue, because the solution is still clearly converged with a coarse time step of 8 ms; the reference solution in Ref. [121] used 1 ms time steps near the power peak.

These results indicate that while 3rd-order convergence of sensitive quantities of interest cannot be guaranteed with nonsmooth time-dependent properties (and, from PWR PKE case, with nonsmooth properties in general), convergence with coarse time steps of 5 ms to 8 ms can still be obtained. This is a clear, reliable advantage of fully coupled time integrators, and with them, of the bond graph formalism.

The bottlenecks of the bond graph processing codes `BGSolver` v1.03 and v2.0 were the symbolic sorting and overall memory management performance, respectively. `Larch`’s bottleneck appears to be the direct linear solvers employed by the nonlinear solver in the BDF3 implicit time integrator; due to the increased band width of the 3D problem’s Jacobian (11 versus 9 nonzero row elements for an interior mesh element), the direct linear solver takes over 90 % of the time integrator’s execution time, and does not scale efficiently. To further increase the size of the problems that `Larch` could address, a fully iterative nonlinear solver, such as JFNK, should be implemented for the BDF3 time integrator.

7.4 Summary

A C++ Trilinos-based bond graph processing library was developed and used to solve the 3D LRA BWR control blade drop problem. The bond graph representation techniques used to represent the multidimensional, multigroup neutron diffusion with precursors and adiabatic thermal feedback, were confirmed to be correctly derived and accurate for 3D spatial kinetics problems. The matrix-

based sorting algorithm worked very efficiently, taking less than a second to process the problem of interest with 47 200 unknowns.

Time steps on the order of 5 ms to 8 ms were again sufficient to converge the transient within 2% of peak power and less than 1 K of final average core temperature, respectively. The desired 3rd-order superlinear convergence was observed for peak power, but not for temperature; again, both quantities could still be considered converged at the above coarse time steps. The coarser time step requirements are, again, an improvement over 1 ms time steps that split operator time integrators required. This is the final, and definitive, confirmation of the fact that full coupling is beneficial even with nonsmooth properties, although superlinear convergence may not be obtainable.

Bond graph formalism was confirmed to accurately and simply represent the 3D LRA BWR problem, and can therefore be recommended, if used with a sufficiently high performance bond graph processing code, to be used as the approach to 3D spatial kinetics code development with thermohydraulic feedback. A time integrator which uses a fully iterative nonlinear solver, such as JFNK, must be used, otherwise the code may not be sufficiently scalable, even if it allows the use of coarse time steps.

Chapter 8

Conclusions and Recommendations for Future Work

All objectives specified in section 1.3 were successfully achieved:

1. Bond graph formalism was developed sufficiently to be able to model a variety of realistic reactor multiphysics transient problems. To do this, a MATLAB symbolic engine-based bond graph processing code was extended to support a new element type — the multiport resistor — which was used to represent multidimensional, multigroup neutron diffusion with precursors. To scale the code, the symbolic engine was eventually phased out; a new, fully numeric and high performance bond graph processing algorithm was developed, and used as the basis for two bond graph processing codes.
2. Several reactivity-initiated reactor multiphysics benchmarks were successfully analyzed using the methods developed.
3. Full coupling was confirmed to be significantly superior to split operator time integration in all observed cases. A factor of 5 to 8 minimum time step improvement was consistently obtained over the split operator time integrators.

These outcomes are detailed below.

8.1 Viability of Bond Graphs for Reactor Multiphysics Analysis

Bond graph representation is clearly a viable approach to the implementation of fully coupled reactor multiphysics simulation codes, assuming the matrix-based sorting algorithm is used. With a sufficiently high performance bond graph processing code or library, the approach is capable of quickly and efficiently formulating and integrating the fully coupled state derivative vector. This was evidenced by the successful solutions of several reactor transients, solved using bond graphs over the course of this work.

Bond graphs allow the use of fully coupled time integrators, which have demonstrated considerable increase in performance, by coarsening the required time step, over split operator time integrators. For the spatial kinetics problems analyzed, 8 ms time step was required for 2% error in peak power, compared to approximately 1 ms time step for the split operator time integrators. For other problems, comparable speedup was observed, including in cases where the

lack of smoothness of the initiating perturbation or the properties prevented actual superlinear convergence from occurring.

It can therefore be clearly concluded that fully coupled, implicit, superlinear multistep time integrators benefit the required time step size nearly regardless of the problem configuration, although the degree of benefit can vary. With high-performance bond graph processing codes, bond graphs are a competitive way to implement these fully coupled state derivative vectors and time integrators.

8.2 Viability of Large-scale Automated Bond Graph Processing

Regardless of the size of the problem analyzed (up to approximately 50 000 unknowns), the state derivative function produced by the matrix-based sorting algorithm evaluated in several (i.e., 3–11) milliseconds, or faster. The sorting itself, for a problem this size, took approximately 2.5 s using the MATLAB bond graph processing code, and is about an order of magnitude faster using the C++ bond graph processing code. It can therefore be clearly concluded that the matrix-based sorting algorithm, in its present form, is fit for processing large scale problems.

The benchmark problems, postulated at the beginning of the project, were all solved successfully. To be able to solve larger, more detailed problems, the focus should transfer from bond graph processing and representation itself to transitioning to better algebraic solvers, which take the bulk (over 90%) of the time for the code.

8.3 Recommendations for Future Work

The following steps are recommended to continue the development of bond graph modeling of nuclear reactor multiphysics:

1. Iterative linear solvers must be introduced into `Larch`, the C++ Trilinos-based bond graph processing library developed in this project. `Larch`'s nonlinear solver currently uses a direct linear solver for its Newton iterations, which is acceptable for 2D problems, but is a significant bottleneck for 3D problems.
2. True modern multiphysics frameworks require parallelization. The matrix-based sorting algorithms are currently fully serial, and therefore need to be parallelized to expand the formalism past serial computations.
3. The reactor bond graph representations developed in this work were sufficient for the benchmarks problems, and problems with comparable, simplified physics. More realistic thermal hydraulic and spatial kinetics models must be represented with bond graphs: systems-level multiphase flow, and more advanced nodal methods, are a good place to start.

The most important recommendation is the first one: a more scalable iterative linear solver that makes use of the fast state derivative vector evaluation times, would greatly benefit the code in its present state. Once this bottleneck is removed, the other tasks may be addressed.

References

- [1] H. M. PAYNTER. *Analysis and Design of Engineering Systems*. The MIT Press, Cambridge, MA, USA, 1961.
- [2] W. M. STACEY. *Nuclear Reactor Physics*. 2nd edition. Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, Germany, 2007.
- [3] N. E. TODREAS AND M. S. KAZIMI. *Nuclear Systems*, volume 1. 2nd edition. CRC Press, Boca Raton, FL, USA, 2012.
- [4] A. HÉBERT. *Applied Reactor Physics*. Presses Internationales Polytechnique, Montreal, Quebec, Canada, 2009.
- [5] A. F. HENRY. *Nuclear-Reactor Analysis*. The MIT Press, Cambridge, MA, USA, 1975.
- [6] J. J. DUDERSTADT AND L. J. HAMILTON. *Nuclear Reactor Analysis*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 1976.
- [7] ANSI/ANS-5.1-2005, *Decay heat power in light water reactors*. American national standard. American Nuclear Society, La Grange Park, IL, USA, 2005.
URL: http://www.ans.org/store/i_240256.
- [8] E. SHWAGERAUS AND E. FRIDMAN. *Decay power calculation for safety analysis of innovative reactor systems*. In: *International Youth Nuclear Congress*. Interlaken, Switzerland, September 20–26, 2008.
URL: http://inis.iaea.org/search/search.aspx?orig_q=RN:40048094.
- [9] A. TOBIAS. *Decay heat*. *Progress in Nuclear Energy*, **5**(1): pp. 1–93, 1980.
DOI: 10.1016/0149-1970(80)90002-5.
URL: [http://dx.doi.org/10.1016/0149-1970\(80\)90002-5](http://dx.doi.org/10.1016/0149-1970(80)90002-5).
- [10] A. TOBIAS. *Errata to: Decay heat*. *Progress in Nuclear Energy*, **5**(3): p. 283, 1980.
DOI: 10.1016/0149-1970(80)90009-8.
URL: [http://dx.doi.org/10.1016/0149-1970\(80\)90009-8](http://dx.doi.org/10.1016/0149-1970(80)90009-8).
- [11] A. L. NICHOLS. *Nuclear data requirements for decay heat calculations*. ICTP Lecture Notes Series, **20**: pp. 67–195, 2005.
URL: <http://publications.ictp.it/lms/vol20.html>.
- [12] P. K. KUNDU AND I. M. COHEN. *Fluid Mechanics*. 4th edition. Academic Press, Burlington, MA, USA, 2008.

- [13] J. C. NEU. *Training Manual on Transport and Fluids, Graduate Studies in Mathematics*, volume 109. American Mathematical Society, Providence, RI, USA, 2010.
- [14] S. M. GHIAASIAAN. *Two-Phase Flow, Boiling and Condensation in Conventional and Miniature Systems*. Cambridge University Press, New York, NY, USA, 2008.
- [15] J. T. ADRIAN ROBERTS. *Structural Materials in Nuclear Power Systems*. Plenum Press, New York, NY, USA, 1981.
- [16] D. M. FRYER AND J. F. HARVEY. *High Pressure Vessels*. Chapman & Hall, New York, NY, USA, 1998.
- [17] K.-J. BATHE. *Finite Element Procedures*. Prentice Hall, Upper Saddle River, NJ, USA, 1996.
- [18] M. K. AU-YANG. *Flow-Induced Vibration of Power and Process Plant Components: A Practical Workbook*. ASME Press, New York, NY, USA, 2001.
- [19] ASTM E693-12, *Standard practice for characterizing neutron exposures in iron and low alloy steels in terms of displacements per atom (DPA), E 706(ID)*. ASTM Standard. ASTM International, West Conshohocken, PA, USA, 2012.
DOI: 10.1520/E0693-12.
URL: <http://dx.doi.org/10.1520/E0693-12>.
- [20] H. R. HIGGY AND F. H. HAMMAD. *Effect of neutron irradiation on the tensile properties of Zircaloy-2 and Zircaloy-4*. *Journal of Nuclear Materials*, **44**: pp. 215–227, 1972.
DOI: 10.1016/0022-3115(72)90099-2.
URL: [http://dx.doi.org/10.1016/0022-3115\(72\)90099-2](http://dx.doi.org/10.1016/0022-3115(72)90099-2).
- [21] B. M. GORDON. *Introduction to corrosion in the nuclear power industry*. In: S. D. CRAMER AND B. S. COVINO, JR. (editors), *Corrosion: Environments and Industries, ASM Handbook*, volume 13C, 10th edition, pp. 339–340. ASM International, Materials Park, OH, USA, 2006.
URL: <http://products.asminternational.org/hbk/>.
- [22] R. A. CASTELLI. *Nuclear Corrosion Modeling: The Nature of CRUD*. Butterworth-Heinemann, Burlington, MA, USA, 2009.
DOI: 10.1016/B978-1-85617-802-0.00016-5.
URL: <http://dx.doi.org/10.1016/B978-1-85617-802-0.00016-5>.
- [23] D. A. KNOLL AND D. E. KEYES. *Jacobian-free Newton-Krylov methods: A survey of approaches and applications*. *Journal of Computational Physics*, **193**(2): pp. 357–397, 2004.
DOI: 10.1016/j.jcp.2003.08.010.
URL: <http://dx.doi.org/10.1016/j.jcp.2003.08.010>.
- [24] M. B. CHADWICK, M. HERMAN, P. OBLOŽINSKÝ, M. E. DUNN, Y. DANON, A. C. KAHLER, D. L. SMITH, B. PRITYCHENKO, G. ARBANAS, R. ARCILLA, R. BREWER, D. A. BROWN, R. CAPOTE, A. D. CARLSON, Y. S. CHO, H. DERRIEN, K. GUBER, G. M. HALE, S. HOBLIT, S. HOLLOWAY, T. D. JOHNSON, T. KAWANO, B. C. KIEDROWSKI, H. KIM, S. KUNIEDA, N. M. LARSON, L. LEAL, J. P. LESTONE, R. C. LITTLE, E. A.

- MCCUTCHAN, R. E. MACFARLANE, M. MACINNES, C. M. MATTOON, R. D. MCKNIGHT, S. F. MUGHABGHAB, G. P. A. NOBRE, G. PALMIOTTI, A. PALUMBO, M. T. PIGNI, V. G. PRONYAEV, R. O. SAYER, A. A. SONZOGNI, N. C. SUMMERS, P. TALOU, I. J. THOMPSON, A. TRKOV, R. L. VOGT, S. C. VAN DER MARCK, A. WALLNER, M. C. WHITE, D. WIARDA AND P. G. YOUNG. *ENDF/B-VII.1 nuclear data for science and technology: Cross sections, covariances, fission product yields and decay data*. Nuclear Data Sheets, **112**(12): pp. 2887–2996, 2011.
DOI: 10.1016/j.nds.2011.11.002.
URL: <http://dx.doi.org/10.1016/j.nds.2011.11.002>.
- [25] P. J. MOHR, B. N. TAYLOR AND D. B. NEWELL. *CODATA recommended values of the fundamental physical constants: 2010*. Reviews of Modern Physics, **84**(4): pp. 1527–1605, 2012.
DOI: 10.1103/RevModPhys.84.1527.
URL: <http://dx.doi.org/10.1103/RevModPhys.84.1527>.
- [26] A. KONING, R. FORREST, M. KELLETT, R. MILLS, H. HENRIKSSON AND Y. RUGAMA. *The JEFF-3.1 nuclear data library*. JEFF Report 21, OECD/NEA, Paris, France, 2006.
URL: http://www.oecd-nea.org/dbdata/nds_jefreports/jefreport-21/.
- [27] R. E. MACFARLANE, D. W. MUIR, R. M. BOICOURT AND A. C. KAHLER. *The NJOY nuclear data processing system, version 2012*. Technical Report LA-UR-12-27079 Rev, Los Alamos National Laboratory, Los Alamos, NM, USA, 2013.
URL: <http://t2.lanl.gov/nis/codes/NJOY12/>.
- [28] R. E. MACFARLANE AND A. C. KAHLER. *Methods for processing ENDF/B-VII with NJOY*. Nuclear Data Sheets, **111**(12): pp. 2739–2890, 2010.
DOI: 10.1016/j.nds.2010.11.001.
URL: <http://dx.doi.org/10.1016/j.nds.2010.11.001>.
- [29] J. RHODES, K. SMITH AND D. LEE. *CASMO-5 development and applications*. In: *PHYSOR*. Vancouver, BC, Canada, September 10–14, 2006.
- [30] A. YAMAMOTO, M. TABUCHI, N. SUGIMURA, T. USHIO AND M. MORI. *Derivation of optimum polar angle quadrature set for the method of characteristics based on approximation error for the Bickley function*. Journal of Nuclear Science and Technology, **44**(2): pp. 129–136, 2007.
DOI: 10.1080/18811248.2007.9711266.
URL: <http://dx.doi.org/10.1080/18811248.2007.9711266>.
- [31] B. G. CARLSON. *On a more precise definition of the discrete ordinates method*. In: *Second Conference on Transport Theory*. Los Alamos, NM, USA, January 26–29, 1971.
URL: <http://www.osti.gov/scitech/biblio/4074311>.
- [32] J. P. JENAL, P. J. ERICKSON, W. A. RHOADES, D. B. SIMPSON AND M. L. WILLIAMS. *The generation of a computer library for discrete ordinates quadrature sets*. Technical Report ORNL/TM-6023, Oak Ridge National Laboratory, Oak Ridge, TN, USA, 1977.
DOI: 10.2172/5237525.
URL: <http://dx.doi.org/10.2172/5237525>.

- [33] A. HÉBERT. *Multigroup neutron transport and diffusion computations*. In: D. G. CACUCI (editor), *Reactor Design, Handbook of Nuclear Engineering*, volume 2, pp. 751–911. Springer, New York, NY, USA, 2010.
DOI: 10.1007/978-0-387-98149-9_8.
URL: http://dx.doi.org/10.1007/978-0-387-98149-9_8.
- [34] Studsvik Scandpower, Inc. *CASMO-4E: Extended capability CASMO-4 user's manual — a university release*, Report SSP-09/442-U Rev 0. Idaho Falls, ID, USA, 2009.
- [35] G. MARLEAU, A. HÉBERT AND R. ROY. *A user guide for DRAGON Version4*. Technical Report IGE-294, École Polytechnique de Montréal, Montreal, Quebec, Canada, 2013.
URL: <http://www.polymtl.ca/merlin/version4.htm>.
- [36] D. F. BIRON, P. T. CHOONG, G. W. CUNNINGHAM, J. H. L. DODDS, V. J. ESPOSITO, H. FINNEMANN, T. B. FOWLER, E. L. FULLER, M. V. GREGORY, K. KOEBKE, S. A. KUSNERIUK, S. LANGENBUCH, R. R. LEE, A. LEONARD, S. K. LOYALKA, A. N. MALLEEN, C. T. MCDANIEL, F. N. MCDONNELL, D. A. MENELEY, B. MICHEELSEN, I. MISFELDT, C. D. MORGAN, A. NERO, F. J. RAHN, D. B. SELBY, R. G. STEINKE, D. R. VONDY, M. R. WAGNER, B. A. WEATE, W. WERNER, W. A. WITTKOPF AND B. A. ZOLOTAR. *Argonne code center: Benchmark problem book*. Technical Report ANL-7416 Suppl.2, Argonne National Laboratory, Argonne, IL, USA, 1977.
DOI: 10.2172/5037820.
URL: <http://dx.doi.org/10.2172/5037820>.
- [37] T. BAHADIR AND S.-Ö. LINDAHL. *Studsvik's next generation nodal code SIMULATE-5*. In: *Advances in Nuclear Fuel Management IV*. Hilton Head Island, SC, USA, April 12–15, 2009.
- [38] K. L. DERSTINE. *DIF3D: A code to solve one-, two-, and three-dimensional finite-difference diffusion theory problems*. Technical Report ANL-82-64, Argonne National Laboratory, Argonne, IL, USA, 1984.
URL: <http://www.osti.gov/scitech/biblio/7157044>.
- [39] A. HÉBERT. *The search for superconvergence in spherical harmonics approximations*. Nuclear Science and Engineering, **154**(2): pp. 134–173, 2006.
URL: http://www.ans.org/pubs/journals/nse/a_2623.
- [40] R. J. LEVEQUE. *Finite Volume Methods for Hyperbolic Problems, Cambridge Texts in Applied Mathematics*, volume 31. Cambridge University Press, Cambridge, UK, 2002.
- [41] A. HÉBERT. *High order diamond differencing schemes*. Annals of Nuclear Energy, **33**(17–18): pp. 1479–1488, 2006.
DOI: 10.1016/j.anucene.2006.10.003.
URL: <http://dx.doi.org/10.1016/j.anucene.2006.10.003>.
- [42] Z. GUO AND S. KUMAR. *Three-dimensional discrete ordinates method in transient radiative transfer*. Journal of Thermophysics and Heat Transfer, **16**(3): pp. 289–296, 2002.
DOI: 10.2514/2.6689.
URL: <http://dx.doi.org/10.2514/2.6689>.

-
- [43] H. G. JOO, J. I. YOON AND S. G. BAEK. *Multigroup pin power reconstruction with two-dimensional source expansion and corner flux discontinuity*. *Annals of Nuclear Energy*, **36**(1): pp. 85–97, 2009.
DOI: 10.1016/j.anucene.2008.10.003.
URL: <http://dx.doi.org/10.1016/j.anucene.2008.10.003>.
- [44] T. M. SUTTON AND B. N. AVILES. *Diffusion theory methods for spatial kinetics calculations*. *Progress in Nuclear Energy*, **30**(2): pp. 119–182, 1996.
DOI: 10.1016/0149-1970(95)00082-U.
URL: [http://dx.doi.org/10.1016/0149-1970\(95\)00082-U](http://dx.doi.org/10.1016/0149-1970(95)00082-U).
- [45] P. J. FINCK, C. L. HOXIE, H. S. KHALIL, D. K. PARSONS AND A. F. HENRY. *The application of nodal methods to light water reactors*. In: *Topical Meeting on Advances in Reactor Physics and Core Thermal Hydraulics*, pp. 348–363. Kiamesha Lake, NY, USA, September 22–24, 1982.
- [46] G. FILEAS. *On numerical methods in non-Newtonian flows*. Technical Report CTH-R-82/07, Chalmers tekniska högskola, Gothenburg, Sweden, 1982.
URL: http://inis.iaea.org/search/search.aspx?orig_q=RN:15009313.
- [47] INTERNATIONAL FORMULATION COMMITTEE OF THE 6TH INTERNATIONAL CONFERENCE ON THE PROPERTIES OF STEAM. *The 1967 IFC formulation for industrial use*. Technical report, Verein Deutscher Ingenieure, Düsseldorf, Germany, 1967.
- [48] W. WAGNER, J. R. COOPER, A. DITTMANN, J. KIJIMA, H.-J. KRETZSCHMAR, A. KRUSE, R. MAREŠ, K. OGUCHI, H. SATO, I. STÖCKER, O. ŠIFNER, Y. TAKAISHI, I. TANISHITA, J. TRÜBENBACH AND T. WILLKOMMEN. *The IAPWS industrial formulation 1997 for the thermodynamic properties of water and steam*. *Journal of Engineering for Gas Turbines and Power*, **122**(1): pp. 150–184, 2000.
DOI: 10.1115/1.483186.
URL: <http://dx.doi.org/10.1115/1.483186>.
- [49] J. K. FINK AND L. LEIBOWITZ. *Thermodynamic and transport properties of sodium liquid and vapor*. Technical Report ANL/RE-95/2, Argonne National Laboratory, Argonne, IL, USA, 1995.
DOI: 10.2172/94649.
URL: <http://dx.doi.org/10.2172/94649>.
- [50] V. P. BOBKOV, L. R. FOKIN, E. E. PETROV, V. V. POPOV, V. N. RUMIANTSEV AND A. I. SAVVATIMSKY. *Thermophysical Properties of Materials for Nuclear Engineering: A Tutorial and Collection of Data*. IAEA-THPH. IAEA, Vienna, Austria, 2008.
- [51] M. M. EL-WAKIL. *Nuclear Heat Transport*. American Nuclear Society, La Grange Park, IL, USA, 1978.
- [52] THE RELAP5-3D CODE DEVELOPMENT TEAM. *RELAP5-3D code manual, rev. 2.4*. Technical Report INEEL-EXT-98-00834, Idaho National Laboratory, Idaho Falls, ID, USA, 2005.
URL: <http://www.inl.gov/relap5/r5manuals.htm>.

- [53] C. W. STEWART, J. M. CUTA, S. D. MONTGOMERY, J. M. KELLY, K. L. BASEHORE, T. L. GEORGE AND D. S. ROWE. *VIPRE-01: A thermal-hydraulic code for reactor cores*. Technical Report EPRI-NP-2511-CCM-A, Revision 3, Pacific Northwest Laboratories, Richland, WA, USA, 1989.
- [54] C. L. WHEELER, C. W. STEWART, R. J. CENA, D. S. ROWE AND A. M. SUTEY. *COBRA-IV-I: An interim version of COBRA for thermal-hydraulic analysis of rod bundle nuclear fuel elements and cores*. Technical Report BNWL-1962, Pacific Northwest Laboratories, Richland, WA, USA, 1976.
DOI: 10.2172/7359476.
URL: <http://dx.doi.org/10.2172/7359476>.
- [55] C. W. STEWART, C. A. MCMONAGLE, M. J. THURGOOD, T. L. GEORGE, D. S. TRENT, J. M. CUTA AND G. D. SEYBOLD. *Core thermal model: COBRA-IV development and applications*. Technical Report BNWL-2212, Pacific Northwest Laboratories, Richland, WA, USA, 1977.
DOI: 10.2172/7100796.
URL: <http://dx.doi.org/10.2172/7100796>.
- [56] C. W. STEWART, C. L. WHEELER, R. J. CENA, C. A. MCMONAGLE, J. M. CUTA AND D. S. TRENT. *COBRA-IV: The model and the method*. Technical Report BNWL-2214, Pacific Northwest Laboratories, Richland, WA, USA, 1977.
DOI: 10.2172/5358588.
URL: <http://dx.doi.org/10.2172/5358588>.
- [57] Y. S. JUNG, C. B. SHIM, C. H. LIM AND H. G. JOO. *Practical numerical reactor employing direct whole core neutron transport and subchannel thermal/hydraulic solvers*. *Annals of Nuclear Energy*, **62**: pp. 357–374, 2013.
DOI: 10.1016/j.anucene.2013.06.031.
URL: <http://dx.doi.org/10.1016/j.anucene.2013.06.031>.
- [58] D. P. WEBER, T. SOFU, W. S. YANG, T. J. DOWNAR, J. W. THOMAS, Z. ZHONG, J. Y. CHO, K. S. KIM, T. H. CHUN, H. G. JOO AND C. H. KIM. *High-fidelity light water reactor analysis with the numerical nuclear reactor*. *Nuclear Science and Engineering*, **155**(3): pp. 395–408, 2007.
URL: http://www.ans.org/pubs/journals/nse/a_2672.
- [59] V. SEKER, J. W. THOMAS AND T. J. DOWNAR. *Reactor physics simulations with coupled Monte Carlo calculation and computational fluid dynamics*. In: *13th International Conference on Emerging Nuclear Energy Systems (ICENES)*. Istanbul, Turkey, June 3–8, 2007.
URL: http://icenes2007.org/icenes_proceedings/.
- [60] Z. XU, J. RHODES AND K. SMITH. *CASMO-5 versus MCNP-5 benchmark of radial power profile in a fuel pin*. In: *International Conference on Mathematics, Computational Methods & Reactor Physics (M&C)*. Saratoga Springs, NY, USA, May 3–7, 2009.
URL: <http://mathematicsandcomputation.cowhosting.net/MC09/math09/program.html>.
- [61] E. E. LEWIS. *Fundamentals of Nuclear Reactor Physics*. Academic Press, Burlington, MA, USA, 2008.

-
- [62] G. GRANDI, K. SMITH, Z. XU AND J. RHODES. *Effect of CASMO-5 cross-section data and Doppler temperature definitions on LWR reactivity initiated accidents*. In: *PHYSOR*. Pittsburgh, PA, USA, May 9–14, 2010.
- [63] T. KOZLOWSKI AND T. J. DOWNAR. *PWR MOX/UO₂ core transient benchmark - final report*. Technical Report NEA/NSC/DOC(2006)20, OECD/NEA, Paris, France, 2007.
URL: <http://www.oecd-nea.org/science/wprs/MOX-UOX-transients/>.
- [64] A. O. GOLTSEV, V. D. DAVIDENKO, V. F. TSIBULSKY AND A. A. LEKOMTSEV. *The influence of a non-uniform radial temperature distribution in the fuel on the results of calculation of transients*. *Annals of Nuclear Energy*, **30**(11): pp. 1135–1153, 2003.
DOI: 10.1016/S0306-4549(03)00042-2.
URL: [http://dx.doi.org/10.1016/S0306-4549\(03\)00042-2](http://dx.doi.org/10.1016/S0306-4549(03)00042-2).
- [65] A. E. WALTAR, D. R. TODD AND P. V. TSVETKOV. *Fast Spectrum Reactors*. Springer, New York, NY, USA, 2012.
DOI: 10.1007/978-1-4419-9572-8.
URL: <http://dx.doi.org/10.1007/978-1-4419-9572-8>.
- [66] J. RHODES, K. SMITH AND Z. XU. *CASMO-5 energy release per fission model*. In: *PHYSOR*. Interlaken, Switzerland, September 14–19, 2008.
- [67] J. K. WATSON AND K. N. IVANOV. *Demonstration of implicit coupling of TRACE/PARCS using simplified one-dimensional problems*. *Nuclear Technology*, **180**(2): pp. 174–190, 2012.
URL: http://www.ans.org/pubs/journals/nt/a_14632.
- [68] P. J. TURINSKY, R. M. K. AL-CHALABI, P. ENGRAND, H. N. SARSOUR, F. X. FAURE AND W. GUO. *NESTLE*. *Nuclear Science and Engineering*, **120**(1): pp. 72–73, 1995.
URL: http://www.ans.org/pubs/journals/nse/a_24107.
- [69] J. JUDD, G. GRANDI AND J. REDWINE. *A best-estimate coupled code for reactor safety analyses using SIMULATE-3K and RELAP5-3D*. In: *Advances in Nuclear Fuel Management IV*. Hilton Head Island, SC, USA, April 12–15, 2009.
- [70] Studsvik Scandpower, Inc. *SIMULATE-3K models & methodology*. Report SSP-98/13 Rev. 7. Idaho Falls, ID, USA, 2011.
- [71] D. KROPACZEK, K. SMITH AND J. BORKOWSKI. *A fully-implicit, five equation channel hydraulics model for SIMULATE-3K*. In: *Joint International Conference on Mathematical Methods and Supercomputing for Nuclear Applications*, volume 2, pp. 1401–1414. Saratoga Springs, NY, USA, October 6–10, 1997.
- [72] G. M. GRANDI. *Effect of the discretization and neutronic thermal hydraulic coupling on LWR transients*. In: *The 13th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH-13)*. Kanazawa City, Ishikawa Prefecture, Japan, September 27–October 2, 2009.
- [73] J. R. STEWART AND H. C. EDWARDS. *The SIERRA framework for developing advanced parallel mechanics applications*. In: L. T. BIEGLER, M. HEINKENSCHLOSS, O. GHATTAS AND B. VAN BLOEMEN WAANDERS (editors), *Large-Scale PDE-Constrained Optimization*,

- Lecture Notes in Computational Science and Engineering*, volume 30, pp. 301–315. Springer, Berlin, Heidelberg, Germany, 2003.
DOI: 10.1007/978-3-642-55508-4_18.
URL: http://dx.doi.org/10.1007/978-3-642-55508-4_18.
- [74] J. R. STEWART AND H. C. EDWARDS. *A framework approach for developing parallel adaptive multiphysics applications*. *Finite Elements in Analysis and Design*, **40**: pp. 1599–1617, 2004.
DOI: 10.1016/j.finel.2003.10.006.
URL: <http://dx.doi.org/10.1016/j.finel.2003.10.006>.
- [75] D. GASTON, C. NEWMAN, G. HANSEN AND D. LEBRUN-GRANDIÉ. *MOOSE: A parallel computational framework for coupled systems of nonlinear equations*. *Nuclear Engineering and Design*, **239**(10): pp. 1768–1778, 2009.
DOI: 10.1016/j.nucengdes.2009.05.021.
URL: <http://dx.doi.org/10.1016/j.nucengdes.2009.05.021>.
- [76] COMSOL, Inc. *COMSOL Multiphysics user’s guide, version 4.3*, Burlington, MA, USA, 2012.
- [77] D. CHANDLER, G. I. MALDONADO, R. T. PRIMM III AND J. D. FREELS. *Neutronics modeling of the High Flux Isotope Reactor using COMSOL*. *Annals of Nuclear Energy*, **38**(11): pp. 2594–2605, 2011.
DOI: 10.1016/j.anucene.2011.06.002.
URL: <http://dx.doi.org/10.1016/j.anucene.2011.06.002>.
- [78] J. MONTBRUN-DI FILIPPO, M. DELGADO, C. BRIE AND H. M. PAYNTER. *A survey of bond graphs: Theory, applications and programs*. *Journal of the Franklin Institute*, **328**(5-6): pp. 565–606, 1991.
DOI: 10.1016/0016-0032(91)90044-4.
URL: [http://dx.doi.org/10.1016/0016-0032\(91\)90044-4](http://dx.doi.org/10.1016/0016-0032(91)90044-4).
- [79] P. C. BREEDVELD, R. C. ROSENBERG AND T. ZHOU. *Bibliography of bond graph theory and application*. *Journal of the Franklin Institute*, **328**(5-6): pp. 1067–1109, 1991.
DOI: 10.1016/0016-0032(91)90069-F.
URL: [http://dx.doi.org/10.1016/0016-0032\(91\)90069-F](http://dx.doi.org/10.1016/0016-0032(91)90069-F).
- [80] G. G. NEFF. *Bond graph methods in reactor dynamics*. In: *American Nuclear Society meeting*. San Francisco, CA, USA, November 12, 1979.
URL: <http://www.osti.gov/scitech/biblio/5981991>.
- [81] J. L. TYLEE. *Bond graph modeling of nuclear reactor dynamics*. In: *Winter Annual Meeting, ASME Technology and Science Division*. Washington, DC, USA, November 16–21, 1981.
URL: <http://www.osti.gov/scitech/biblio/5463457>.
- [82] J. L. TYLEE. *Bond graph model of a pool-type nuclear reactor*. In: *American Control Conference*, volume 3, pp. 1939–1942. Seattle, WA, USA, June 18–20, 1986.
URL: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4789240>.

-
- [83] J. L. TYLEE. *Pseudo bond graph representation of PWR pressurizer dynamics*. Journal of Dynamic Systems, Measurement, and Control, **105**(4): pp. 255–261, 1983.
DOI: 10.1115/1.3140667.
URL: <http://dx.doi.org/10.1115/1.3140667>.
- [84] E. SOSNOVSKY. *Investigation of Bond Graphs for Nuclear Reactor Simulations*. Master's thesis, Massachusetts Institute of Technology, Department of Nuclear Science and Engineering, 2010.
DOI: 1721.1/62609.
URL: <http://hdl.handle.net/1721.1/62609>.
- [85] E. SOSNOVSKY, B. FORGET AND C. NEWMAN. *Bond graph based coupled reactor simulations*. In: *PHYSOR*. Pittsburgh, PA, USA, May 9–14, 2010.
- [86] F. T. BROWN. *Engineering System Dynamics: A Unified Graph-Centered Approach*. 2nd edition. CRC Press, Boca Raton, FL, USA, 2007.
- [87] D. C. KARNOPP, D. L. MARGOLIS AND R. C. ROSENBERG. *System Dynamics: Modeling and Simulation of Mechatronic Systems*. 4th edition. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2006.
- [88] B. J. JOSEPH AND H. R. MARTENS. *The method of relaxed causality in the bond graph analysis of nonlinear systems*. Journal of Dynamic Systems, Measurement, and Control, **96**(1): pp. 95–99, 1974.
DOI: 10.1115/1.3426781.
URL: <http://dx.doi.org/10.1115/1.3426781>.
- [89] D. C. KARNOPP. *Alternative bond graph causal patterns and equation formulations for dynamic systems*. Journal of Dynamic Systems, Measurement, and Control, **105**(2): pp. 58–63, 1983.
DOI: 10.1115/1.3149645.
URL: <http://dx.doi.org/10.1115/1.3149645>.
- [90] S. J. HOOD, E. R. PALMER AND P. M. DANTZIG. *A fast, complete method for automatically assigning causality to bond graphs*. Journal of the Franklin Institute, **326**(1): pp. 83–92, 1989.
DOI: 10.1016/0016-0032(89)90061-6.
URL: [http://dx.doi.org/10.1016/0016-0032\(89\)90061-6](http://dx.doi.org/10.1016/0016-0032(89)90061-6).
- [91] P. J. GAWTHROP AND L. SMITH. *Causal augmentation of bond graphs with algebraic loops*. Journal of the Franklin Institute, **329**(2): pp. 291–303, 1992.
DOI: 10.1016/0016-0032(92)90035-F.
URL: [http://dx.doi.org/10.1016/0016-0032\(92\)90035-F](http://dx.doi.org/10.1016/0016-0032(92)90035-F).
- [92] D. C. KARNOPP. *A bond graph modeling philosophy for thermo-fluid systems*. Journal of Dynamic Systems, Measurement, and Control, **100**(1): pp. 70–75, 1978.
DOI: 10.1115/1.3426342.
URL: <http://dx.doi.org/10.1115/1.3426342>.

- [93] D. C. KARNOPP. *Pseudo bond graphs for thermal energy transport*. Journal of Dynamic Systems, Measurement, and Control, **100**(3): pp. 165–169, 1978.
DOI: 10.1115/1.3426363.
URL: <http://dx.doi.org/10.1115/1.3426363>.
- [94] D. C. KARNOPP. *State variables and pseudo bond graphs for compressible thermo-fluid systems*. Journal of Dynamic Systems, Measurement, and Control, **101**(3): pp. 201–204, 1979.
DOI: 10.1115/1.3426425.
URL: <http://dx.doi.org/10.1115/1.3426425>.
- [95] F. T. BROWN. *Convection bonds and bond graphs*. Journal of the Franklin Institute, **328**(5-6): pp. 871–886, 1991.
DOI: 10.1016/0016-0032(91)90059-C.
URL: [http://dx.doi.org/10.1016/0016-0032\(91\)90059-C](http://dx.doi.org/10.1016/0016-0032(91)90059-C).
- [96] G. STRANG. *Computational Science and Engineering*. Wellesley-Cambridge Press, Wellesley, MA, USA, 2007.
- [97] CONTROLLAB PRODUCTS. *20-sim, the dynamic modeling and simulation package for iconic diagram, bond graph, block diagram and equation models*, February, 2010.
URL: <http://www.20sim.com/product/20sim.html>.
- [98] CONTROLLAB PRODUCTS. *TUTSIM, the ancestor of 20-sim*, February, 2010.
URL: <http://www.20sim.com/tutsim.html>.
- [99] R. C. ROSENBERG. *ENPORT-6 user's manual*, Rosencode Associates, Inc., Lansing, MI, USA, 1985.
- [100] CADSIM ENGINEERING. *CAMP-G: The universal bond graph preprocessor for modeling and simulation of mechatronics systems*, May, 2014.
URL: <http://www.bondgraph.com/>.
- [101] F. T. BROWN. *A unified approach to the analysis of uniform one-dimensional distributed systems*. Journal of Basic Engineering, **89**(2): pp. 423–432, 1967.
DOI: 10.1115/1.3609623.
URL: <http://dx.doi.org/10.1115/1.3609623>.
- [102] R. C. ROSENBERG. *State-space formulation for bond graph models of multiport systems*. Journal of Dynamic Systems, Measurement, and Control, **93**(1): pp. 35–40, 1971.
DOI: 10.1115/1.3426458.
URL: <http://dx.doi.org/10.1115/1.3426458>.
- [103] R. C. ROSENBERG. *Modeling and simulation of large-scale, linear, multiport systems*. Automatica, **9**(1): pp. 87–95, 1973.
DOI: 10.1016/0005-1098(73)90015-0.
URL: [http://dx.doi.org/10.1016/0005-1098\(73\)90015-0](http://dx.doi.org/10.1016/0005-1098(73)90015-0).
- [104] B. MOULTRIE. *State Formulation of Large-Scale Linear Time-Invariant Bond Graph Models*. Ph.D. thesis, Michigan State University, Department of Mechanical Engineering, 1979.

-
- [105] S. H. BIRKETT AND P. H. ROE. *The mathematical foundations of bond graphs - I. Algebraic theory*. Journal of the Franklin Institute, **326**(3): pp. 329–350, 1989.
DOI: 10.1016/0016-0032(89)90015-X.
URL: [http://dx.doi.org/10.1016/0016-0032\(89\)90015-X](http://dx.doi.org/10.1016/0016-0032(89)90015-X).
- [106] S. H. BIRKETT AND P. H. ROE. *The mathematical foundations of bond graphs - II. Duality*. Journal of the Franklin Institute, **326**(5): pp. 691–708, 1989.
DOI: 10.1016/0016-0032(89)90027-6.
URL: [http://dx.doi.org/10.1016/0016-0032\(89\)90027-6](http://dx.doi.org/10.1016/0016-0032(89)90027-6).
- [107] S. H. BIRKETT AND P. H. ROE. *The mathematical foundations of bond graphs - III. Matroid theory*. Journal of the Franklin Institute, **327**(1): pp. 87–108, 1990.
DOI: 10.1016/0016-0032(90)90059-R.
URL: [http://dx.doi.org/10.1016/0016-0032\(90\)90059-R](http://dx.doi.org/10.1016/0016-0032(90)90059-R).
- [108] S. H. BIRKETT AND P. H. ROE. *The mathematical foundations of bond graphs - IV. Matrix representations and causality*. Journal of the Franklin Institute, **327**(1): pp. 109–128, 1990.
DOI: 10.1016/0016-0032(90)90060-V.
URL: [http://dx.doi.org/10.1016/0016-0032\(90\)90060-V](http://dx.doi.org/10.1016/0016-0032(90)90060-V).
- [109] J. D. LAMB, D. R. WOODALL AND G. M. ASHER. *Bond graphs I: Acausal equivalence*. Discrete Applied Mathematics, **72**(3): pp. 261–293, 1997.
DOI: 10.1016/S0166-218X(97)85249-3.
URL: [http://dx.doi.org/10.1016/S0166-218X\(97\)85249-3](http://dx.doi.org/10.1016/S0166-218X(97)85249-3).
- [110] J. D. LAMB, D. R. WOODALL AND G. M. ASHER. *Bond graphs II: Causality and singularity*. Discrete Applied Mathematics, **73**(2): pp. 143–173, 1997.
DOI: 10.1016/S0166-218X(96)00006-6.
URL: [http://dx.doi.org/10.1016/S0166-218X\(96\)00006-6](http://dx.doi.org/10.1016/S0166-218X(96)00006-6).
- [111] J. D. LAMB, G. M. ASHER AND D. R. WOODALL. *Bond graphs III: Bond graphs and electrical networks*. Discrete Applied Mathematics, **73**(3): pp. 211–250, 1997.
DOI: 10.1016/S0166-218X(96)00009-1.
URL: [http://dx.doi.org/10.1016/S0166-218X\(96\)00009-1](http://dx.doi.org/10.1016/S0166-218X(96)00009-1).
- [112] F. E. CELLIER AND E. KOFMAN. *Continuous System Simulation*. Springer, New York, NY, USA, 2006.
DOI: 10.1007/0-387-30260-3.
URL: <http://dx.doi.org/10.1007/0-387-30260-3>.
- [113] E. SOSNOVSKY. *BGSolver bond graph processing code*, 2013.
URL: <http://github.com/mit-crpg/BGSolver/releases/>.
- [114] E. SOSNOVSKY AND B. FORGET. *Bond graph representation of nuclear reactor point kinetics and nearly incompressible thermal hydraulics*. Annals of Nuclear Energy, **68**: pp. 15–29, 2014.
DOI: 10.1016/j.anucene.2013.12.013.
URL: <http://dx.doi.org/10.1016/j.anucene.2013.12.013>.

- [115] M. A. HEROUX, R. A. BARTLETT, V. E. HOWLE, R. J. HOEKSTRA, J. J. HU, T. G. KOLDA, R. B. LEHOUCQ, K. R. LONG, R. P. PAWLOWSKI, E. T. PHIPPS, A. G. SALINGER, H. K. THORNQUIST, R. S. TUMINARO, J. M. WILLENBRING, A. WILLIAMS AND K. S. STANLEY. *An overview of the Trilinos project*. ACM Transactions on Mathematical Software, **31**(3): pp. 397–423, 2005.
DOI: [10.1145/1089014.1089021](https://doi.org/10.1145/1089014.1089021).
URL: <http://dx.doi.org/10.1145/1089014.1089021>.
- [116] R. J. LEVEQUE. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007.
- [117] D. E. SALANE. *Adaptive routines for forming Jacobians numerically*. Technical Report SAND86-1319, Sandia National Laboratory, Albuquerque, NM, USA, 1986.
URL: <http://www.ntis.gov/search/product.aspx?ABBR=DE86016057>.
- [118] J. WEISMAN. *Heat transfer to water flowing parallel to tube bundles*. Nuclear Science and Engineering, **6**(1): pp. 78–79, 1959.
URL: http://www.ans.org/pubs/journals/nse/a_25631.
- [119] C. B. DAVIS. *Verification and validation of corrected versions of RELAP5 for ATR reactivity analyses*. In: *RELAP5 International Users' Group Meeting*. Idaho National Laboratory, Idaho Falls, ID, USA, November 18–20, 2008.
URL: <http://www.osti.gov/scitech/servlets/purl/940047>.
- [120] E. SOSNOVSKY AND B. FORGET. *Bond graphs for spatial kinetics analysis of nuclear reactors*. Annals of Nuclear Energy, **56**: pp. 208–226, 2013.
DOI: [10.1016/j.anucene.2013.01.012](https://doi.org/10.1016/j.anucene.2013.01.012).
URL: <http://dx.doi.org/10.1016/j.anucene.2013.01.012>.
- [121] K. S. SMITH. *An Analytic Nodal Method for Solving the Two-group, Multidimensional, Static and Transient Neutron Diffusion Equations*. Nuclear Engineer's/Master's thesis, Massachusetts Institute of Technology, Department of Nuclear Engineering, 1979.
DOI: [1721.1/15979](https://doi.org/10.1721.1/15979).
URL: <http://hdl.handle.net/1721.1/15979>.
- [122] D. ROZON. *Introduction to Nuclear Reactor Kinetics*. Polytechnic International Press, Montreal, Quebec, Canada, 1998.