# OBJECT-ORIENTED MULTI-PHYSICS APPLIED TO SPATIAL REACTOR DYNAMICS

by

**I.D. Clifford**

Mini-dissertation submitted for the degree of

MASTER OF ENGINEERING (NUCLEAR)

at the Potchefstroom Campus of the

North-West University

Supervisor: Dr. O. Ubbink

Co-supervisor: Prof. E. Mulder

November 2007

# ABSTRACT

Traditionally coupled field reactor analysis has been carried out using several loosely coupled solvers, each having been developed independently from the others. In the field of multi-physics, the current generation of object-oriented toolkits provides robust close coupling of multiple fields on a single framework. This research investigates the suitability of such frameworks, in particular the Open-source Field Operation and Manipulation (OpenFOAM) framework, for the solution of spatial reactor dynamics problems. For this a subset of the theory of the TIme-dependent Neutronics and TEmperatures (TINTE) code, a time-dependent two-group diffusion solver, was implemented in the OpenFOAM framework. This newly created code, called diffusionFOAM, was tested for a number of steady-state and transient cases. The solver was found to perform satisfactorily, despite a number of numerical issues. The object-oriented structure of the framework allowed for rapid and efficient development of the solver. Further investigations suggest that more advanced transport methods and higher order spatial discretization schemes can potentially be implemented using such a framework as well.

# ACKNOWLEDGEMENTS

I would like to give my sincerest gratitude and appreciation to my supervisor Dr. Onno Ubbink. This research would not have been possible without your continued support, enthusiasm and patience.

To Prof. Hrvoje Jasak I also extend my sincerest thanks. Your assistance with OpenFOAM was greatly appreciated.

Many thanks to my co-supervisor Dr. Eben Mulder, and to my colleagues Frederik, Gerhard, Piet and Zain for your feedback and contributions.

And finally to my family and friends, in particular Héloïse… your encouragement and understanding have been invaluable.

*Ivor Clifford, 2007*

# TABLE OF CONTENTS

Page

Page

Page

# LIST OF FIGURES

Page

# LIST OF TABLES

Page

# ABBREVIATIONS

| | |
|---|---|
| AMG | agglomerated algebraic multigrid |
| BC | boundary condition |
| BICCG | incomplete Cholesky preconditioned biconjugate gradient |
| CCM | computational continuum mechanics |
| CFD | computational fluid dynamics |
| FE | finite-element |
| FEA | finite-element analysis |
| FEM | finite-element method |
| FOAM | Field Operation and Manipulation |
| FV | finite-volume |
| FVM | finite-volume method |
| HDF | Hierarchical Data Format |
| HTGR | high temperature gas-cooled reactor |
| HTR | high temperature reactor |
| LAM | Local Area Multicomputer |
| MP | multi-physics |
| MPI | message passing interface |
| NASA | National Aeronautics and Space Administration |
| NCTAM | National Committee on Theoretical and Applied Mechanics |
| OECD | Organization for Economic Co-operation and Development |
| OOP | object-oriented programming |
| OpenFOAM | Opensource Field Operation and Manipulation |
| PARCS | Purdue Advanced Reactor Core Simulator |
| PBMR | Pebble Bed Modular Reactor |
| PDE | partial differential equation |
| PJA | prompt jump approximation |
| TINTE | TIme dependent Neutronics and TEmperatures |
| TMI | Three Mile Island |
| U.S. | United States |
| U.S.NRC | United States Nuclear Regulatory Commission |
| VVER | Voda-Vodyanoi Energetichesky Reaktor |

# NOMENCLATURE

## *Latin Characters*

$A$ - in-cell coefficient

$\mathbf{A}$ - outward facing area vector / coefficient matrix

$B$ - neutron buckling

$C$ - delayed neutron precursor concentration / between group cross-term

$d$ - length

$\mathbf{d}$ - length vector

$D$ - material diffusion constant

$f$ - arbitrary function name

$E_f$ - Energy per fission

$F$ - nuclear fission rate

$G$ - total number of energy groups

$I$ - generic parent isotope

$k$ - effective reactor multiplication constant

$K$ - total number of ordinates

$L$ - neutron leakage / total number of delayed neutron precursor groups

$N$ - generic isotope concentration

$\mathbf{r}$ - position vector

$P$ - neutron production rate

$Q$ - source term / power production

$R$ - reaction rate

Re - time-dependent term in the neutron diffusion equation

$s$ - control volume surface

$S$ - source term

$\mathbf{S}$ - source term vector

$t$ - time

$\mathbf{U}$ - fluid velocity vector

$v$ - mean neutron velocity

$V$ - volume

$w$ - quadrature weight

$X$ - generic daughter isotope

## *Greek Characters*

$\alpha$ - albedo / relaxation factor

$\beta$ - delayed neutron fraction per fission

$\chi$ - neutron spectrum

$\Delta$ - discrete time interval

$\phi$ - scalar neutron flux

$\mathbf{\Phi}$ - neutron flux vector

$\varphi$ - mass flux

$\gamma$ - yield per fission

$\lambda$ - decay constant / extrapolated length

$\Lambda$ - neutron generation time

$\mu$ - integrating factor

$\nu$ - neutron yield per fission

$\rho$ - fluid density or inserted reactivity

$\sigma$ - microscopic neutron cross-section

$\Sigma$ - macroscopic neutron cross-section

$\mathbf{\Omega}$ - ordinate unit direction vector

$\psi$ - directional neutron flux

$\zeta$ - higher order production term

## *Superscripts*

$q^{1}$ - new time value (in the case of neutron flux and delayed neutron concentration)

$q^{0}$ - old time value (in the case of neutron flux and delayed neutron concentration)

$q^{g' \to g}$ - from the g'$^{th}$ energy group to the g$^{th}$ energy group

$q^{k' \to k}$ - from the k'$^{th}$ ordinate to the k$^{th}$ ordinate

$q^{*}$ - calculated value / after the prompt jump

## *Subscripts*

$q_0$ - old time value

$q_1$ - new time value

$q_a$ - absorption

$q_{albedo}$ - with reference to the albedo boundary condition

$q_B$ - at the boundary

$q_d$ - delayed

$q_{extrap}$ - with reference to the extrapolated length boundary condition

$q_f$ - fission / at the face

$q_g$ - with reference to the $g^{th}$ energy group

$q_i$ - with reference to the $i^{th}$ isotope

$q_j$ - with reference to the $j^{th}$ control volume

$q_k$ - with reference to the $k^{th}$ angular ordinate

$q_l$ - with reference to the $l^{th}$ delayed neutron precursor group

$q_s$ - scattering

$q_t$ - total

$q_\phi$ - with reference to the scalar neutron flux

## *Embellishments*

$q'''$ - per unit volume

$\dot{q}$ - first time derivative

$\overline{q}$ - average value

$\widetilde{q}$ - local value

# 1. INTRODUCTION

Nuclear reactor analysis deals with the coupled solution of the many physical processes taking place in a nuclear reactor. The solution of these individual physical processes has traditionally been carried out using several loosely-coupled solvers, each having been developed independently from the others. In particular, the calculation of the spatial distribution of neutrons in space and time is traditionally separated completely from the heat transfer calculation. This separation was introduced in the past for a number of reasons; the solution of each class of problem is typically undertaken by specialists in each field, the complexity of the problems differ, and there are numerical differences between the classes of problems being modeled. This separation leads to problems when coupling the solvers. Often differences in data management and spatial discretization require complex interface codes to be developed for the mapping and passing of data. Often independent source code is written to perform the same tasks in each solver and there is a significant amount of duplication. This in turn makes the verification of the coupled codes a time consuming and often labour-intensive task.

This particular problem is also encountered in the field of general multi-physics, which deals with the coupled solution of multiple fields. In the past, the fields of reactor analysis and of general multi-physics analysis, e.g. computational fluid dynamics or structural analysis, were considered to be separate entities, and therefore each has developed independently from the other over the years. The developments in each field have shown very different trends, driven largely by external influences in industry. In particular, strict regulations in the nuclear industry require that newly developed codes undergo a detailed verification and validation process, often prolonging the development times considerably. Thus there has been a reluctance to develop new codes. More often than not an older code will be updated, with the disadvantage that the older programming methodologies and structures remain unchanged.

In contrast, general multi-physics analysis applied to other engineering fields has advanced rapidly over recent years, embracing newer programming methodologies such as object-

oriented programming. This has in turn led to the development of several multi-physics toolkits, allowing the solution many classes of engineering problems in a simultaneous fashion, and readily extendable to new classes of problems. One such example is the Open-source Field Operation and Manipulation (OpenFOAM) toolkit, a set of classes written in the C++ programming language, which solves general partial differential equations using the finite-volume approach. The finite-volume approach is the standard methodology used today in computational fluid dynamics (CFD) calculations for the solution of fluid flow problems. It may be considered an extension to the finite-difference approach, which conserves the properties of a variable over a control volume.

## 1.1 *Research Objectives*

The objective of this research is to show that modern object-oriented multi-physics toolkits can effectively be used for the solution of spatial reactor dynamics and other classes of reactor analysis problems. In achieving this objective, the following questions will be considered and answered.

1. Can the OpenFOAM toolkit be successfully used to solve the spatial- and time-dependent multi-group neutron diffusion equation?
2. Does the OpenFOAM toolkit provide advantages in terms of the development and maintenance of a reactor analysis code?
3. Can the OpenFOAM toolkit be extended to allow for more advanced transport approximations such as discrete-ordinates and spherical-harmonics?
4. Can high-order spatial discretization schemes such as the nodal methods be generalized such that they may be implemented using the OpenFOAM toolkit?

Questions 1 and 2 are the focus of this research and will be answered using a practical approach. The remaining questions are essentially speculative, and answers will be given based on experience gained over the duration of this research. A step-by-step approach is followed which allows the above questions to be answered. Each step represents a logical

progression towards an understanding of the requirements of reactor analysis codes as well as the capabilities and advantages provided by the OpenFOAM toolkit. An existing code, the TIme-dependent Neutronics and TEmperatures (TINTE) code, provides the reference theory for a basic spatial- and time-dependent solver. The implementation of a subset of the TINTE functionality using OpenFOAM is the primary means by which experience will be gained for the purposes of answering the above questions.

## 1.2 *Outline of Dissertation*

Chapter 2 provides a review of the available literature that pertains to this research. Included in this chapter are a discussion and background on general reactor analysis and its development over the years in section 2.1. A basic introduction to the TINTE code system is also provided. The concept of multi-physics analysis is discussed in section 2.2, and we explore the current-day field of CFD analysis as a form of multi-physics analysis. The objective of the discussion in section 2.3 is to provide a general comparison between the current solution methods employed in both multi-physics analysis and reactor analysis codes. The concept of object-oriented programming and the advantages it provides for code development are introduced in section 2.4, followed by an introduction to object-oriented toolkits, which have been developed specifically for multi-physics analysis. In particular, one example of such toolkits, the OpenFOAM toolkit, is discussed in section 2.5.

The OpenFOAM toolkit is studied in more detail in chapter 3. Here the structure and functionality of the toolkit is examined from a reactor analysis perspective, addressing the major features. Chapter 4 details the basic subset of theory of the TINTE code, rewritten in a form that is more suited for direct implementation in OpenFOAM. The implementation of this theory in OpenFOAM is then described in chapter 5. Along with this implementation description useful and convenient features are noted, as well as certain missing features that would have been of assistance had they been available. Chapter 6 contains a summary of results, obtained using the newly implemented solver, for a compiled set of simple analytical

cases and numerical benchmark calculations. A discussion of findings and conclusions follows this in chapter 7.

# 2. LITERATURE SURVEY

## 2.1 *Nuclear Reactor Dynamics Methods*

The principle equation of use in reactor analysis is the neutron transport equation (Stacey 2001), which is derived from the Boltzmann equation for the kinetic theory of gases. This equation can be used to determine the distribution of neutrons and photons in space as a function of time. The transport equation may be solved directly in only a very limited number of cases. For this reason, approximations and simplifications to the transport equation are applied to solve engineering problems.

The solution methods may be divided into two classes, namely stochastic (Monte Carlo) and deterministic methods. The deterministic methods may be further classified into integral and integro-differential transport methods. The integro-differential transport methods include the discrete-ordinates and spherical-harmonics methods.

The discrete-ordinates methods (S-N methods) are based on the concept of evaluating the transport equation in a number of discrete angular directions. Quadrature relationships are used to replace the scattering and fission source angular integrals with sums over the angular directions (ordinates) (Stacey 2001). The result is a coupled set of equations for each ordinate and energy group, which are solved simultaneously to obtain the directional group fluxes.

The spherical harmonics methods (P-L methods) are based on the concept of representing the angular flux and differential scattering cross-section by means of Legendre polynomials (Stacey 2001). The result is a coupled set of equations for the N-Legendre flux moments and each energy group, which are solved simultaneously to obtain group fluxes.

A well known simplification to the transport equation is the diffusion approximation. Diffusion methods make use of Fick's law of diffusion to approximate the neutron current at a point in the reactor using a diffusion coefficient.

The diffusion equation is mathematically equivalent to the first order discrete-ordinates (S-1) and spherical harmonics (P-1) approximations. The diffusion approximation, in its derivation, assumes that neutron scattering is isotropic, neutron absorption is less likely than scattering, and that there is a linear spatial variation in the neutron distribution. These assumptions are valid for moderating materials, but not for fuels, strong absorbers and other regions of strong flux gradient, or cavities. Somewhat better approximations may, however, be obtained for the situations above by means of adjusted nuclear parameters. As an example, effective homogenized cross-sections (Stacey 2001) may be used to approximate the flux in regions containing strong absorber materials and to model the influences of control rods. Similarly, direction-dependent diffusion coefficients may be used to model the neutron streaming effects in cavities.

Despite the assumptions made and the inaccuracies associated with the diffusion approximation, the multi-group diffusion equation is still in common use today for spatial- and time-dependent reactor analysis because of its relative simplicity and speed. As an example, the United States Nuclear Regulatory Commission (U.S.NRC) currently uses the Purdue Advanced Reactor Core Simulator (PARCS) code (Joo et. al. 1998), a diffusion equation based solver, to predict the time-dependent behaviour of reactors during operation and during postulated accident conditions.

### 2.1.1 A Brief Background on Computational Reactor Analysis

Smith gives a very thorough overview of the development of reactor core analysis methods over the past decades (Smith 2003). Early reactor designs made use of the so-called four- and six-factor formulae. For this, extensive use of data fits, geometrical approximations and analytical solutions was required. In the 1950s, methods were driven largely by the needs of the naval light-water reactors. A large emphasis was placed on creating simple mathematical models for the many analytical concepts necessary for reactor analysis. These simplified analytical models relied heavily on an understanding of the underlying physics of the problem.

With the advent of the electronic computer in the 1960s and 1970s, reactor design began to make extensive use of computational methods. Early reactor dynamics codes solved the one-dimensional few-group diffusion equations, taking into account the effects of delayed neutrons and the fission products $^{135}I$ and $^{135}Xe$. This was later extended to two-dimensional finite-difference codes.

During the 1980s more advanced methods such as the finite-element method (FEM), amongst others, began to gain popularity. A number of codes were written making use of these 'more exotic' spatial discretization methods. An example of this is the TINTE code, discussed in more detail in upcoming sections, which makes use of the leakage iteration method, an extension of the finite-difference method. It was during these years that the personal computer industry boomed. It was also during this time, however, that accidents such as Three-Mile Island (TMI) and Chernobyl took place. This caused the nuclear industry to lose much momentum, and also reactor analysis code development. More stringent safety requirements resulted in increased code development times and the nuclear industry was reluctant to develop new codes. Over the last decade (mid 1990s onwards) the nuclear industry has since regained some momentum and, with this, a number of more modern codes have been developed.

There is currently an emphasis on replacing the older simplified methods of solution with a first principles approach to solving the neutron transport equation (Ragusa 2006). At present, a number of three-dimensional implementations of the discrete ordinates methods are available. One good example of a modern deterministic neutron transport solver is the research code ATTILA (Lucas et. al. 2004). ATTILA solves a first-order form of the steady-state transport equation on a three-dimensional unstructured spatial mesh, using tetrahedral mesh elements. ATTILA is coded in FORTRAN 90.

Ivanov (Ivanov 2007) states that current trends in nuclear power generation and in the design of next-generation plants are resulting in a greater emphasis being placed on improving analyses through improved coupled methodologies. The concept of multi-physics multi-scale

reactor analysis code systems has recently been introduced, aiming towards flexible and efficient coupling of reactor analysis models.

### 2.1.2 The TINTE Code System

The TIme-dependent Neutronics and TEmperatures (TINTE) code system (Gerwin 1987) (Gerwin et. al. 1989) is a two-group diffusion code for the calculation of the time-dependent nuclear and thermal behavior of high temperature gas-cooled reactors (HTGRs), in two-dimensional axisymmetric geometry. The code was originally written for the prediction of the behavior of pebble-bed reactors for short-term dynamics (power excursions, etc.) but this was later extended to medium term dynamics (xenon oscillations, etc.). The code was specifically written with speed in mind. A number of approximations and simplifications have been introduced to the code that have allowed full spatial and time-dependent reactor analysis at real-time or faster speeds using modern personal computers.

Written in FORTRAN 77, the neutronic module has recently been reverse engineered at PBMR (Clifford 2007), therefore the underlying equations and solution algorithms are well understood. TINTE solves the two-group neutron diffusion equations, taking into account the effects of delayed neutrons, fission-product poisoning and temperature changes. The reactor is modeled using a structured, rectangular, two-dimensional axisymmetric mesh.

## 2.2 *Multi-physics Analysis*

Multi-physics deals with coupled-field analysis, allowing analysts to determine the combined effects of multiple fields (physical phenomena) on a design (Lethbridge 2004/2005). In the past, the effects of these various phenomena were treated separately, utilizing a single analysis for each phenomena. As an example, the deflection of an aircraft wing was determined in the past by first analyzing the fluid flow over an undeflected wing. The resulting forces were then used as inputs to a wing deflection calculation. The modern multi-physics approach would be to couple a finite-volume (FV) computational fluid dynamics (CFD) and a finite-element (FE) material stress calculation together as a single calculation. The wing deflection is used to

update the mesh for the CFD calculation and the CFD calculation yields surface pressures and shear forces for the material stress calculation.

Many traditional nuclear reactor analysis codes may in fact be regarded as multi-physics codes. However, of interest to us are recent developments that have taken place in this field. Over recent years, generic CFD and finite-element analysis (FEA) codes have evolved into very competent multi-physics platforms. Typical examples of these codes are CFD-FASTRAN (CFD-FASTRAN 2007), ANSYS Multi-physics (Ansys MP 2007) and CFD-ACE+ (CFD-ACE+ 2007), combining fluid mechanics, solid stress and deflection analysis, heat transfer and chemical reaction kinetics as coupled modules within the overall package. To a large extent, these packages consist of a collection of coupled modules. The coupling between various fields may be either direct (implicit) or iterative (explicit) (Waterman 2004), depending on the complexity of the equations being solved. Implicit coupling requires a single matrix solution for all fields, while explicit coupling sequentially solves the individual problems, passing explicit values across the field interfaces and iterating until all solutions converge. This explicit coupling is achieved by means of tailored third party interfaces. The modules themselves are built on existing and well established CFD codes, solid mechanics codes, etc., the former of which are briefly discussed below. For reasons given below, modern CFD codes can be regarded as multi-physics codes and, because of this, the historical development and current status of this class of codes are discussed in the upcoming sections.

### 2.2.1 Computational Fluid Dynamics

The field of computational fluid dynamics (CFD) deals with the solution of the set of partial-differential-equations governing fluid flow, using a combination of mathematical modeling and numerical methods. The basis of CFD is the three conservation laws of mass, momentum and energy, using a continuum approach (Fletcher 1990). It should be noted that CFD, as it is applied today, deals with many closely coupled physical phenomena such as fluid flow, multiple fluid phase interactions, heat transfer, chemical reaction kinetics and particle transport. A modern CFD code may therefore be regarded as a multi-physics code.

### 2.2.2 A Brief Background on Computational Fluid Dynamics

The U.S. National Committee on Theoretical and Applied Mechanics gives a basic historical overview of CFD code development up to the 1990s (U.S. NCTAM et. al. 1991). The first methods for solving fluid flow using computational methods were based on conformal transformations of the flow around a cylinder to flow around airfoil cross-sections. The extension of these methods to three-dimensions was limited, at the time, by the available computing power. In 1966 the so-called panel method, allowing the three-dimensional solution of the potential flow equations, was first presented. This method represents the surfaces of the model geometry as several panels. These methods were largely developed by the aircraft industries of the time: NASA, Boeing, Lockheed, etc.

Panel codes were followed by full potential codes in the mid to late 1970s. The potential equations have limited applicability and with the appearance of more advanced computers in the 1970s, the solution of the Euler equations of fluid flow was considered. The upwind finite-difference, finite-volume and finite-element methods were developed during that decade. A number of commercial codes were developed in response, featuring multi-grid and other fast direct or iterative solvers. Initially, only structured grids were considered, but over time this was extended to unstructured grids.

In the 1980s the CFD service industry was created and this expanded very quickly into the 1990s. The growth and development of CFD codes and methodologies over these decades followed that of computers. This growth was largely driven by target industries, the greatest developments being seen in the aerodynamics, numerical weather prediction, acoustics and fluid-structure interaction, propulsion systems, and nuclear reactor design fields. This diverse set of target industries has meant that CFD has been a topic of great interest for the past two decades. Fletcher states that ‘perhaps the most important reason for the growth of CFD is that for much mainstream flow simulation, CFD is significantly cheaper than wind-tunnel testing and will become even more so in the future’ (Fletcher 1990). A more recent description on the

current status of CFD, and computational mechanics in general, is given by Oden et al. (Oden et. al. 2002).

When modern CFD codes are compared with the codes of the 1980s, there are vast differences in functionality, capabilities, as well as ease-of-use. When compared with the current generation of reactor analysis codes, there are also significant differences as a result of historical influences. Some of these differences are discussed in the next section.

## 2.3 *Comparison Between Modern Computational Fluid Dynamics and Reactor Analysis Codes*

If we consider the status of development of reactor analysis versus CFD codes up to the late 1970s, common trends are shared by both. By the early 1980s common features of codes included the use of finite-difference discretization on structured meshes, a linear programming style in FORTRAN 77. Additionally, the coupling of phenomena was generally achieved by externally coupling existing solvers. If one considers the changes made in each field since then, an obvious contrast emerges.

Modern reactor analysis codes employ methods such as the nodal and finite-element methods. Only in a few cases are non-orthogonal unstructured meshes used. A code will generally consist of several loosely coupled modules. It is interesting to note that despite the availability of more advanced programming languages such as FORTRAN 90/95 and C++, which support structured and object-oriented programming features, many modern reactor analysis codes are still written in a linear fashion using FORTRAN 77. One contributing factor is that the licensing of new reactor analysis codes is a very time-consuming and drawn out process. Developers are therefore reluctant to create new codes from scratch.

While the underlying theory has not changed significantly, the methodologies used in CFD analysis have changed substantially over the last few decades. Current commercial CFD codes almost exclusively use the finite-volume method. Typical examples of such codes are Star-CD (Star-CD 2007), Fluent (Fluent 2007) and CFX (Ansys CFX 2007). These codes

provide robust multi-grid solvers for the three-dimensional heat and mass transport equations, using fully unstructured meshes. Non-orthogonality of mesh cells is compensated for. The solvers are often extensible to allow for the solution of different classes of problems such as chemical reaction kinetics. Easy-to-use graphical user interfaces are provided for pre-processing, post-processing and the management of calculations. Modern CFD codes are almost exclusively written in an object-oriented language such as FORTRAN 90 or C++.

Despite obvious differences in the physics being modelled, it is clear that the field of reactor analysis would potentially benefit by taking careful advantage of the advancements which have been made in CFD and other general multi-physics fields over the years.

## 2.4 *Object-Oriented Programming*

Rumbaugh et al. defines object-oriented programming (OOP) as programming in terms of a collection of discrete objects that incorporate both data and behavior (Rumbaugh et. al. 1991). Historically, a program was viewed as a logical procedure that takes input data, processes it, and produces output data. In this context, the programming challenge was seen as how to write the logic, not how to define the data. Object-oriented programming takes the view that what we really care about are the objects we want to manipulate rather than the logic required to manipulate them. This is not to say that the logic no longer has importance but rather that, in the object-oriented context, each object is responsible for its own logic.

Object-oriented programming was initially conceived in the 1960s in response to the increasing complexity of hardware and software systems at the time (Meyer 1988). An object-oriented approach to programming was conceptualized to improve the quality of large complex hardware and software systems.

FORTRAN 77 is the classical scientific programming language on which most reactor analysis code systems have been developed in the past. This programming standard has been rendered obsolete by the more advanced FORTRAN 90 (Brainerd et. al. 1996) and FORTRAN 95 standards, both of which include enhancements and extensions over

FORTRAN 77 for high-level scientific programming. These enhancements include the support for a number of object-oriented concepts. The primary reason for the popularity of the FORTRAN derivatives in scientific programming is the ease with which multidimensional arrays and matrices can be manipulated. The FORTRAN derivatives, however, do not have full support for all object-oriented features.

While scientists would argue that a language such as C++ is not suitable for scientific code development and dedicated programmers would argue that FORTRAN 90/95 is too restrictive, it is clear that an object-oriented approach, which is supported by any number of modern programming languages, provides significant advantages for both code development and maintenance.

## 2.5 *Multi-physics Toolkits*

Numerous multi-physics toolkits (scientific computation frameworks) currently exist, providing general users and scientists flexible platforms on which sets of equations may be formulated and solved. Often these frameworks rely quite heavily on object-oriented structures and techniques to provide flexibility (Kruger 2004). The C++ language is often used as the basis for these frameworks for this reason. In the domain of FORTRAN-based programming languages, the concept of modular toolkits is found. One such environment (Filippone et. al. 1999) provides for the distributed solutions to general problems. With such modular toolkits, however, the user is often restricted to a fixed set of features.

The Open-source Field Operation and Manipulation (OpenFOAM) C++ class library (Weller et. al. 1998) provides a framework on which reliable and efficient computational continuum mechanics (CCM) codes may be developed. Prior to being released into the public domain the framework was known simply as FOAM and therefore, in this text, the terms FOAM and OpenFOAM are used interchangeably. The framework has been developed such that the top-level syntax of the code resembles closely the conventional mathematical notation used to represent tensors and partial-differential equations (PDEs). As an example (Weller et. al.

1998), the fluid mechanics mass conservation equation may be written in the mathematical form as shown below.

$$\frac{\partial \rho}{\partial t} + \nabla \bullet (\boldsymbol{\varphi}) = 0$$

where $\boldsymbol{\varphi} = \rho \mathbf{U}$, $\mathbf{U}$ is the fluid velocity vector, $\rho$ the fluid density and $t$ the time.

The solution to this equation is programmed in FOAM as shown below.

```
fvMatrix<scalar> rhoEqn
(
    fvm::ddt(rho)
    + fvc::div(phi)
);
rhoEqn.solve();
```

In the above code, the variables `rho` and `phi` are FOAM objects, based on the object-oriented concepts introduced in section 2.4. Each contains full spatial- and time-dependent definitions for the variables they represent. This high-level representation of equations allows for easy error-checking and rapid implementation of solvers. Additional detail on the internal FOAM representation of these objects is provided in chapter 3.

The framework was initially developed for the solution of CFD problems using the finite-volume method, but has been successfully used for the solution of other classes of problems such as solid material stress modeling and magneto-hydrodynamics. More recently, the framework has been applied to the typical multi-physics problems of fluid-solid interaction (Jasak 2006). The object-orientated structure of the framework is such that extensions (discretization schemes, boundary conditions, etc.) for new classes of problems may be introduced without any modification to the existing code. The framework is flexible enough that new functionality may be implemented at both the high level (tensors, PDEs) as well as at the low level (matrix solvers, acceleration methods, etc.), thus making it suitable for both research and production versions of a solver.

The FOAM framework provides many of the features normally found in today's commercial CFD packages, namely steady-state and time-dependent finite-volume solutions on arbitrary

unstructured meshes, with non-orthogonality correction, as well as multiple time and spatial discretization schemes. Further detail on the structure and functionality of the FOAM framework as it pertains to this research is given in chapter 3.

## 2.6 *Closure*

In this chapter introductions were given to the concepts of reactor analysis and multi-physics analysis. As part of this, the TINTE code was introduced as an example of a time-dependent multi-group diffusion solver. Further, a comparison was made between the development and current status of reactor analysis and general multi-physics codes. From this comparison it was shown that the field of reactor analysis could potentially benefit from current multi-physics methods. The concept of an object-oriented approach to software development was introduced. This then led on to an introduction to object-oriented multi-physics toolkits, in particular the OpenFOAM toolkit. This toolkit is discussed further in the chapter 3.

# 3. THE FOAM FRAMEWORK

The Field Operation and Manipulation (FOAM) framework, which was briefly described in section 2.5, will now be discussed in more detail. An emphasis is placed on the framework's functionality as it pertains to this research. In particular, an attempt has been made to provide examples relevant to neutronic calculations. For a more comprehensive description refer to the FOAM Programmer's Guide (OpenFOAM PG 2005).

## 3.1 *Tensors and Fields*

In FOAM mathematical equations are represented using tensors of varying rank. The most commonly found in nuclear and CFD calculations are tensors of rank 0 and 1, namely scalars and vectors. In FOAM a ranked tensor can be allocated dimensions; in this way, dimension checking is carried out for all operations.

The field class is the basic container class for scalars, vectors and higher rank tensors. Spatial discretization is handled in FOAM using the finite-volume method. A three-dimensional unstructured finite-volume mesh (`fvMesh`) is defined, consisting of any number of discrete cells. This mesh object, when associated with a field of tensors, is sufficient to describe the spatial distribution of the tensor over a given domain. Consider the scalar $\phi(\mathbf{r},t)$, which has both spatial- and time-dependence. This variable, when associated with a mesh, will have discrete scalar values $\phi_i(t)$ within each cell. Similarly if the time-domain is discretized into the current time $t_1$, old time $t_0$, and any number of older time points, the fully discretized representation for the scalar can be written $\phi_i^{\,1}$, $\phi_i^{\,0}$, $\phi_i^{\,-1}$, etc. FOAM therefore defines fields of tensors at each point in time. In FOAM terminology, the combination of a dimensioned tensor field at a discrete time point with a given mesh structure at the same time point is called a geometric field. Each geometric field is associated with its predecessors, i.e. the geometric field at previous time points. In FOAM a geometric field of scalars is termed a `volScalarField`, and a geometric field of vectors, a `volVectorField`.

FOAM includes the functionality to perform any number of operations on fields and geometric fields, including negation, addition, inversion, multiplication, trigonometric functions, cross-products, etc. depending on the rank of the tensor. This allows algebraic manipulation of the ranked tensor fields. The FOAM Programmer's Guide (OpenFOAM PG 2005) contains a more complete list of supported operations and functions. This functionality has been achieved in FOAM using C++ overloaded operators and functions. As an example, consider the typical example of the buildup of a fission product over time in a constant flux.

$$N_1 = N_0 e^{-\lambda\Delta} + \frac{F'''\gamma}{\lambda}\left(e^{-\lambda\Delta} - 1\right)$$

where

$N_1$ and $N_0$ are the isotope concentrations at the end and start of the time interval

$\gamma$ and $\lambda$ are the isotope fission yield and decay constant respectively

$F'''$ is the fission reaction rate

$\Delta$ is the time interval

In FOAM, the coding for this would resemble that given below.

```
N = N.oldTime()*EXP(-lambda*deltaT)
    + F*gamma/lambda*( EXP(-lambda*deltaT)-1);
```

Here the concentration `N`, at time 1, and `N.oldTime()`, at time 0, and the constant fission rate `F` are geometric fields of dimensioned scalars (`volScalarField`). In this way the clumsiness traditionally associated with array operations in C++ has been removed and replaced by a functionality similar to that of FORTRAN 90, where operations are carried out for entire blocks of data.

## 3.2 *Spatial Discretization*

The solution domain is discretized to form a computational mesh, consisting of many discrete control volumes or cells. Each variable is principally defined at the cell volumetric centres. FOAM makes use of an arbitrarily unstructured mesh, thus any number of cells of any shape are allowed. The only limitation on this is that control volumes may not overlap and they

must completely fill the solution domain. A typical mesh structure and computational cell is depicted in Figure 1.

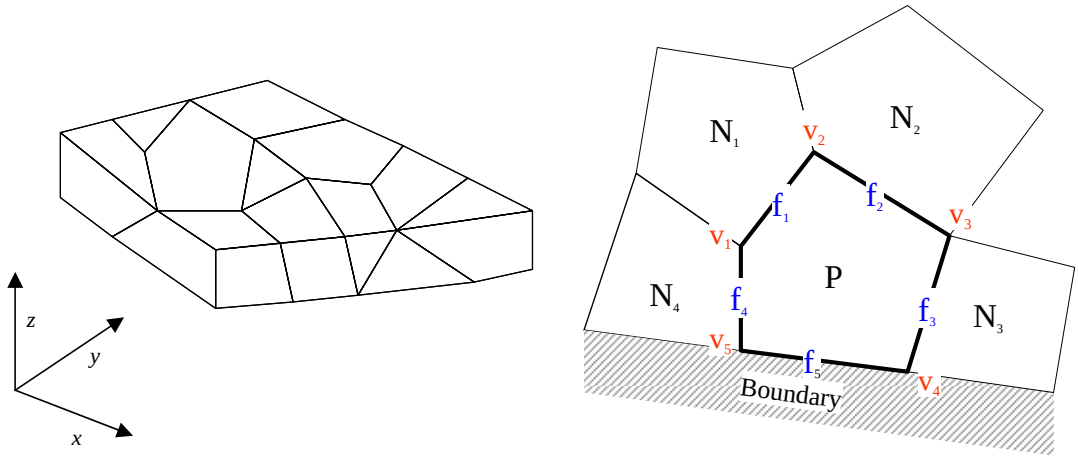



***Figure 1: A Typical FOAM Mesh and Computational Cell***

Each computational cell is defined by several faces forming the cell boundary. The faces may be shared between cells, or alternatively lie on the edge of the domain, forming a boundary. Each face, in turn, consists of a number of vertices. A face may be shared by two cells, in the case of an internal face, and a vertex may be shared by any number of faces. Each face is therefore constructed from any number of vertices on a flat plane. The full geometric definition of a mesh consists of a list of vertices, a list of faces based on vertex IDs, and a list of cells based on face IDs. For these, FOAM defines the `pointList`, `faceList` and `cellList` classes.

Additionally, the boundaries of the model must be defined. For this FOAM provides the `polyPatch` class, where each `polyPatch` object represents a cell face on the solution domain boundary. It is typical in CFD applications to define boundaries on a global scale, e.g. for the simulation of flow in a tube, one would define the tube walls, the tube inlet and the tube outlet as global boundaries. Typically, on the discretized mesh, each of these global boundaries consists of several boundary cell faces. This collection of `polyPatch` objects is contained in a `polyPatchList` object representing one global boundary. All global boundaries on a mesh are grouped together into a single `polyBoundaryMesh` object. The complete finite-volume

mesh definition, including the list of points, faces and cells, as well as the boundary definitions is contained in a `fvMesh` object.

## 3.3 *The Finite-Volume Method and Discretization*

Consider a simplified representation of the diffusion equation involving the scalar neutron flux $\phi$.

$$-\nabla \bullet (D\nabla\phi) + \Sigma\phi = S_\phi$$

For the purposes of this explanation, $D$, $\Sigma$ and $S_\phi$ are considered arbitrary constants. The finite-volume methodology may be applied to this conservation equation by integrating the equation over a discrete control volume $V$, in this case the computational cell. This control volume integration is the key step which distinguishes the finite-volume method from other numerical methods (Versteeg and Malalasekera 1995).

$$-\int_V \nabla \bullet (D\nabla\phi) dV + \int_V \Sigma\phi \, dV = \int_V S_\phi \, dV$$

For the diffusion term, one may apply Gauss' theorem to transform the volume integral to a surface integral. For other terms the properties are assumed constant over the control volume.

$$-\int_s (D\nabla\phi) \bullet d\mathbf{A} + \Sigma V\phi = S_\phi V$$

Here $\mathbf{A}$ is the control volume surface area vector and $s$ the surface of the control volume. The control volume is assumed to be bounded by any number of flat faces. The surface integral can be written as a sum over each of the faces.

$$-\sum_f \int_f (D\nabla\phi) \bullet d\mathbf{A} + \Sigma V\phi = S_\phi V$$

Here the subscript $f$ denotes the cell face. The midpoint approximation can be applied at each face, yielding the following.

$$-\sum_f (D\nabla\phi)_f \bullet \mathbf{A}_f + \Sigma V\phi = S_\phi V$$

where $\mathbf{A}_f$ is the outward pointing face area vector. Thus we note that applying the finite-volume method to a PDE results in an equation involving a sum over the cell faces. It is at this point that assumptions need to be made regarding properties at the faces. In the case above,

the neutron current $D\nabla\phi$ at the face would need to be determined. It is at this point that a spatial differencing scheme is chosen for $(D\nabla\phi)_f$. Such differencing schemes generally relate the value at the boundary to the cell centre value and neighbouring cell values. These schemes are discussed further in section 3.4.

For the case of time-dependent equations, the finite-volume approach requires a spatial as well as a time integration. Consider now a simplified form of the time-dependent diffusion equation.

$$\frac{1}{v}\frac{\partial\phi}{\partial t} - \nabla \bullet (D\nabla\phi) + \Sigma\phi = S_\phi$$

Again, for the purposes of this explanation, $v$, $D$, $\Sigma$ and $S_\phi$ are considered to be arbitrary constants. Using the approach shown previously, this may be written as shown.

$$\frac{1}{v}\frac{\partial\phi}{\partial t}V - \sum_f (D\nabla\phi)_f \bullet \mathbf{A}_f + \Sigma V\phi = S_\phi V$$

or more simply

$$\frac{\partial\phi}{\partial t} = f(t, \phi(t))$$

In the absence of analytical solutions to the terms contained within $f(t, \phi(t))$, these values are calculated at discrete points in time (Ferziger and Peric 2001). If an explicit Euler (forward-differencing) scheme is used, these values are evaluated at times for which the solution is already known. A fully implicit scheme (backwards-differencing) evaluates these values at times for which the solution is not already known. The Crank-Nicholson scheme is a combination of forwards and backwards differencing and assumes that these values are evaluated at some time in-between. The choice of differencing scheme affects the speed, stability and accuracy of the problem. Fully explicit schemes tend to be less stable while requiring little computational effort. Small time-intervals are necessary to achieve suitable stability and accuracy. Fully implicit schemes are unconditionally stable but require more computational effort. In general, the resulting discretized equation will have the form

$$\frac{\phi^1 - \phi^0}{t_1 - t_0} = f_1(\phi^1) + f_0(\phi^0)$$

where the superscripts/subscripts 0 and 1 denote the values at two consecutive time points.

After the PDE is fully discretized, a matrix equation is constructed. For an arbitrary PDE, this matrix equation generally takes on the form

$$\mathbf{A}\Phi = \mathbf{S}$$

Here $\mathbf{A}$ is a coefficient matrix, $\mathbf{S}$ is a source term vector and $\mathbf{\Phi}$ the vector of ranked tensors being solved for. An important feature of FOAM is the automatic construction of the coefficient matrix $\mathbf{A}$ and source term vector $\mathbf{S}$ for an arbitrary PDE. This is handled in FOAM using the classes of static functions contained in `finiteVolumeMethod`, abbreviated as `fvm`. Each `fvm` function or operation returns a `fvMatrix` object, which contains the coefficient matrix and source vector contributions, as well as a reference to the geometric field being solved for. The discretization method used to construct the coefficient matrix and source vector is dependent on user input. This is discussed further in section 3.4. Consider the simplified form of the time-dependent diffusion equation given below.

$$\frac{1}{v}\frac{d\phi}{dt} - \nabla \bullet (D\nabla\phi) + \Sigma\phi = S_\phi$$

Grouping the implicit terms (terms involving $\phi$) on the left of the equation, and explicit terms (independent of $\phi$) on the right of the equation, the above equation may be defined in FOAM as follows.

```
fvMatrix diffusionEqn
(
    1/v*fvm::ddt(phi) - fvm::laplacian(D, phi) + sigma*phi == S
);
```

Note that there is a distinction between the explicit and implicit forms of expressions. In the FOAM context, explicit refers to expressions that are calculated using already known variable values at the time they are requested. In general numerics, these are often referred to as source terms. Explicit terms contribute towards the source vector $\mathbf{S}$. In the FOAM context, implicit terms refers to expressions involving unknowns, and for which a solution is required. These

terms contribute towards the coefficient matrix $\mathbf{A}$. Implicit terms may be made explicit, if necessary, and placed in the source term. This would primarily be done to stabilize the matrix inversion process, yielding the same solution but requiring iteration for convergence.

The `fvm` namespace functions aim, wherever possible, to return a coefficient matrix with no explicit terms, i.e. they aim to be fully implicit. In most cases, however, explicit sources are unavoidable, resulting from non-linearity within problems. The use of higher order spatial differencing schemes, mesh non-orthogonality correction, solution under-relaxation and time differencing, amongst others, will all contribute towards the source vector.

As an example, the `fvm::ddt` operator, for the case of Euler time integration over a time interval $\Delta$, would evaluate to $\frac{\phi^1 - \phi^0}{\Delta}$. The coefficient matrix would in this case evaluate to $\frac{1}{\Delta}$ on the main diagonal with a contribution of $\frac{\phi^0}{\Delta}$ added to the explicit source term.

Consider also the case of the Laplacian (diffusion) term $\nabla \bullet (D\nabla\phi)$ which was linearised previously.

$$\sum_f (D\nabla\phi)_f \bullet \mathbf{A}_f$$

The face current $(D\nabla\phi)_f \bullet \mathbf{A}_f$ may be approximated by the cell-centre-to-cell-centre gradient $D_f \frac{\phi_P - \phi_{N_f}}{|\mathbf{d}|} \frac{\mathbf{d}}{|\mathbf{d}|} \bullet \mathbf{A}_f$, where $\mathbf{d}$ is the cell-centre-to-cell-centre vector. Thus, in the case where $\mathbf{d}$ is parallel to $\mathbf{A}_f$ (orthogonal mesh), the terms $\frac{D_f A_f}{d}$ and $-\frac{D_f A_f}{d}$ are added to the coefficient matrices for cells P and $N_f$ respectively. If $\mathbf{d}$ is not parallel to $\mathbf{A}_f$ (non-orthogonal mesh), an explicit source term contribution is necessary to compensate for the non-orthogonality (Peric 1985) (Jasak 1996) (Ferziger and Peric 2001).

A fully explicit equivalent to `fvm` is provided by the `finiteVolumeCalculus` class of static functions, abbreviated as `fvc`. All of the functionality of `fvm` is replicated in `fvc`. In this case

the expression is evaluated as-is using the current values in each variable. The `fvm` functions and operators provide the basis for the functionality shown in the example given in section 3.1. A typical neutronic example would be the calculation of cell neutron leakages using the diffusion approximation.

```
Leakage = fvc::laplacian(D, phi);
```

The Laplacian operator is just one of the many operators provided by the framework. The FOAM Programmer's Guide (OpenFOAM PG 2005) provides a list of the available `fvm` and `fvc` operators and functions.

## 3.4 *Numerical Differencing Schemes*

Finite-volume integration produces equations that require us to make approximations for the value and/or gradient of a ranked tensor at the cell faces. For this, one of numerous available spatial differencing schemes may be chosen. FOAM allows the user to choose from many differencing schemes for each PDE operator. Similarly, the user has a choice of a variety of time-differencing schemes, including Euler, backwards differencing and Crank Nicholson. As was the case for matrix solvers and boundary conditions, custom numerical schemes may be defined. Table 1 summarizes the classes from which custom differencing schemes may be derived.

***Table 1: FOAM Base Classes for Numerical Differencing Schemes***

| Operator | FOAM Base Class |
|---|---|
| Convection | convectionScheme |
| Divergence | divScheme |
| Laplacian | laplacianScheme |
| Gradient | gradScheme |
| Surface normal gradient | snGradScheme |
| d/dt | ddtScheme |
| $d^2/dt^2$ | d2dtScheme |

In the case of the spatial differencing, a surface interpolation scheme is necessary to determine the value at the face. FOAM provides several commonly used surface interpolation shemes, including linear, harmonic, upwind and quadratic upwind differencing, amongst others. These schemes are derived from the `surfaceInterpolationScheme` class. A custom surface interpolation scheme may thus be derived from this base class.

## 3.5 *Boundary Conditions*

Boundary conditions (BCs) for PDEs are divided into three groups:

- Dirichlet BC - prescribes a fixed value at the boundary
- Neumann BC - prescribes a fixed gradient at the boundary
- A combination of Dirichlet and Neumann boundary conditions

The FOAM framework makes provision for all of the above. A list of available boundary conditions is provided in the FOAM Programmer's Guide (OpenFOAM PG 2005). FOAM does not, however, provide the typical albedo and extrapolated length boundary conditions used in neutronic calculations. This issue is addressed in section 4.1.4.

A description of domain boundaries and their discretized representation has already been given in section 3.2. Some description is, however, necessary with regards to the treatment of boundaries by the operators and the functions of `finiteVolumeMethod` and `finiteVolumeCalculus`. When performing the discretization of equation terms, it is necessary to consider the contribution of the boundary faces to the overall face sum in the finite-volume discretized equation. Consider the discretization for the Laplacian operator given in section 3.3. For this operator, it is necessary to define the gradient $(\nabla\phi)_b$ at the boundary face. For other operators it may be necessary to define the value $\phi_b$ at the boundary face. Thus any boundary condition needs be able to specify both the face value and face gradient as a function of the cell value. For this, FOAM provides the `fvPatchField` class, which in turn provides the necessary functions to calculate boundary values and gradients for a given `polyPatchList`. Custom boundary conditions may be defined by deriving a new class

from the `fvPatchField` class. Typical examples of this in FOAM are `uniformFixedValueFvPatchField` and `zeroGradientFvPatchField`.

## 3.6 *Solvers*

The solution of the matrix equation $\mathbf{A\Phi = S}$ requires the computationally expensive inversion of the coefficient matrix $\mathbf{A}$. In general $\mathbf{A}$ is a sparse matrix, containing a large proportion of empty (zero) elements, and therefore the matrix inversion may be accelerated using any number of methods, including matrix preconditioning. FOAM provides the `lduMatrix::solver` class as the basis for inverting matrices, from which specific solvers are derived. Several matrix solvers are included in FOAM, for both symmetric and asymmetric matrices, including a Gauss Seidel, an agglomerated algebraic multigrid (AMG) solver tuned to elliptic problems, an incomplete Cholesky preconditioned biconjugate gradient (BICCG) solver, and several other sparse matrix solvers. For a more complete list of available matrix solvers, see the FOAM User's Guide (OpenFOAM UG 2005). Custom solvers may be defined by deriving a new class from `lduMatrix::solver`.

## 3.7 *Parallel Processing Support*

FOAM supports the domain decomposition method for parallel computing of large problems. In essence, this method separates the spatial domain into several smaller meshes. The solution is obtained for each mesh, while passing data at the separated faces between processors. Data is transferred using the Local Area Multicomputer (LAM) implementation of the standard message passing interface (MPI) (Burns et. al. 1994). The procedure of running a case in parallel requires three steps; decomposition of the mesh, parallel execution of the decomposed case, and reconstructing the solution mesh and data for postprocessing. An important feature of FOAM is that, by design, all newly developed applications automatically support parallel processing using the domain decomposition method.

## 3.8 *User Input*

Neutronic calculations are renowned for having large and complex input and output datasets. It is therefore important that input and output be handled in an organized and structured manner. This functionality is provided by the FOAM library classes. The inner workings of the FOAM library classes will not be discussed but, as an introduction to the structured layout of input and output data, a brief description of FOAM cases is provided here. For a more detailed case description, the FOAM User Guide (OpenFOAM UG 2005) may be consulted.

A typical FOAM case is given a name and stored in a directory of the same name. Within this, a number of subdirectories are required, specifically the `system`, `constant` and time directories. A graphical layout of this structure is given later in 5.2. The `system` directory contains information regarding the control and type of calculation to be performed. The `constant` directory contains the mesh and fixed physical properties for the system being solved. In a typical nuclear calculation this would include nuclear data such as decay constants, fission yields, etc.

Individual time directories are created at user-specified time intervals, containing individual files of data for particular fields and properties. These files are either supplied by the user or are written by FOAM during program execution.

The input and output format of FOAM is designed specifically to be flexible. Data is contained in individual files, and is organized into a number of dictionaries. These dictionaries have a free format similar to that of C++ code. Essentially each dictionary defines a hierarchical data structure, allowing any number of input or output objects to be specified using keywords. This approach may be compared to that of other data storage libraries such as the Hierarchical Data Format library (HDF5 2007), which uses a multi-object file format and allows a variety of different object types to be stored in a single file.

## 3.9 *Closure*

In this chapter, the OpenFOAM framework was discussed to a certain level of detail. Included in this discussion was an introduction to the finite-volume method as a general equation discretization method. An emphasis was placed on the framework's functionality as it pertains to this research. In particular, an attempt was made to provide examples relevant to neutronic calculations. In the upcoming chapter 4 a subset of the theory of the TINTE code is rederived and suitable solution algorithms are proposed for a time-dependent neutron diffusion code. This is done in such a way as to take advantage of the features of OpenFOAM discussed in this chapter.

# 4. THEORETICAL DESCRIPTION

This chapter includes rederivations of a subset of the theory of the TINTE code (Gerwin 1987). In particular, the theory has been rewritten in a form more suited for implementation in OpenFOAM. The derivation of a higher order discretization for the group diffusion equation, including delayed neutron treatment, is given in section 4.1. Section 4.2 outlines the modelling of saturation fission products such as $^{135}$Xe. Section 4.3 describes the very simple heat production model assumed. Section 4.4 describes the algorithms and solution methods to be used for the numerical solution of the equations of 4.1 through 4.4.

## 4.1 *The Few-Group Diffusion Equations*

The time-dependent group-diffusion equation for the g$^{th}$ energy group is given below (Stacey 2001).

$$\frac{1}{v_g}\frac{\partial \phi_g}{\partial t} = \nabla D_g \nabla \phi_g - \left(\Sigma_{ag} + \Sigma_{sg}\right)\phi_g + \sum_{g' \neq g} \Sigma_s^{g' \to g} \phi_{g'} + \chi_{p,g}(1-\beta)P''' + \sum_{l=1}^{6} \chi_{d,g,l} \lambda_l C_l + Q_g \quad , \qquad g = 1, \ldots, G \tag{4.1}$$

where

$\phi_g$ is the g$^{th}$ group flux

$v_g$ and $D_g$ are the g$^{th}$ group mean neutron velocity and diffusion constant respectively

$\Sigma_{ag}$ and $\Sigma_{sg}$ are the g$^{th}$ group macroscopic absorption and scattering-out cross-sections respectively

$\Sigma_s^{g' \to g}$ is the macroscopic scattering cross-section from group g' into group g

$\chi_{p,g}$ and $\chi_{d,g,l}$ are the prompt and delayed neutron spectra for the g$^{th}$ energy group and l$^{th}$ delayed neutron precursor group

$\beta$ is the delayed neutron fraction per fission

$\lambda_l$ and $C_l$ are the l$^{th}$ delayed neutron precursor group decay constant and precursor concentrations respectively

$Q_g$ is a fixed external source

The neutron production density term $P'''$ is defined as

$$P''' = \frac{1}{k}\nu F''' = \frac{1}{k}\nu \sum_{g'} \Sigma_{fg'}\phi_{g'} \tag{4.2}$$

where

$k$ is the effective reactor multiplication constant (k-effective), introduced to ensure criticality of the steady-state solution

$\nu$ is the total neutron yield per fission

$F'''$ is the fission rate density

$\Sigma_{fg'}$ is the g'$^{th}$ group macroscopic fission cross-section

### 4.1.1 Delayed Neutron Treatment

A small fraction of neutrons produced during fission are emitted with some delay after fission has taken place. These neutrons are known as delayed neutrons and they are formed primarily through the decay of fission products. Approximately 40 of the 500 total fission product nuclides emit delayed neutrons (Ott and Neuhold 1985). The accurate modeling of all these delayed neutron emitting nuclides is a complex task and, for this reason, a commonly used approximation assumes that the time-dependent integral behaviour of the delayed neutrons is well represented by six delayed neutron precursor groups, as is shown in Equation (4.1). Each delayed neutron precursor group is characterized by a precursor concentration $C_l$, a decay constant $\lambda_l$ and a group delayed neutron yield per fission $\nu_{d,l}$. The group delayed neutron fraction $\beta_l$ is defined as

$$\beta_l = \frac{\nu_{d,l}}{\nu} \tag{4.3}$$

where $\nu$ is the total net neutron yield per fission, defined previously

$$\nu = \nu_p + \nu_d \tag{4.4}$$

Here $\nu_p$ is the prompt neutron yield per fission and $\nu_d$ the total delayed neutron yield per fission, defined as

$$\nu_d = \sum_{l=1}^{6} \nu_{d,l} \tag{4.5}$$

The total delayed neutron fraction is defined as the sum of the delayed neutron fraction for all precursor groups.

$$\beta = \sum_{l=1}^{6} \beta_l \tag{4.6}$$

#### 4.1.1.1 Calculating Delayed Neutron Parameters for Fuel Mixtures

The prompt and delayed neutron yields are dependent on the fissionable nuclide under consideration. For materials consisting of a mixture of fissionable nuclides, current approaches use a fission rate weighting to calculate the effective delayed neutron yields for the mixture. Based on simplified form of the CASMO-3 implementation (Edenius and Forssen 1989), the delayed neutron yield for a mixture of isotopes may be written as shown below.

$$\nu_{d,l} = \frac{\sum_i \nu_{d,l,i} F_i'''}{\sum_i F_i'''} \tag{4.7}$$

$$\nu = \frac{\sum_i \nu_i F_i'''}{\sum_i F_i'''} \tag{4.8}$$

Here, the subscript $i$ denotes each fissionable isotope and $F_i'''$ is the fission rate density for each isotope in the material. The delayed neutron fraction may then be calculated using Equation (4.3).

$$\beta_l = \frac{\sum_i \nu_{d,l,i} F_i'''}{\sum_i \nu_i F_i'''} \tag{4.9}$$

It should be noted that, as is the case in the TINTE code (Gerwin 1987), no attempt is made to correct for the group structure during the calculation of $\beta$, i.e. the physical $\beta$ is used without correction, regardless of group structure.

#### 4.1.1.2 Delayed Neutron Data

The delayed neutron data supplied in the ENDF/B nuclear data libraries (Chadwick et. al. 2006) is given for each fissionable isotope $i$. Thus values for $\lambda_{l,i}$ and $\nu_{d,l,i}$ are known. In order to simplify the calculation, a common set of six decay constants for all fissionable isotopes can be chosen, and the values for $\nu_{d,l,i}$ recalculated using least squares regression. These modified delayed neutron yields can be obtained from a number of sources. Those values used in the TINTE code are given in Table 2, Table 3 and Table 4 (Clifford 2007).

***Table 2: Common Set of Decay Constants for the 6 Delayed Neutron Precursor Groups***

| **Delayed Neutron Group** | **Group Decay Constant $\lambda_l$** |
|---|---|
| 1 | 3.87 |
| 2 | 1.4 |
| 3 | 0.311 |
| 4 | 0.116 |
| 5 | 0.03174 |
| 6 | 0.01272 |

***Table 3: Isotope-Dependent Fractional Fission Yield ($\beta$) of Delayed Neutrons***

| **Fractional Fission Yield ($\beta$) of Delayed Neutrons [%]** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **$^{235}$U** | **$^{232}$Th** | **$^{233}$U** | **$^{234}$U** | **$^{236}$U** | **$^{238}$U** | **$^{239}$Pu** | **$^{240}$Pu** | **$^{241}$Pu** | **$^{242}$Pu** |
| 0.6904 | 2.3981 | 0.2962 | 0.4342 | 1.1693 | 1.7510 | 0.2245 | 0.2850 | 0.5354 | 1.0524 |

***Table 4: Isotope- and Group-Dependent Delayed Neutron Fractions ($\beta_l / \beta$)***

| **Group** | **Fractional Fission Yield ($\beta_l / \beta$) for Delayed Neutron Precursor Group [%]** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **$^{235}$U** | **$^{232}$Th** | **$^{233}$U** | **$^{234}$U** | **$^{236}$U** | **$^{238}$U** | **$^{239}$Pu** | **$^{240}$Pu** | **$^{241}$Pu** | **$^{242}$Pu** |
| 1 | 2.6 | 2.8 | 0.6 | 2.6 | 2.6 | 4.0 | 1.2 | 2.2 | 0.3 | 1.0 |
| 2 | 12.8 | 18.0 | 11.9 | 12.8 | 12.8 | 30.5 | 14.1 | 15.3 | 24.7 | 23.7 |
| 3 | 40.7 | 45.6 | 26.0 | 40.7 | 40.7 | 37.7 | 31.0 | 32.8 | 32.1 | 39.1 |
| 4 | 18.8 | 16.0 | 27.0 | 18.8 | 18.8 | 13.0 | 26.3 | 24.1 | 22.1 | 18.9 |
| 5 | 21.3 | 14.1 | 25.8 | 21.3 | 21.3 | 13.6 | 18.6 | 18.1 | 15.2 | 12.9 |
| 6 | 3.8 | 3.5 | 8.7 | 3.8 | 3.8 | 1.2 | 8.8 | 7.6 | 5.6 | 4.5 |

#### 4.1.1.3 Delayed Neutron Precursor Concentrations

The time-dependent behavior of each delayed neutron precursor group may be represented by the differential equation shown below (Ott and Neuhold 1985).

$$\frac{dC_l}{dt} = \frac{1}{k}\nu_{d,l}F''' - \lambda_l C_l \tag{4.10}$$

Where $F''' = \sum_i F_i''' = \sum_g \Sigma_{fg}\phi_g$ is the material fission rate density (fission rate per unit volume). Here we include the eigenvalue $k$ to be consistent with the Equation (4.2).

##### 4.1.1.3.1 *Steady-State Case*

For steady-state operation the time-derivative in equation (4.10) is zero and the equation reduces to

$$\lambda_l C_l = \frac{1}{k}\nu_{d,l}F''' \tag{4.11}$$

Where $F'''$ is the steady-state fission rate density.

##### 4.1.1.3.2 *Time-Dependent Case*

The derivation that follows is a slightly modified form of that which is applied in the TINTE code (Gerwin 1987). A linear time-variation in fission rate and constant delayed neutron yield per fission are assumed for a time interval $\Delta = t_1 - t_0$. The fission rate density is written as

$$F'''(t) = F_0''' + (F_1''' - F_0''')\frac{t - t_0}{\Delta};\ t \in (t_{0,}, t_1)$$

Substitution of this into Equation (4.10) allows the time-dependent group concentration to be solved for.

$$\frac{dC_l}{dt} + \lambda_l C = \frac{1}{k}\nu_{d,l}F'''$$

The above equation is an ordinary differential equation of the first kind $\dot{C}_l + p(t)C_l = q(t)$.

The solution for $C_l(t)$ for time $t > t_0$ may be determined using integrating factors.

$$p(t) = \lambda_l$$
$$q(t) = \frac{1}{k}\nu_{d,l}F'''(t)$$

The integrating factor $\mu(t)$ is found as follows.

$$\mu = e^{\int_{t_0}^{t} p(t')dt'} = e^{\int_{t_0}^{t} \lambda_l dt'} = e^{\lambda_l (t-t_0)}$$

The solution for $C_l(t)$ becomes

$$C_l(t) = \frac{\int_{t_0}^{t} \mu\, q(t)dt}{\mu} = \frac{\int_{t_0}^{t} e^{\lambda_2 (t'-t_0)} \frac{1}{k}\nu_{d,l}F'''(t')dt' + C_l(t_0)}{e^{\lambda_l (t-t_0)}}$$

Assuming $\lambda_l$ constant over the time-interval, this may be rewritten to solve for $\lambda_l C_l(t)$.

$$\begin{aligned}
\lambda_l C_l(t) &= e^{-\lambda_l (t-t_0)}\left\{\lambda_l C_l(t_0) + \lambda_l \frac{1}{k}\nu_{d,l}\int_{t_0}^{t} dt'\, F'''(t')e^{\lambda_l (t'-t_0)}\right\} \\
&= e^{-\lambda_l (t-t_0)}\left\{\lambda_l C_l(t_0) + \lambda_l \frac{1}{k}\nu_{d,l}\int_{t_0}^{t} dt'\left[F_0''' + (F_1''' - F_0''')\frac{t'-t_0}{\Delta}\right]e^{\lambda_l (t'-t_0)}\right\} \\
&= e^{-\lambda_l (t-t_0)}\left\{\lambda_l C_l(t_0) + \lambda_l \frac{1}{k}\nu_{d,l}\left[F_0'''\int_{t_0}^{t} dt'\, e^{\lambda_l (t'-t_0)} + \frac{F_1''' - F_0'''}{\Delta}\int_{t_0}^{t} dt'\,(t'-t_0)e^{\lambda_l (t'-t_0)}\right]\right\} \\
&= e^{-\lambda_l (t-t_0)}\left\{\lambda_l C_l(t_0) + \lambda_l \frac{1}{k}\nu_{d,l}\left[F_0'''\frac{1}{\lambda_l}\left(e^{\lambda_l (t-t_0)} - 1\right) + \frac{F_1''' - F_0'''}{\Delta}\frac{1}{\lambda_l{}^2}\left(1 + e^{\lambda_l (t-t_0)}(\lambda_l (t-t_0) - 1)\right)\right]\right\} \\
&= e^{-\lambda_l (t-t_0)}\left\{\lambda_l C_l(t_0) + \frac{1}{k}\nu_{d,l}\left[F_0'''\left(e^{\lambda_l (t-t_0)} - 1\right) + \frac{F_1''' - F_0'''}{\lambda_l \Delta}\left(1 + e^{\lambda_l (t-t_0)}(\lambda_l (t-t_0) - 1)\right)\right]\right\}
\end{aligned}$$

$$\lambda_l C_l(t) = \lambda_l C_l(t_0)e^{-\lambda_l (t-t_0)} + \frac{1}{k}\nu_{d,l}\left[F_0'''\left(1 - e^{-\lambda_l (t-t_0)}\right) + \frac{F_1''' - F_0'''}{\lambda_l \Delta}\left(e^{-\lambda_l (t-t_0)} + \lambda_l (t-t_0) - 1\right)\right] \qquad (4.12)$$

At $t = t_1$

$$\lambda_l C_l(t_1) = \lambda_l C_l^1 = \lambda_l C_l^0 e^{-\lambda_l \Delta} + \frac{1}{k}\nu_{d,l}\left[F_0'''\left(1 - e^{-\lambda_l \Delta}\right) + (F_1''' - F_0''')\frac{e^{-\lambda_l \Delta} + \lambda_l \Delta - 1}{\lambda_l \Delta}\right]$$

The final expression for the precursor concentration at the end of the time interval becomes

$$\lambda_l C_l^1 = \lambda_l C_l^0 e^{-\lambda_l \Delta} + \frac{1}{k} \nu_{d,l} \left[ F_0''' \left( \frac{1 - e^{-\lambda_l \Delta}}{\lambda_l \Delta} - e^{-\lambda_l \Delta} \right) + F_1''' \left( 1 - \frac{1 - e^{-\lambda_l \Delta}}{\lambda_l \Delta} \right) \right] \tag{4.13}$$

For the discretization of the time-dependent diffusion equation (in upcoming section 4.1.2), an accurate expression is required for the interval mean precursor concentrations. The reason for this is discussed in the upcoming section. The interval mean neutron production is calculated by a time integration of equation (4.12) as follows.

$$\begin{aligned}
\lambda_l \overline{C_l} &= \frac{1}{\Delta} \int_{t_0}^{t_1} dt \left\{ \lambda_l C_l^0 \, e^{-\lambda_l (t - t_0)} + \frac{1}{k} \nu_{d,l} \left[ F_0''' \left( 1 - e^{-\lambda_l (t - t_0)} \right) + \frac{F_1''' - F_0'''}{\lambda_l \Delta} \left( e^{-\lambda_l (t - t_0)} + \lambda_l (t - t_0) - 1 \right) \right] \right\} \\
&= \frac{1}{\Delta} \int_{t_0}^{t_1} dt \left\{ \left[ \lambda_l C_l^0 - \frac{1}{k} \nu_{d,l} \left( F_0''' - \frac{F_1''' - F_0'''}{\lambda_l \Delta} \right) \right] e^{-\lambda_l (t - t_0)} + \nu_{d,l} \left[ F_0''' + \frac{F_1''' - F_0'''}{\lambda_l \Delta} \left( \lambda_l (t - t_0) - 1 \right) \right] \right\} \\
&= \frac{1}{\Delta} \left\{ \begin{array}{l} \left[ \lambda_l C_l^0 - \frac{1}{k} \nu_{d,l} \left( F_0''' - \frac{F_1''' - F_0'''}{\lambda_l \Delta} \right) \right] \frac{-1}{\lambda_l} \left( e^{-\lambda_l (t_1 - t_0)} - e^{-\lambda_l (t_0 - t_0)} \right) \\ \quad + \int_{t_0}^{t_1} dt \left\{ \frac{1}{k} \nu_{d,l} F_0''' + \frac{1}{k} \nu_{d,l} \frac{F_1''' - F_0'''}{\lambda_l \Delta} \left( \lambda_l (t - t_0) - 1 \right) \right\} \end{array} \right. \\
&= \frac{1}{\Delta} \left\{ \begin{array}{l} \left[ \lambda_l C_l^0 - \frac{1}{k} \nu_{d,l} \left( F_0''' - \frac{F_1''' - F_0'''}{\lambda_l \Delta} \right) \right] \frac{1 - e^{-\lambda_l \Delta}}{\lambda_l} \\ \quad + \frac{1}{k} \nu_{d,l} F_0''' (t_1 - t_0) + \frac{1}{k} \nu_{d,l} \frac{F_1''' - F_0'''}{\lambda_l \Delta} \left( \lambda_l \frac{1}{2} \left[ (t_1 - t_0)^2 - (t_0 - t_0)^2 \right] - (t_1 - t_0) \right) \end{array} \right. \\
&= \frac{1}{\Delta} \left\{ \left[ \lambda_l C_l^0 - \frac{1}{k} \nu_{d,l} \left( F_0''' - \frac{F_1''' - F_0'''}{\lambda_l \Delta} \right) \right] \frac{1 - e^{-\lambda_l \Delta}}{\lambda_l} + \frac{1}{k} \nu_{d,l} F_0''' \Delta + \nu_{d,l} \frac{F_1''' - F_0'''}{\lambda_l \Delta} \left( \lambda_l \frac{\Delta^2}{2} - \Delta \right) \right\} \\
&= \left[ \lambda_l C_l^0 - \frac{1}{k} \nu_{d,l} \left( F_0''' - \frac{F_1''' - F_0'''}{\lambda_l \Delta} \right) \right] \frac{1 - e^{-\lambda_l \Delta}}{\lambda_l \Delta} + \frac{1}{k} \nu_{d,l} \left[ F_0''' + \frac{F_1''' - F_0'''}{\lambda_l \Delta} \left( \frac{\lambda_l \Delta}{2} - 1 \right) \right] \\
&= \lambda_l C_l^0 \frac{1 - e^{-\lambda_l \Delta}}{\lambda_l \Delta} + \frac{1}{k} \nu_{d,l} F_0''' \left( 1 - \frac{1 - e^{-\lambda_l \Delta}}{\lambda_l \Delta} - \frac{1 - e^{-\lambda_l \Delta}}{\lambda_l \Delta} - \frac{1}{2} + \frac{1}{\lambda_l \Delta} \right) \\
&\quad + \frac{1}{k} \nu_{d,l} F_1''' \left( \frac{1 - e^{-\lambda_l \Delta}}{\lambda_l \Delta} + \frac{1}{2} - \frac{1}{\lambda_l \Delta} \right)
\end{aligned}$$

And finally

$$\lambda_l \overline{C}_l = \lambda_l C_l^0 \frac{1-e^{-\lambda_l \Delta}}{\lambda_l \Delta} + \frac{1}{k} \nu_{d,l} \left[ F_0''' \left( \frac{1}{2} + \frac{e^{-\lambda_l \Delta} - \frac{1-e^{-\lambda_l \Delta}}{\lambda_l \Delta}}{\lambda_l \Delta} \right) + F_1''' \left( \frac{1}{2} - \frac{1 - \frac{1-e^{-\lambda_l \Delta}}{\lambda_l \Delta}}{\lambda_l \Delta} \right) \right] \qquad (4.14)$$

### 4.1.2 Time Discretization of the Few-Group Diffusion Equations

The non-discretized form of the few-group diffusion equation is given by Equation (4.1). The production $P'''$ may be replaced by the fission rate $F'''$, using the previous definition $P''' = \frac{1}{k} \nu F'''$.

$$\frac{1}{v_g} \frac{\partial \phi_g}{\partial t} = \nabla D_g \nabla \phi_g - \left( \Sigma_{ag} + \Sigma_{sg} \right) \phi_g + \sum_{g' \neq g} \Sigma_s^{g' \to g} \phi_{g'} + \chi_{p,g} \frac{1}{k} \nu_p F''' + \sum_{l=1}^{6} \chi_{d,g,l} \lambda_l C_l + Q_g \ , \qquad g = 1, \ldots, G$$

In considering a choice of time-discretization for the above equation, one must consider the accuracy requirements for each physical process taking place. We note that the required time intervals may vary from fractions of a second to minutes. Similarly, we note that the six delayed neutron groups lie within this range of times, therefore it is important to treat the delayed neutron terms with some accuracy. The derivation that follows, similar to that used in TINTE (Gerwin 1987), is a manipulation of the diffusion equation into a time-discretized form, which pays particular attention to the delayed neutron treatment and assumes an average rate of change for the time interval. The final outcome of this derivation is presented in Equations (4.21) through (4.24).

We seek the solution to this equation for the case of $\frac{\partial \phi_g(t)}{\partial t} = \overline{\dot{\phi}_g}$, where $\overline{\dot{\phi}_g}$ is the average time derivative over the interval $\Delta = t_1 - t_0$, such that

$$\overline{\dot{\phi}_g} = \frac{1}{2} \left( \dot{\phi}_g(t_1) + \dot{\phi}_g(t_0) \right) \qquad (4.15)$$

We further assume that

$$\dot{\phi}_g(t_1) = \frac{\phi_g^1 - \phi_g^0}{\Delta}$$

We define the end-of-interval and start-of-interval functions

$$\begin{aligned}\mathrm{Re}_g^1 &= \frac{1}{\nu_g}\dot{\phi}_g(t_1) \\ &= \nabla\bullet\left(D_g\nabla\phi_g^1\right) - \left(\Sigma_{ag}+\Sigma_{sg}\right)\phi_g^1 + \sum_{g'\neq g}\Sigma_s^{g'\to g}\phi_{g'}^1 + \chi_{p,g}\frac{1}{k}\nu_p F_1''' + \sum_{l=1}^{6}\chi_{d,g,l}\lambda_l C_{l1} + Q_g\end{aligned} \tag{4.16}$$

and

$$\begin{aligned}\mathrm{Re}_g^0 &= \frac{1}{\nu_g}\dot{\phi}_g(t_0) \\ &= \nabla\bullet\left(D_g\nabla\phi_g^0\right) - \left(\Sigma_{ag}+\Sigma_{sg}\right)\phi_g^0 + \sum_{g'\neq g}\Sigma_s^{g'\to g}\phi_{g'}^0 + \chi_{p,g}\frac{1}{k}\nu_p F_0''' + \sum_{l=1}^{6}\chi_{d,g,l}\lambda_l C_{l0} + Q_g\end{aligned} \tag{4.17}$$

We may now write

$$\frac{1}{\nu_g}\overline{\dot{\phi}_g} = \frac{1}{2}\left(\frac{1}{\nu_g}\dot{\phi}_g(t_1) + \frac{1}{\nu_g}\dot{\phi}_g(t_0)\right) = \frac{1}{2}\left(\mathrm{Re}_g^1 + \mathrm{Re}_g^0\right)$$

Because the average delayed neutron precursor concentrations $\overline{C_l}$ are known (derived previously in section 4.1.1), we replace the term $\frac{1}{2}\left(\sum_{l=1}^{6}\chi_{d,g,l}\lambda_l C_{l1} + \sum_{l=1}^{6}\chi_{d,g,l}\lambda_l C_{l0}\right)$, contained in $\frac{1}{2}\left(\mathrm{Re}_g^1 + \mathrm{Re}_g^0\right)$, with $\lambda_l\overline{C_l}$ as given in Equation (4.14).

$$\begin{aligned}\frac{1}{\nu_g}\overline{\dot{\phi}_g} &= \frac{1}{2}\left(\mathrm{Re}_g^1 + \mathrm{Re}_g^0 - \sum_{l=1}^{6}\chi_{d,g,l}\lambda_l C_{l1} - \sum_{l=1}^{6}\chi_{d,g,l}\lambda_l C_{l0}\right) + \sum_{l=1}^{6}\chi_{d,g,l}\lambda_l\overline{C_l} \\ &= \frac{1}{2}\left(\mathrm{Re}_g^1 + \mathrm{Re}_g^0 - \sum_{l=1}^{6}\chi_{d,g,l}\lambda_l C_{l1} - \sum_{l=1}^{6}\chi_{d,g,l}\lambda_l C_{l0}\right) \\ &\quad + \sum_{l=1}^{6}\chi_{d,g,l}\left\{\lambda_l C_l^0\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta} + \frac{1}{k}\nu_{d,l}\left[F_0'''\left(\frac{1}{2} + \frac{e^{-\lambda_l\Delta} - \frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta}\right) + F_1'''\left(\frac{1}{2} - \frac{1-\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta}\right)\right]\right\}\end{aligned}$$

$$\frac{1}{\nu_g}\overline{\dot{\phi}_g} = \frac{1}{2}\left( \mathrm{Re}_g^1 + \mathrm{Re}_g^0 - \sum_{l=1}^{6} \chi_{d,g,l}\lambda_l C_{l1} - \sum_{l=1}^{6} \chi_{d,g,l}\lambda_l C_{l0} \right) + \sum_{l=1}^{6} \chi_{d,g,l}\left\{ \lambda_l C_{l0}\, \frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta} \right\}$$
$$+ \frac{1}{k} F_0''' \sum_{l=1}^{6} \chi_{d,g,l}\nu_{d,l} \left( 1 + 2\frac{e^{-\lambda_l\Delta} - \dfrac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta} \right) + \frac{1}{k} F_1''' \sum_{l=1}^{6} \chi_{d,g,l}\nu_{d,l} \left( 1 - 2\frac{1 - \dfrac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta} \right)$$
$$= \frac{1}{2}\left( \mathrm{Re}_g^1 + \mathrm{Re}_g^0 - \sum_{l=1}^{6} \chi_{d,g,l}\lambda_l C_{l1} - \sum_{l=1}^{6} \chi_{d,g,l}\lambda_l C_{l0}\left( 1 - 2\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta} \right) \right)$$
$$+ \frac{1}{k} F_0''' \sum_{l=1}^{6} \chi_{d,g,l}\nu_{d,l} \left( 1 + 2\frac{e^{-\lambda_l\Delta} - \dfrac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta} \right) + \frac{1}{k} F_1''' \sum_{l=1}^{6} \chi_{d,g,l}\nu_{d,l} \left( 1 - 2\frac{1 - \dfrac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta} \right)$$

Equation (4.15) can now be used to solve for $\frac{1}{\nu_g}\dot{\phi}_g(t_1)$.

$$\frac{1}{\nu_g}\dot{\phi}_g(t_1) + \frac{1}{\nu_g}\dot{\phi}_g(t_0) = \mathrm{Re}_g^1 + \mathrm{Re}_g^0 - \sum_{l=1}^{6} \chi_{d,g,l}\lambda_l C_{l1} - \sum_{l=1}^{6} \chi_{d,g,l}\lambda_l C_{l0}\left( 1 - 2\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta} \right)$$
$$+ \frac{1}{k} F_0''' \sum_{l=1}^{6} \chi_{d,g,l}\nu_{d,l} \left( 1 + 2\frac{e^{-\lambda_l\Delta} - \dfrac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta} \right)$$
$$+ \frac{1}{k} F_1''' \sum_{l=1}^{6} \chi_{d,g,l}\nu_{d,l} \left( 1 - 2\frac{1 - \dfrac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta} \right)$$

Given that $\frac{1}{\nu_g}\dot{\phi}_g(t_0) = \mathrm{Re}_g^0$, the equation may now be reduced.

$$\frac{1}{\nu_g}\dot{\phi}_g(t_1) = \mathrm{Re}_g^1 - \sum_{l=1}^{6} \chi_{d,g,l}\lambda_l C_{l1} - \sum_{l=1}^{6} \chi_{d,g,l}\lambda_l C_{l0}\left( 1 - 2\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta} \right)$$
$$+ \frac{1}{k} F_0''' \sum_{l=1}^{6} \chi_{d,g,l}\nu_{d,l} \left( 1 + 2\frac{e^{-\lambda_l\Delta} - \dfrac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta} \right) + \frac{1}{k} F_1''' \sum_{l=1}^{6} \chi_{d,g,l}\nu_{d,l} \left( 1 - 2\frac{1 - \dfrac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta} \right)$$

This may be expanded using the definition for $\mathrm{Re}_g^1$ in Equation (4.16).

$$\frac{1}{\nu_g}\dot{\phi}_g(t_1) = \nabla\bullet\left(D_g\nabla\phi_g^1\right) - \left(\Sigma_{ag}+\Sigma_{sg}\right)\phi_g^1 + \sum_{g'\neq g}\Sigma_s^{g'\to g}\phi_{g'}^1 + \chi_{p,g}\frac{1}{k}\nu_p F_1 + \sum_{l=1}^{6}\chi_{d,g,l}\lambda_l C_{l1} + Q_g$$
$$-\sum_{l=1}^{6}\chi_{d,g,l}\lambda_l C_{l1} - \sum_{l=1}^{6}\chi_{d,g,l}\lambda_l C_{l0}\left(1-2\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}\right)$$
$$+\frac{1}{k}F_0'''\sum_{l=1}^{6}\chi_{d,g,l}\nu_{d,l}\left(1+2\frac{e^{-\lambda_l\Delta}-\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta}\right)+\frac{1}{k}F_1'''\sum_{l=1}^{6}\chi_{d,g,l}\nu_{d,l}\left(1-2\frac{1-\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta}\right)$$
$$=\nabla\bullet\left(D_g\nabla\phi_g^1\right)-\left(\Sigma_{ag}+\Sigma_{sg}\right)\phi_g^1+\sum_{g'\neq g}\Sigma_s^{g'\to g}\phi_{g'}^1+Q_g-\sum_{l=1}^{6}\chi_{d,g,l}\lambda_l C_{l0}\left(1-2\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}\right)$$
$$+\chi_{p,g}\frac{1}{k}\nu_p F_1'''+\frac{1}{k}F_1'''\sum_{l=1}^{6}\chi_{d,g,l}\nu_{d,l}\left(1-2\frac{1-\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta}\right)$$
$$+\frac{1}{k}F_0'''\sum_{l=1}^{6}\chi_{d,g,l}\nu_{d,l}\left(1+2\frac{e^{-\lambda_l\Delta}-\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta}\right)$$

$$\frac{1}{\nu_g}\dot{\phi}_g(t_1) = \nabla\bullet\left(D_g\nabla\phi_g^1\right) - \left(\Sigma_{ag}+\Sigma_{sg}\right)\phi_g^1 + \sum_{g'\neq g}\Sigma_s^{g'\to g}\phi_{g'}^1 + Q_g$$
$$+\frac{1}{k}F_1'''\left\{\chi_{p,g}\nu_p+\sum_{l=1}^{6}\chi_{d,g,l}\nu_{d,l}\left(1-2\frac{1-\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta}\right)\right\}$$
$$+\frac{1}{k}F_0'''\sum_{l=1}^{6}\chi_{d,g,l}\nu_{d,l}\left(1+2\frac{e^{-\lambda_l\Delta}-\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta}\right)-\sum_{l=1}^{6}\chi_{d,g,l}\lambda_l C_{l0}\left(1-2\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}\right)$$

Three factors may defined from the given equation.

- The new prompt production term is given by

$$\nu\zeta_g^1 = \chi_{p,g}\nu_p + \sum_{l=1}^{6}\chi_{d,g,l}\nu_{d,l}\left(1-2\frac{1-\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta}\right)$$

  In terms of the delayed neutron fraction, this may be written as

$$\zeta_g^1 = \chi_{p,g}(1-\beta) + \sum_{l=1}^{6}\chi_{d,g,l}\beta_l\left(1-2\frac{1-\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta}\right) \tag{4.18}$$

- The old prompt production term is given by

$$\nu\zeta_g^0 = \sum_{l=1}^{6}\chi_{d,g,l}\nu_{d,l}\left(1+2\frac{e^{-\lambda_l\Delta}-\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta}\right)$$

  In terms of the delayed neutron fraction, this may be written as

$$\zeta_g^0 = \sum_{l=1}^{6}\chi_{d,g,l}\beta_l\left(1+2\frac{e^{-\lambda_l\Delta}-\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}}{\lambda_l\Delta}\right) \tag{4.19}$$

- The delayed neutron source term is given as

$$Q_{d,g} = \sum_{l=1}^{6}\chi_{d,g,l}\lambda_l C_{l0}\left(1-2\frac{1-e^{-\lambda_l\Delta}}{\lambda_l\Delta}\right) \tag{4.20}$$

The fully time-discretized group diffusion equation with delayed neutron feedback may now be written.

$$\frac{\phi_g^1-\phi_g^0}{v_g\Delta} = \nabla\bullet\left(D_g\nabla\phi_g^1\right)-\left(\Sigma_{ag}+\Sigma_{sg}\right)\phi_g^1+\sum_{g'\neq g}\Sigma_s^{g'\to g}\phi_{g'}^1+Q_g+\frac{1}{k}\left[\zeta_g^1\nu F_1'''+\zeta_g^0\nu F_0'''\right]-Q_{d,g}$$

This may be written in a simplified form

$$\frac{\phi_g^1 - \phi_g^0}{v_g \Delta} - \nabla \bullet \left(D_g \nabla \phi_g^1\right) + A_g \phi_g^1 = \sum_{g' \neq g} C_{g' \to g} \phi_{g'}^1 + S_g, \; g = 1, \ldots, G \tag{4.21}$$

where

$$A_g = \Sigma_{ag} + \Sigma_{sg} - \zeta_g^1 \frac{1}{k} \nu \Sigma_{fg} \tag{4.22}$$

$$C_{g' \to g}, \ldots = \Sigma_s^{g' \to g} + \zeta_g^1 \frac{1}{k} \nu \Sigma_{fg'}, \; g' \neq g \tag{4.23}$$

$$S_g = \zeta_g^0 \frac{1}{k} \nu F_0''' - Q_{d,g} + Q_g \tag{4.24}$$

Equations (4.21) through (4.24) represent the fully time-discretized set of multi-group diffusion equations and are suitable for direct implementation in FOAM.

#### 4.1.2.1 Steady-State Case

The steady-state forms of equations (4.18), (4.19) and (4.20) may be written by taking the limit as the time interval $\Delta$ tends to infinity, yielding the following.

$$\zeta_g^1 = \chi_{p,g} (1 - \beta) + \sum_{l=1}^{6} \chi_{d,g,l} \beta_l \tag{4.25}$$

$$\zeta_g^0 = \sum_{l=1}^{6} \chi_{d,g,l} \beta_l \tag{4.26}$$

$$Q_{d,g} = \sum_{l=1}^{6} \chi_{d,g,l} \lambda_l C_{l0} \tag{4.27}$$

### 4.1.3 The In-cell Spectrum Solution

In order to apply the predictor-corrector algorithm, which is discussed later in section 4.4.1.3, a coupled solution for the group fluxes is required in each mesh cell. The derivation that follows yields a matrix equation which may be used to solve for the coupled solution of the group fluxes in a control volume, assuming a fixed leakage through the control volume surface.

If we assume that the spatial dependence of the neutron flux $\nabla \bullet \left( D_g \nabla \phi_g \right)$ may be linearised by defining buckling terms, a buckling term $B_g$ may be used to replace the Laplacian operator such that

$$\nabla \bullet \left( D_g \nabla \phi_g \right) \approx \left( D_g B_g{}^2 \right) \phi_g \tag{4.28}$$

Equation (4.21) may therefore be rewritten as

$$\frac{1}{v_g \Delta} \phi_g^1 - \left( D_g B_g{}^2 \right) \phi_g^1 + A_g \phi_g^1 - \sum_{g' \neq g} C_{g' \to g} \phi_{g'}^1 = \frac{1}{v_g \Delta} \phi_g^0 + S_g \,, \; g = 1, \ldots, G \tag{4.29}$$

In matrix form, this becomes

$$\begin{bmatrix} \frac{1}{v_1 \Delta} - D_1 B_1{}^2 + A_1 & -C_{2\to1} & -C_{3\to1} & \cdots \\ -C_{1\to2} & \frac{1}{v_2 \Delta} - D_2 B_2{}^2 + A_2 & -C_{3\to2} & \cdots \\ -C_{1\to3} & -C_{2\to3} & \frac{1}{v_3 \Delta} - D_3 B_3{}^2 + A_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \phi_1^1 \\ \phi_2^1 \\ \phi_3^1 \\ \vdots \end{bmatrix} = \begin{bmatrix} \frac{1}{v_1 \Delta} \phi_1^0 + S_1 \\ \frac{1}{v_2 \Delta} \phi_2^0 + S_2 \\ \frac{1}{v_3 \Delta} \phi_3^0 + S_3 \\ \vdots \end{bmatrix}$$

The above matrix equation is similar in form to the coupled equations used for spectrum calculations. The solution to the set of equations is unstable in reflectors, and other regions with low fission rates (Gerwin 1987). In these regions, the absorption of neutrons is significantly greater than the production. In order to obtain a non-zero solution this must be transformed into a fixed source problem. As a first attempt to bypass this problem, the cell neutron leakage is included as an explicit source rather than through implicit linearised buckling values. This assumption may be applied for all cells within the solution domain, regardless of whether they contain fuel or not. Thus the in-cell solution, in matrix form, becomes

$$\begin{bmatrix} \frac{1}{v_1 \Delta} + A_1 & -C_{2\to1} & -C_{3\to1} & \cdots \\ -C_{1\to2} & \frac{1}{v_2 \Delta} + A_2 & -C_{3\to2} & \cdots \\ -C_{1\to3} & -C_{2\to3} & \frac{1}{v_3 \Delta} + A_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \phi_1^1 \\ \phi_2^1 \\ \phi_3^1 \\ \vdots \end{bmatrix} = \begin{bmatrix} \frac{1}{v_1 \Delta} \phi_1^0 - L_1 + S_1 \\ \frac{1}{v_2 \Delta} \phi_2^0 - L_2 + S_2 \\ \frac{1}{v_3 \Delta} \phi_3^0 - L_3 + S_3 \\ \vdots \end{bmatrix} \tag{4.30}$$

where $L_g = D_g B_g{}^2 \phi_g^1$, the leakage based on the guess value for $\phi_g^1$.

### 4.1.4 Eigenvalue Calculation

The effective reactor multiplication constant (k-effective) is generally defined as the ratio of neutron production to neutron losses. In the presence of delayed neutrons, the definition is somewhat changed. We therefore consider the expanded form of Equation (4.21) in order to calculate this value.

$$\frac{\phi_g^1 - \phi_g^0}{v_g \Delta} = \nabla \bullet \left(D_g \nabla \phi_g^1\right) - \left(\Sigma_{ag} + \Sigma_{sg}\right)\phi_g^1 + \sum_{g' \neq g} \Sigma_s^{g' \to g} \phi_{g'}^1 + Q_g + \frac{1}{k}\left[\zeta_g^1 \nu F_1''' + \zeta_g^0 \nu F_0'''\right] - Q_{d,g}$$

This is rearranged to solve for $k$.

$$\tilde{k} = \frac{\zeta_g^1 \nu F_1''' + \zeta_g^0 \nu F_0'''}{\frac{\phi_g^1 - \phi_g^0}{v_g \Delta} - \nabla \bullet \left(D_g \nabla \phi_g^1\right) + \left(\Sigma_{ag} + \Sigma_{sg}\right)\phi_g^1 - \sum_{g' \neq g} \Sigma_s^{g' \to g} \phi_{g'}^1 - Q_g + Q_{d,g}}$$

This gives a definition for the local k-effective $\tilde{k}$. The global k-effective is calculated using domain and energy group integrated forms of the terms in the expression above.

$$k = \frac{\int_V \left[\sum_{g=1}^{G} \left(\zeta_g^1 \nu F_1''' + \zeta_g^0 \nu F_0'''\right)\right] dV}{\int_V \left[\sum_{g=1}^{G} \frac{\phi_g^1 - \phi_g^0}{v_g \Delta}\right] dV + \int_V \left[\sum_{g=1}^{G} \left(\left(\Sigma_{ag} + \Sigma_{sg}\right)\phi_g^1 - \nabla \bullet \left(D_g \nabla \phi_g^1\right) - \sum_{g' \neq g} \Sigma_s^{g' \to g} \phi_{g'}^1\right)\right] dV + \int_V \left[\sum_{g=1}^{G} Q_{d,g}\right] dV - \int_V \left[\sum_{g=1}^{G} Q_g\right] dV}$$

We note that the scattering terms between energy groups cancel each other, i.e.

$$\sum_{g=1}^{G} \left(\Sigma_{sg} \phi_g^1 - \sum_{g' \neq g} \Sigma_s^{g' \to g} \phi_{g'}^1\right) = 0$$

The expression for $k$ therefore becomes

$$k = \frac{\int_V \left[\sum_{g=1}^{G} \left(\zeta_g^1 \nu F_1''' + \zeta_g^0 \nu F_0'''\right)\right] dV}{\int_V \left[\sum_{g=1}^{G} \frac{\phi_g^1 - \phi_g^0}{v_g \Delta}\right] dV + \int_V \left[\sum_{g=1}^{G} \Sigma_{ag} \phi_g^1 - \nabla \bullet \left(D_g \nabla \phi_g^1\right)\right] dV + \int_V \left[\sum_{g=1}^{G} Q_{d,g}\right] dV - \int_V \left[\sum_{g=1}^{G} Q_g\right] dV}$$

We are generally only concerned with calculating k-effective for the steady-state case. In this case the time-dependent term is zero. Also, the volume integral can be replaced by a discrete sum over the mesh elements.

$$k = \frac{P_p^{global}}{R_{loss}^{global} + P_d^{global}} \tag{4.31}$$

The global prompt neutron production rate is defined as

$$P_p^{global} = \sum_j \left[ \sum_{g=1}^{G} \left( \zeta_g^1 \, \nu F_1''' + \zeta_g^0 \, \nu F_0''' \right) \right] V_j \tag{4.32}$$

The global neutron loss rate is defined as

$$R_{loss}^{global} = \sum_j \left[ \sum_{g=1}^{G} \Sigma_{ag} \phi_g^1 - \nabla \bullet \left( D_g \nabla \phi_g^1 \right) \right] V_j \tag{4.33}$$

The global delayed and fixed neutron production rate is defined as

$$P_d^{global} = \sum_j \left[ \sum_{g=1}^{G} \left( Q_{d,g} + Q_g \right) \right] V_j \tag{4.34}$$

In the above equations, the subscript $j$ indicates the mesh cells.

### 4.1.5 Boundary Conditions

The extrapolated length and albedo boundary conditions (BCs), commonly used in neutron diffusion calculations, specify the neutron current at the boundary as a linear function of the neutron flux in the cell lying adjacent to the boundary. These cannot easily be defined using a combination of Dirichlet or Neumann BCs, which require fixed values or fixed gradients at the boundary. A mathematical description for each of these BCs is derived in the next sections, in a form which can be directly implemented in FOAM.

#### 4.1.5.1 The Extrapolated Length Boundary Condition

We consider a discrete unstructured mesh cell P, with an edge coinciding with the domain boundary. Beyond this boundary, a vacuum is assumed to exist. A widely used approximation to this vacuum boundary for diffusion calculations is the extrapolated length boundary condition (Stacey 2001). The extrapolated length boundary specifies that the neutron flux will

vanish at some point beyond the boundary. Thus the neutron flux is zero at a given distance $\lambda_{extrap}$ past the boundary, where $\lambda_{extrap} = 0.7104 \times 3D$. The boundary condition is depicted in Figure 2.

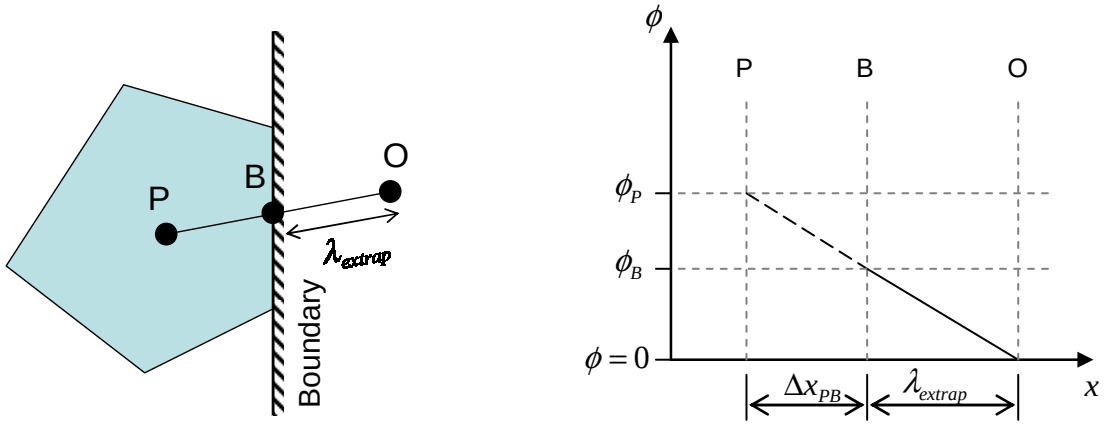



***Figure 2: The Extrapolated Length Boundary Condition***

The gradient at the boundary, $(\nabla\phi)_B$, is numerically approximated as

$$(\nabla\phi)_B \approx \frac{\phi_B - \phi_P}{\Delta x_{PB}}$$

This must correspond with the gradient from point B to point O.

$$\frac{\phi_B - \phi_P}{\Delta x_{PB}} = \frac{0 - \phi_B}{\lambda_{extrap}} = -\frac{\phi_B}{\lambda_{extrap}}$$

Rearranging this yields the neutron flux at the boundary.

$$\phi_B = \frac{\lambda_{extrap}}{\lambda_{extrap} + \Delta x_{PB}} \phi_P \tag{4.35}$$

The gradient at the boundary may now be written as

$$(\nabla\phi)_B = \frac{-1}{\lambda_{extrap} + \Delta x_{PB}} \phi_P \tag{4.36}$$

Equations (4.35) and (4.36) are sufficient to fully define the extrapolated length boundary condition in FOAM.

#### 4.1.5.2 The Albedo Boundary Condition

The albedo $\alpha$, the ratio of outgoing to incoming neutron current at the boundary, may be used to determine the neutron flux within a mesh according to the following relationship (Stacey 2001).

$$\left(\frac{1}{\phi}D\nabla\phi\right)_B = -\frac{1}{2}\left(\frac{1-\alpha}{1+\alpha}\right)$$

Thus the flux gradient at the boundary may be written directly as

$$(\nabla\phi)_B = -\frac{1}{2}\left(\frac{1-\alpha}{1+\alpha}\right)D_B\phi_B \approx \frac{\phi_B - \phi_P}{\Delta x_{PB}}$$

$$\therefore \phi_B\left(1+\frac{1}{2}\Delta x_{PB}\left(\frac{1-\alpha}{1+\alpha}\right)D_B\right) = \phi_P$$

Rearranging this yields the neutron flux at the boundary.

$$\phi_B = \frac{1}{1+\frac{1}{2}\Delta x_{PB}\left(\frac{1-\alpha}{1+\alpha}\right)D_B}\phi_P \tag{4.37}$$

The gradient at the boundary may now be written.

$$(\nabla\phi)_B = \frac{-\frac{1}{2}\left(\frac{1-\alpha}{1+\alpha}\right)D_B}{1+\frac{1}{2}\left(\frac{1-\alpha}{1+\alpha}\right)D_B\Delta x_{PB}}\phi_P$$

$$(\nabla\phi)_B = \frac{-1}{\frac{2}{D_B}\left(\frac{1+\alpha}{1-\alpha}\right)+\Delta x_{PB}}\phi_P \tag{4.38}$$

From the above equations, it is possible to relate the albedo boundary condition to an equivalent extrapolated length $\lambda_{albedo}$ (see Section 4.1.5.1) using the expression

$$\frac{1}{\lambda_{albedo}} = \frac{1}{2}\left(\frac{1-\alpha}{1+\alpha}\right)D_B \tag{4.39}$$

Equations (4.37) and (4.38) can therefore be rewritten as

$$\phi_B = \frac{\lambda_{albedo}}{\lambda_{albedo}+\Delta x_{PB}}\phi_P \tag{4.40}$$

$$(\nabla\phi)_B = \frac{-1}{\lambda_{albedo} + \Delta x_{PB}}\phi_P \tag{4.41}$$

Equations (4.40) and (4.41) relate directly to Equations (4.35) and (4.36).

## 4.2 *Iodine, Xenon and Other Neutron Poisons*

Certain fission products (Stacey 2001) will act as neutron absorbers and their formation tends to reduce the global reactor multiplication constant (k-effective). Some of these fission products are known as saturating fission products because their half-lives are sufficiently short that an equilibrium is reached between their production, decay and absorption during normal reactor operation. These isotopes will influence reactor operation in many cases such as reactor startup, shutdown and power level changes and therefore their influence must be taken into account. Of the saturating fission products, the isotopes $^{135}$Xe and $^{149}$Sm are generally considered the most important.

Xenon-135 has a large thermal absorption cross-section of approximately $2.6\times10^6\ b$ and is produced directly from fission and from the decay of $^{135}$I. $^{135}$I is produced from the decay of $^{135}$Te, which is a direct fission product. The half-life of $^{135}$Te (19 s) is sufficiently small, that a common approximation is to assume the $^{135}$I is formed directly from fission with yield $\gamma_I = \gamma_{Te}$ (Stacey 2001).

Samarium-149 has a large thermal absorption cross-section of approximately $4\times10^4\ b$ and is produced by the decay of $^{149}$Pm, which in turn is formed after the decay of $^{149}$Nd. The half-life of $^{149}$Nd is sufficiently small (1.7 h) that $^{149}$Pm can be assumed to be a direct fission product with yield $\gamma_{Pm} = \gamma_{Nd}$.

In the cases of both $^{135}$Xe and $^{149}$Sm, as well as the isotopes $^{151}$Sm and $^{157}$Gd, the decay chain may be represented as shown in Figure 3. Note that in this context, we refer to the production and decay of the generic isotopes $I \rightarrow X$, which can refer to any isotope pair that may be modeled according to Figure 3.

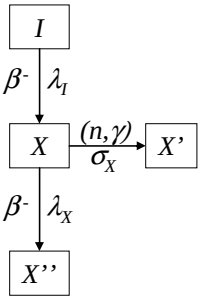



***Figure 3: Transmutation Decay chain for a Generic Neutron Poison***

The time-dependent concentration of the generic isotopes $X$ and $I$ in the above figure may be written in differential equation form.

$$\frac{d}{dt}I(t) = \gamma_I F'''(t) - \lambda_I I(t) \tag{4.42}$$

$$\frac{d}{dt}X(t) = \gamma_X F'''(t) + \lambda_I I(t) - \left(\lambda_X + \sum_g \sigma_{X,g}\phi_g(t)\right)X(t) \tag{4.43}$$

The TINTE code models what are considered to be the four important isotope pairs in short term HTGR dynamics, namely $^{135}I \rightarrow ^{135}Xe$, $^{149}Sm \rightarrow ^{149}Pm$, $^{151}Sm \rightarrow ^{151}Pm$ and $^{157}Eu \rightarrow ^{157}Gd$. Table 5 summarizes the decay constants for these isotope pairs as implemented in TINTE.

Note that no assumption has been made regarding the fission yields $\gamma_X$ and $\gamma_I$, or regarding the group-wise microscopic absorption cross-section of the daughter isotope $\sigma_{X,g}$. These values are assumed to be provided as calculation input.

*Table 5: Decay Constants of Important Neutron Poisons Decay Chains*

| Isotope Pair | Parent Isotope | | Daughter Isotope | |
|---|---|---|---|---|
| | Isotope (I) | Decay Constant $\lambda_I$ [s$^{-1}$] | Isotope (X) | Decay Constant $\lambda_X$ [s$^{-1}$] |
| 1 | I-135 | 2.88E-5 | Xe-135 | 2.12E-05 |
| 2 | Pm-149 | 3.63E-6 | Sm-149 | 1.00E-30* |
| 3 | Pm-151 | 6.88E-6 | Sm-151 | 5.75E-09 |
| 4 | Eu157 | 1.26E-5 | Gd-157 | 1.00E-30* |

### 4.2.1 Steady-State Case

For the steady-state case, the time-derivatives in equations (4.42) and (4.43) are zero. The steady-state concentration of the parent isotope *I* may be written.

$$I = \frac{\gamma_I}{\lambda_I} F''' \tag{4.44}$$

This may be substituted into equation (4.43) to yield the steady-state concentration of the daughter isotope $X$.

$$X = \frac{(\gamma_X + \gamma_I)}{\lambda_X + \sum_g \sigma_{X,g}\phi_g} F''' \tag{4.45}$$

### 4.2.2 Time-Dependent Case

We assume a constant fission rate for the time interval $\Delta = t_1 - t_0$.

$$F'''(t) = \overline{F}''' = \frac{1}{2}(F_1''' + F_0'''), \qquad t \in (t_{0,}, t_1)$$

* These isotopes are stable.

Substitution of this into Equation (4.42) allows the time-dependent concentration of the parent isotope $I$ to be solved for.

$$\frac{d}{dt}I(t) = \gamma_I \overline{F}''' - \lambda_I I(t)$$

$$I(t) = e^{-\lambda_I (t-t_0)} \left\{ I(t_0) + \int_{t_0}^{t} dt' \gamma_I \overline{F}''' e^{\lambda_I (t'-t_0)} \right\}$$

$$= e^{-\lambda_I (t-t_0)} \left\{ I_0 + \gamma_I \overline{F}''' \int_{t_0}^{t} dt' e^{\lambda_I (t'-t_0)} \right\}$$

$$= e^{-\lambda_I (t-t_0)} \left[ I_0 + \frac{\gamma_I}{\lambda_I} \overline{F}''' \left( e^{\lambda_I (t-t_0)} - 1 \right) \right]$$

$$I(t) = I_0 e^{-\lambda_I (t-t_0)} + \frac{\gamma_I}{\lambda_I} \overline{F}''' \left( 1 - e^{-\lambda_I (t-t_0)} \right)$$

At the end of the time interval ($t = t_1$)

$$I_1 = I_0 e^{-\lambda_I \Delta} + \frac{\gamma_I}{\lambda_I} \overline{F}''' \left( 1 - e^{-\lambda_I \Delta} \right) \tag{4.46}$$

A solution may now be found for the daughter isotope $X$, starting with equation (4.43). We define

$$\lambda_2 = \lambda_X + \sum_g \sigma_{X,g} \overline{\phi}_g \tag{4.47}$$

In the above definition, $\sigma_{X,g}$ and $\overline{\phi}_g$ are assumed constant over the time interval. Equation (4.43) may now be written as

$$\frac{d}{dt}X(t) + \lambda_2 X(t) = \gamma_X \overline{F}''' + \lambda_I I(t)$$

$$= \gamma_X \overline{F}''' + \lambda_I I_0 e^{-\lambda_I (t-t_0)} + \gamma_I \overline{F}''' \left( 1 - e^{-\lambda_I (t-t_0)} \right)$$

$$= (\gamma_X + \gamma_I) \overline{F}''' + \left( \lambda_I I_0 - \gamma_I \overline{F}''' \right) e^{-\lambda_I (t-t_0)}$$

This is an ordinary differential equation of the first kind $\dot{X} + p(t)X = q(t)$. The solution for $X(t)$ for time $t > t_0$ may be determined using integrating factors.

$$p(t) = \lambda_2$$

$$q(t) = (\gamma_X + \gamma_I) \overline{F}''' + \left( \lambda_I I_0 - \gamma_I \overline{F}''' \right) e^{-\lambda_I (t-t_0)}$$

The integrating factor $\mu(t)$ is found as follows.

$$\mu = e^{\int_{t_0}^{t} p(t')dt'} = e^{\int_{t_0}^{t} \lambda_2 dt'} = e^{\lambda_2 (t-t_0)}$$

The solution for $X(t)$ becomes

$$\begin{aligned} X(t) &= \frac{\int_{t_0}^{t} \mu\, q(t) dt}{\mu} = \frac{\int_{t_0}^{t} e^{\lambda_2 (t'-t_0)} \left[ (\gamma_X + \gamma_I)\overline{F}''' + (\lambda_I I_0 - \gamma_I \overline{F}''') e^{-\lambda_I (t-t_0)} \right] dt' + X(t_0)}{e^{\lambda_2 (t-t_0)}} \\ &= e^{-\lambda_2 (t-t_0)} \left\{ X(t_0) + \int_{t_0}^{t} e^{\lambda_2 (t-t_0)} \left[ (\gamma_X + \gamma_I)\overline{F}''' + (\lambda_I I_0 - \gamma_I \overline{F}''') e^{-\lambda_I (t'-t_0)} \right] dt' \right\} \\ &= e^{-\lambda_2 (t-t_0)} \left\{ X(t_0) + \int_{t_0}^{t} (\gamma_X + \gamma_I)\overline{F}''' e^{\lambda_2 (t'-t_0)} dt' + \int_{t_0}^{t} (\lambda_I I_0 - \gamma_I \overline{F}''') e^{(\lambda_2 - \lambda_I)(t'-t_0)} dt' \right\} \end{aligned}$$

$$\therefore X(t) = e^{-\lambda_2 (t-t_0)} \left\{ X(t_0) + (\gamma_X + \gamma_I)\overline{F}''' \int_{t_0}^{t} e^{\lambda_2 (t'-t_0)} dt' + (\lambda_I I_0 - \gamma_I \overline{F}''') \int_{t_0}^{t} e^{(\lambda_2 - \lambda_I)(t'-t_0)} dt' \right\}$$

The integrals may be evaluated and the equation simplified.

$$\begin{aligned} X(t) &= e^{-\lambda_2 (t-t_0)} \left\{ X(t_0) + (\gamma_X + \gamma_I)\overline{F}''' \frac{1}{\lambda_2} \left( e^{\lambda_2 (t-t_0)} - 1 \right) + (\lambda_I I_0 - \gamma_I \overline{F}''') \frac{1}{\lambda_2 - \lambda_I} \left( e^{(\lambda_2 - \lambda_I)(t-t_0)} - 1 \right) \right\} \\ &= X(t_0) e^{-\lambda_2 (t-t_0)} + (\gamma_X + \gamma_I)\overline{F}''' \frac{1 - e^{-\lambda_2 (t-t_0)}}{\lambda_2} + (\lambda_I I_0 - \gamma_I \overline{F}''') \frac{e^{-\lambda_I (t-t_0)} - e^{-\lambda_2 (t-t_0)}}{\lambda_2 - \lambda_I} \end{aligned}$$

At the end of the time interval ($t = t_1$) the daughter isotope concentration becomes

$$X_1 = X_0 e^{-\lambda_2 \Delta} + (\gamma_X + \gamma_I)\overline{F}''' \frac{1 - e^{-\lambda_2 \Delta}}{\lambda_2} + (\lambda_I I_0 - \gamma_I \overline{F}''') \frac{e^{-\lambda_I \Delta} - e^{-\lambda_2 \Delta}}{\lambda_2 - \lambda_I} \tag{4.48}$$

## 4.3 *Power Production*

The time-dependent power production, including decay heat production, were not considered for the FOAM implementation. As an approximation, all heat produced is assumed to be prompt and proportional to the fission rate.

$$Q''' = E_f F''' \tag{4.49}$$

where $Q'''$ is the power density and $E_f$ the energy per fission.

## 4.4 *Solution Algorithms*

In the preceding sections 4.1 through 4.3, a set of equations has been presented in a form suitable for implementing in FOAM. It is at this point that we now consider the solution strategy and algorithms that are required for the implementation. Section 4.4.1 considers the coupled solution of the neutron diffusion equation. Sections 4.4.2 and 4.4.3 introduce the complete algorithms for the steady-state and transient calculations respectively, and section 4.4.4 describes the inner iteration, i.e. the coupled calculation of neutron flux and neutron poison concentrations.

### 4.4.1 The Solution of the Time-Dependent Few Group Diffusion Equations

The implicit solution of the set of equations defined by equation (4.21) is not straightforward using the present FOAM framework. While the framework readily solves the $g^{th}$ group equation, the framework does not directly handle the coupling between the different energy group equations. The addition of this direct coupling to the framework is work in progress (Jasak 2007). This is discussed further in section 6.4.2. In the present absence of this feature an implicit solution for all energy groups requires iteration, explicitly updating the source contribution $\sum_{g' \neq g} B_{g' \to g} \phi_{g'}^{1}$ at each step.

The coupled solution of the few-group diffusion equations requires a suitable algorithm that will ensure stability up to time intervals in the order of 60 s, using the present framework's features. This stability cannot be easily achieved using an explicit coupling scheme. The equations for the fast energy groups are a factor of approximately a thousand stiffer than the thermal group equations. The coupling of these equations therefore poses a problem. This stiffness difference is due to the differences in mean neutron velocity for fast and thermal neutrons.

Similarly, the between-group coupling (neutron scattering) forms a relatively large proportion of the neutron source terms in each equation. Therefore one cannot assume that spectrum effects are of secondary importance to spatial effects. This presents a problem when

implementing the few-group diffusion equation in FOAM. The framework is specifically tailored towards problems where spatial effects dominate.

The solution of the one-group time-dependent diffusion equation may be carried out very efficiently using just one line of code.

```
solve(1/v*fvm::ddt(phi) – fvm::laplacian(D,phi) + A*phi = S);
```

For more than one energy group, however, because there is currently no implicit block solver in FOAM, a suitable algorithm for the implicit solution of the group fluxes is required. Some proposed options for this implicit solution are discussed in the upcoming sections.

#### 4.4.1.1 Explicit (Forward Difference) Group Flux Coupling

The simplest algorithm is an explicit coupling of the group fluxes. Here the out-of-group source terms are assumed to be dependent only on the start-of-interval fluxes $\phi_{g'}^{0}$, i.e.

$$\sum_{g' \neq g} B_{g' \to g} \phi_{g'}^{0}$$

This approximation requires no iteration but is only stable for small time intervals. The method also has limited accuracy, further requiring small time intervals.

#### 4.4.1.2 Implicit (Backward Difference) Group Flux Coupling

The numerical instability of the time-integration can be ensured by using the backward-difference algorithm. Here the out-of-group source terms are assumed to be dependent on the end-of-interval fluxes $\phi_{g'}^{1}$, i.e.

$$\sum_{g' \neq g} B_{g' \to g} \phi_{g'}^{1}$$

The problem, however, arises that the end of interval fluxes are not known and therefore an iterative scheme is necessary to obtain a coupled solution. A spatially dependent source term is assumed. For the first iteration, this is assumed to be based on the start-of-interval fluxes. Starting with the fastest flux group and working down to the slowest group, the group

diffusion equations are solved one-by-one to obtain an updated guess for the end-of-interval group fluxes. The updated guess fluxes are then used to obtain an updated guess of the source terms and the process is repeated until convergence is obtained.

This algorithm is represented below using pseudo-code.

```
Guess group source terms
while not converged
    for g=1,2,..., number of energy groups
          solve gth group diffusion equation
    end
    update group source terms

    check convergence
end
```

It is possible to improve the convergence of the implicit algorithm using a number of methods, including:

- If the source terms are updated directly following the spatial solution of each group's fluxes, the updated source terms propagate faster into the equation system and convergence can be improved in this way.
- Successive overrelaxation may be used to improve the rate of convergence. Here, a relaxation factor $\alpha$ is chosen $(0 < \alpha < 2)$. Each updated group flux is calculated as

$$\phi_g^1 = \alpha \phi_g^{1*} + (1-\alpha)\phi_g^0$$

  where $\phi_g^1$ is the updated group flux, and $\phi_g^{1*}$ the solution to the g$^{th}$ group diffusion equation. The choice of factor $\alpha$ greatly affects the rate of convergence. If $\alpha = 1$, this method reduces to the standard backwards differencing scheme.

If we consider the solution of the diffusion equation for a single energy group, assuming the between group terms to be fixed sources, it is clear that this single equation gives an implicit spatial solution, while the energy-dependence is treated explicitly. Thus, this algorithm is referred to as spatially-implicit.

The explicit treatment of the energy-dependence results in very poor numerical stability, largely due to the high stiffness of the fast energy group equations in relation to the thermal group equations. This numerical instability may only be improved using a more advanced energy coupling.

#### 4.4.1.3 Predictor-Corrector Algorithm

The stability problems associated with the spatially-implicit algorithm of section 4.4.1.2 may be improved by coupling this with in-cell spectrum calculations (refer to section 4.1.3) for each mesh cell to obtain a predictor-corrector type algorithm. The spectrum calculation implicitly couples the energy groups, and treats the spatial coupling explicitly (through buckling terms). It is thus referred to as an energy-implicit solution.

The predictor-corrector algorithm is represented below using pseudo-code.

```
Guess group source terms
while not converged
    for g=1,2,..., number of energy groups
          solve for gth energy group fluxes (spatially-implicit)
    end

    update buckling terms

    for i=1,2,..., number of mesh cells
          in-cell spectrum solution for ith mesh cell (energy-implicit)
    end

    update group source terms

    check convergence
end
```

This simple predictor-corrector algorithm is used as an initial attempt to obtain a stable multi-group flux solution. The implementation of a more advanced algorithm or block coupled solution is considered outside of the scope of this research.

### 4.4.2 Steady-State Eigenvalue Calculation

A pseudo-transient algorithm is used to calculate the eigenvalue and steady-state neutron fluxes, as in the case of the TINTE code. Initially the neutron flux profile is guessed. This flux profile is assumed to be user-supplied. An initial eigenvalue (k-effective) of unity is assumed.

An artificial time interval is then chosen and the updated neutron fluxes at the end of this time-interval are calculated. These updated neutron fluxes are then used to calculate an updated eigenvalue. At each step, the reactor power is normalized to a user-specified power level. With iteration, the global reactor power, neutron fluxes and eigenvalue will converge to the steady-state values. This algorithm is depicted in Figure 4.

#### 4.4.2.1 Reducing the Number of Iterations to Convergence

The TINTE code has an optimized controller which ‘steers’ the steady-state calculation, in order to reduce the number of iterations required for convergence. In order to simplify the FOAM implementation, only one optimizing measure is applied. The mean neutron velocities of all energy groups are assumed unity for the duration of the steady-state calculation. This eliminates the problem of stiffness differences between the diffusion equations for fast and thermal energy groups, allowing large artificial time intervals to be chosen.

### 4.4.3 Time-Dependent Calculation

A time-dependent calculation is an initial-value problem, and can only be carried out once the reactor eigenvalue is known, and the steady-state calculation therefore precedes this. The time-dependent algorithm is illustrated in Figure 5. The basic iterative strategy of the algorithm shows only small differences from the steady-state algorithm of section 4.4.2. These differences include the following.

- No normalization of the reactor power is performed.
- The eigenvalue (k-effective) is not calculated. The value is kept constant following the steady-state calculation.
- No outer iteration is required for each time-interval.

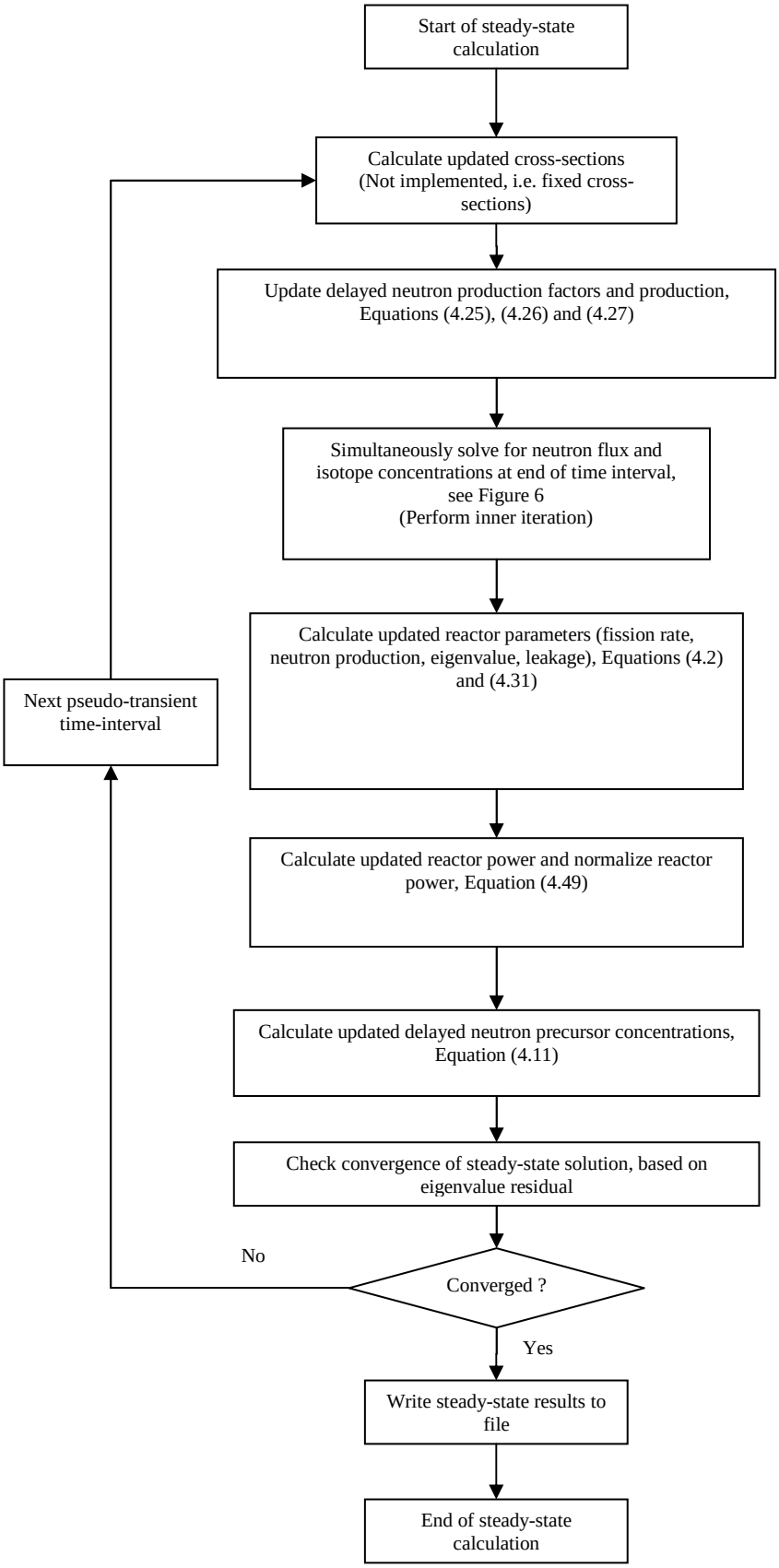



***Figure 4: Algorithm for the Steady-state Eigenvalue Calculation***

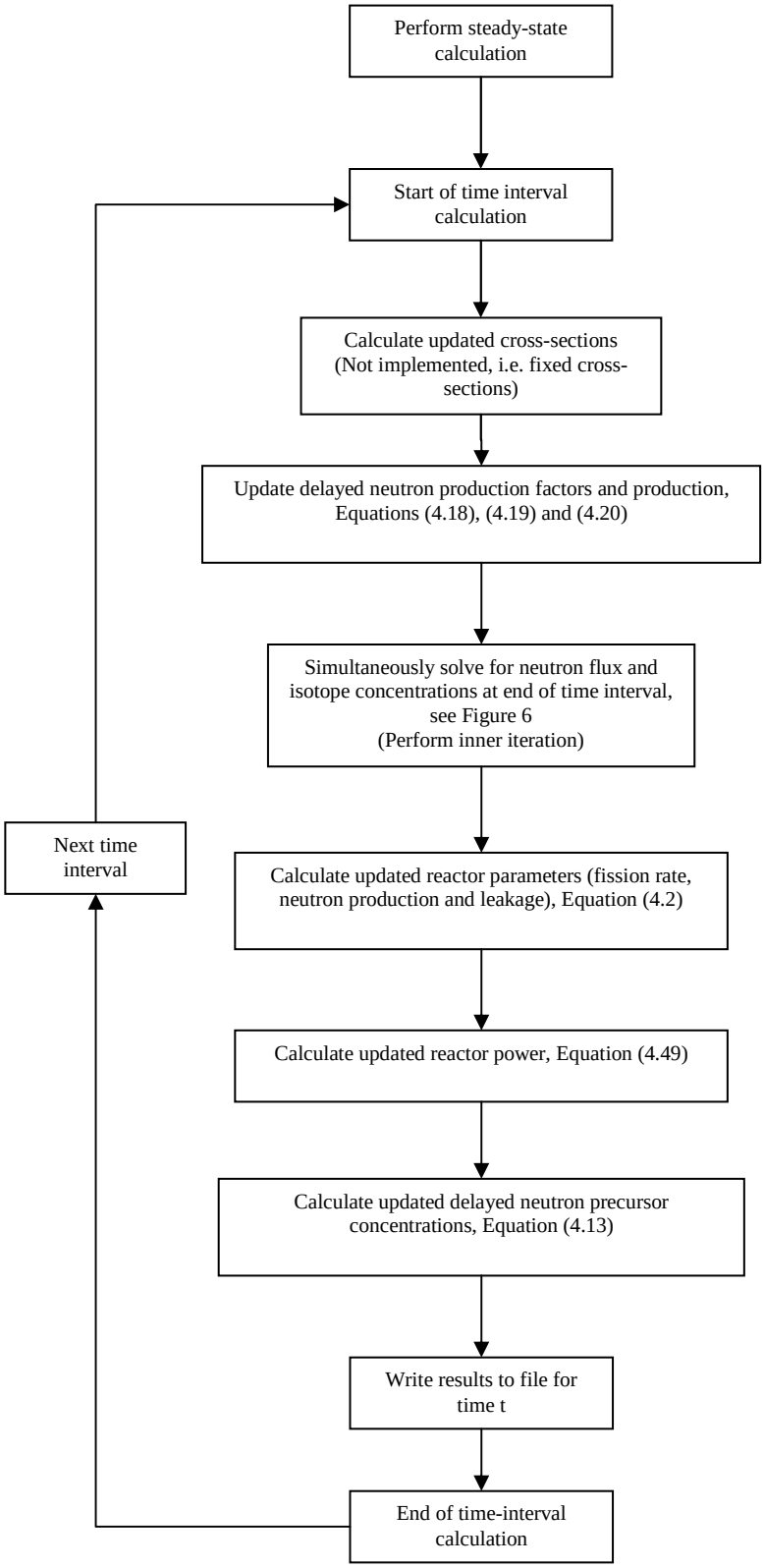



***Figure 5: Algorithm for Time-Dependent Calculation***

### 4.4.4 The Inner Iteration

Both steady-state and time-dependent algorithms require a simultaneous solution for the neutron flux and strong absorber isotope concentrations. For this, an inner iteration is used to obtain converged values.

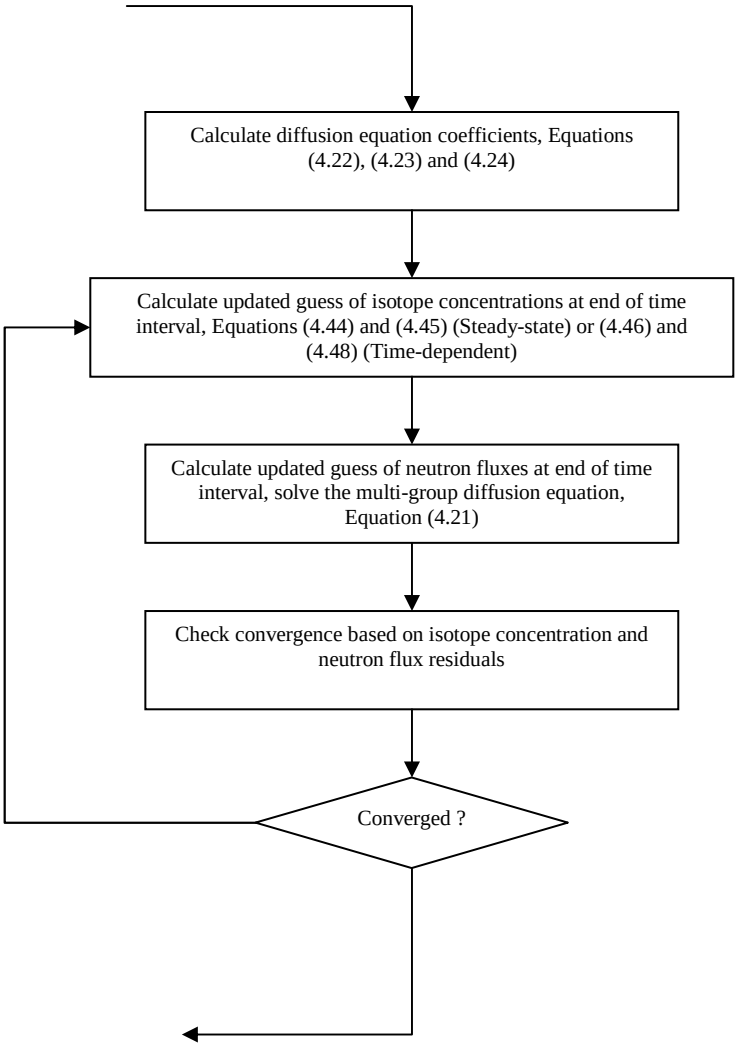



***Figure 6: Algorithm for the Inner Iteration***

## 4.5 *Closure*

In this chapter, a set of equations was presented in a form suitable for direct implementation in OpenFOAM. These equations, based on the TINTE code, include the discretized multi-

group diffusion equation, and equations for delayed neutron treatment, fission product poisoning and power production. A set of algorithms for the neutron flux solution, and for full steady-state and transient solutions were proposed. The OpenFOAM implementation, based on the equations and algorithms of this chapter, is discussed in chapter 5.

# 5. IMPLEMENTATION DESCRIPTION

Based on the equations and algorithms proposed in chapter 4, a FOAM multi-group diffusion solver, called diffusionFoam, was coded in C++. Significant effort was devoted to ensuring that an object-oriented approach to the coding was followed. Specifically, the code was modularized into a number of classes. In the interest of being concise, detailed information on all aspects of the implementation have not been provided, however in certain instances examples have been provided to illustrate the methods used and to emphasise the advantages of the FOAM framework.

## 5.1 *Class Structure*

In total, nine custom classes were created to model different aspects of the nuclear calculation being performed. An attempt has been made, as far as possible, to separate the various nuclear phenomena being modelled. In this way future development will allow different models for each phenomena to be applied, without introducing unnecessary complication. Class diagrams for the diffusionFoam application are given in Figure 7 and Figure 8. A cross-reference between the equations of chapter 4 and the diffusionFoam class and namespace members is given in Table 6.

### 5.1.1 nuclearField Class

The `nuclearField` class is primarily concerned with global nuclear parameters, such as k-effective and global power production. It contains several child `fluxGroup` objects, each responsible for the storage of the spatially- and time-dependent scalar neutron flux and flux leakage for a single broad group. Future development will likely see these broad group fluxes, as well as calculations such as neutron production, fission rates, total leakage, among others moved as children into separate objects. Similarly, at present, power production is calculated within this class. If decay and/or non-local power production is to be taken into account this should be included in a separate class.

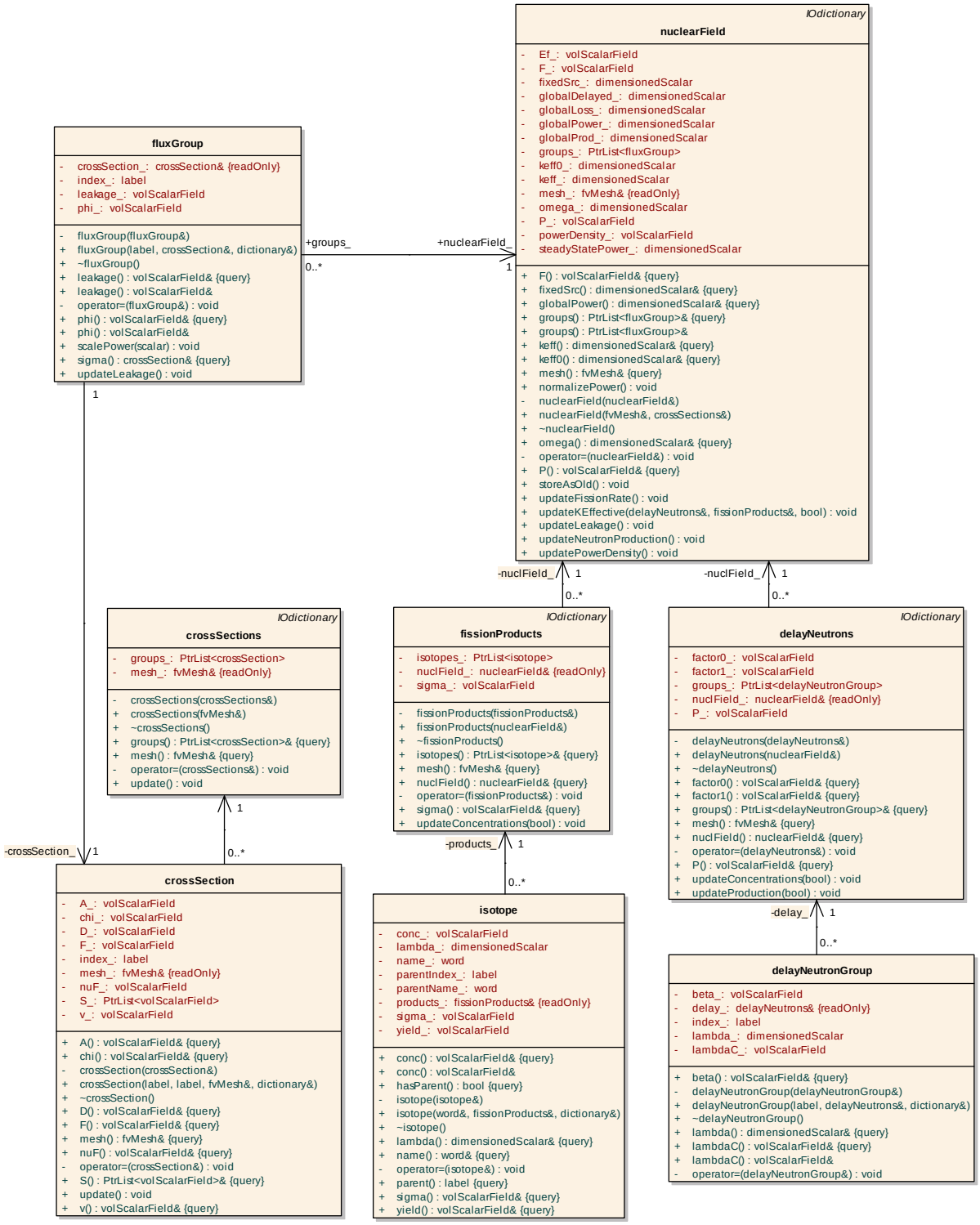



***Figure 7: The diffusionFoam Class Structure***

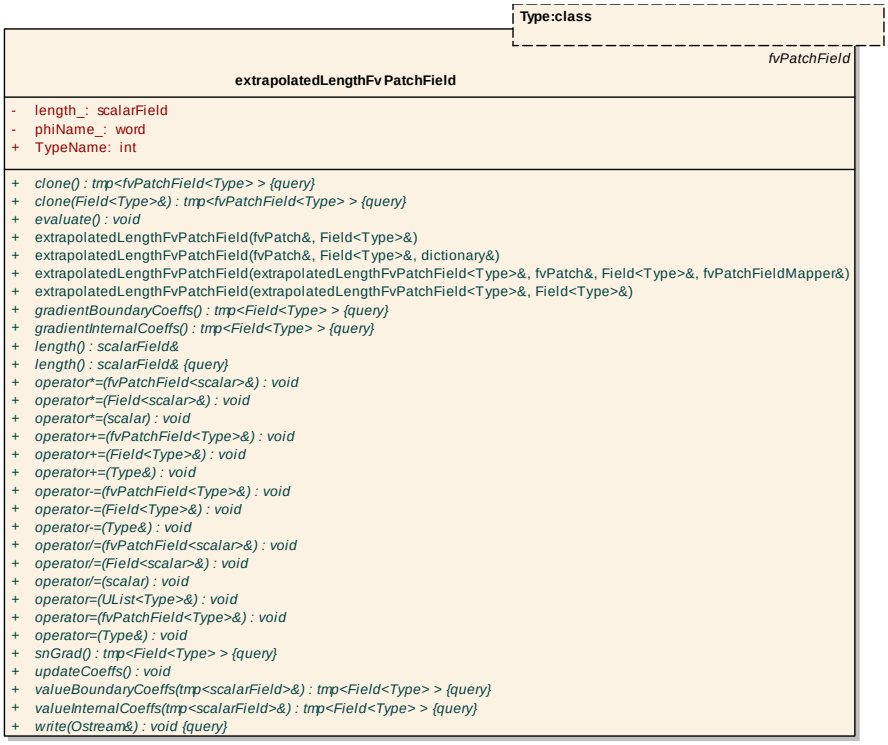



***Figure 8: The diffusionFoam Class Structure (continued)***

***Table 6: diffusionFoam Member Function and Equation Cross-References***

| Reference Equation | Class or Namespace Member |
|---|---|
| (4.2) | `nuclearField::updateFissionRate`<br>`nuclearField::updateProduction` |
| (4.11), (4.13) | `delayNeutrons::updateConcentrations` |
| (4.22), (4.23), (4.24) | `Foam::innerIteration` |
| (4.18), (4.19), (4.20),<br>(4.25), (4.26), (4.27) | `delayedNeutrons::updateProduction` |
| (4.21) | `Foam::transportSolve`<br>`Foam::groupSolve` |
| (4.31) | `nuclearField::updateKEffective` |
| (4.32) | `extrapolatedlengthFvPatchField::evaluate`<br>`extrapolatedlengthFvPatchField::valueInternalCoeffs`<br>`extrapolatedlengthFvPatchField::valueBoundaryCoeffs` |
| (4.33) | `extrapolatedlengthFvPatchField::snGrad`<br>`extrapolatedlengthFvPatchField::gradientInternalCoeffs`<br>`extrapolatedlengthFvPatchField::gradientBoundaryCoeffs` |
| (4.44), (4.45), (4.46), (4.48) | `fissionProducts::updateConcentrations` |
| (4.49) | `nuclearField::updatePowerDensity` |

### 5.1.2 crossSections Class

The `crossSections` class is primarily a container class for the neutron cross-sections and other diffusion related constants. It is invisaged that this class will ultimately include more advanced cross-section library functionality such as the collapsing of cross-sections, etc. A single `crossSection` object is defined for each broad energy group. Each `crossSection` object is responsible for supplying the spatially-dependent macroscopic absorption, fission, nu-fission, and scattering cross-sections, as well as diffusion constant, mean neutron velocity and fission spectrum for a single broad energy group. Currently, fixed value cross-sections are used but the structure is in place for more advanced cross-section calculations to be implemented.

### 5.1.3 delayNeutrons Class

The `delayNeutrons` class is responsible for providing the delayed neutron production terms for the neutron diffusion equation. These include steady-state and transient spatial prompt neutron production factors and the delayed neutron production. The class contains one or more `delayNeutronGroup` objects, representing each of the delayed neutron precursor groups. Each precursor group object is responsible for updating its own precursor concentration.

### 5.1.4 fissionProducts Class

The `fissionProducts` class is responsible for providing updated macroscopic absorption cross-sections for, and calculating updated concentrations of fission products. One or more child `isotope` objects are defined, each representing a single isotope. Each isotope object is responsible for calculating its updated concentration and macroscopic neutron absorption cross-sections. The current implementation is limited to the iodine and xenon type neutron poisons, with only a single parent and daughter isotope. The class structure is such that detailed decay chain calculations could potentially be carried out.

### 5.1.5 extrapolatedLengthFvPatchField Class

This class is derived from the FOAM `fvPatchField` class, providing the underlying code for an extrapolated length boundary condition, identified by the keyword `extrapolatedLength` in diffusionFoam. The internal operation of the class will not be discussed, however, it is necessary to explain that each `extrapolatedLength` boundary condition is responsible for updating its own extrapolated length values, given the name of a `volScalarField` from which to obtain diffusion length values (`extrapolatedLengthFvPatchField::phiName_`). This would, as a general rule, be the same name as the diffusion length associated with each `crossSection` object (`crossSection::D_`), although this is not enforced in the code.

While such flexibility may seem redundant in this case, since the extrapolated length will always be a function of diffusion length, it serves to illustrate how more complex coupling schemes may be achieved at mesh boundaries using FOAM.

## 5.2 *User Input*

A brief description of FOAM input and output is provided in section 3.8. A graphical representation of the structural layout of a typical diffusionFoam case is given in Figure 9. The diffusionFoam implementation takes full advantage of the input/output libraries of FOAM. In particular, each of the classes described in section 5.1 is assigned a unique dictionary in the constant directory, with the same name as the class. This dictionary contains all the necessary initialization data for the class. Consider, as an example, the following input dictionary for the `fissionProducts` class.

```
isotopes
(
    Xe135
    {
        parent       I135;
        lambda       lambda      [0 0 -1 0 0 0 0]   2.116E-5;
        yield        yield_Xe135;
        sigma        sigma_Xe135;
    }

    I135
    {
        parent       none;
        lambda       lambda       [0 0 -1 0 0 0 0]   2.883E-5;
        yield        yield_I135;
        sigma        sigma_I135;

    }
}
```

Here, we can see that the decay chain of the isotopes $^{135}$Xe and $^{135}$I are defined, including their decay constants and the names of the fission yield fraction and microscopic absorption cross-section dictionaries for each isotope. Thus, any number of isotopes may be defined in an easily understood and readable format.

## 5.3 *Known Issues*

For reasons discussed in section 4.4, the simple predictor-corrector arrangement proposed in section 4.4.1.3 was implemented as an initial attempt to obtain stable multi-group solutions. This algorithm was found to be unstable for multi-group time-dependent calculations. Therefore the current implementation allows time-dependent calculations in one energy group only.

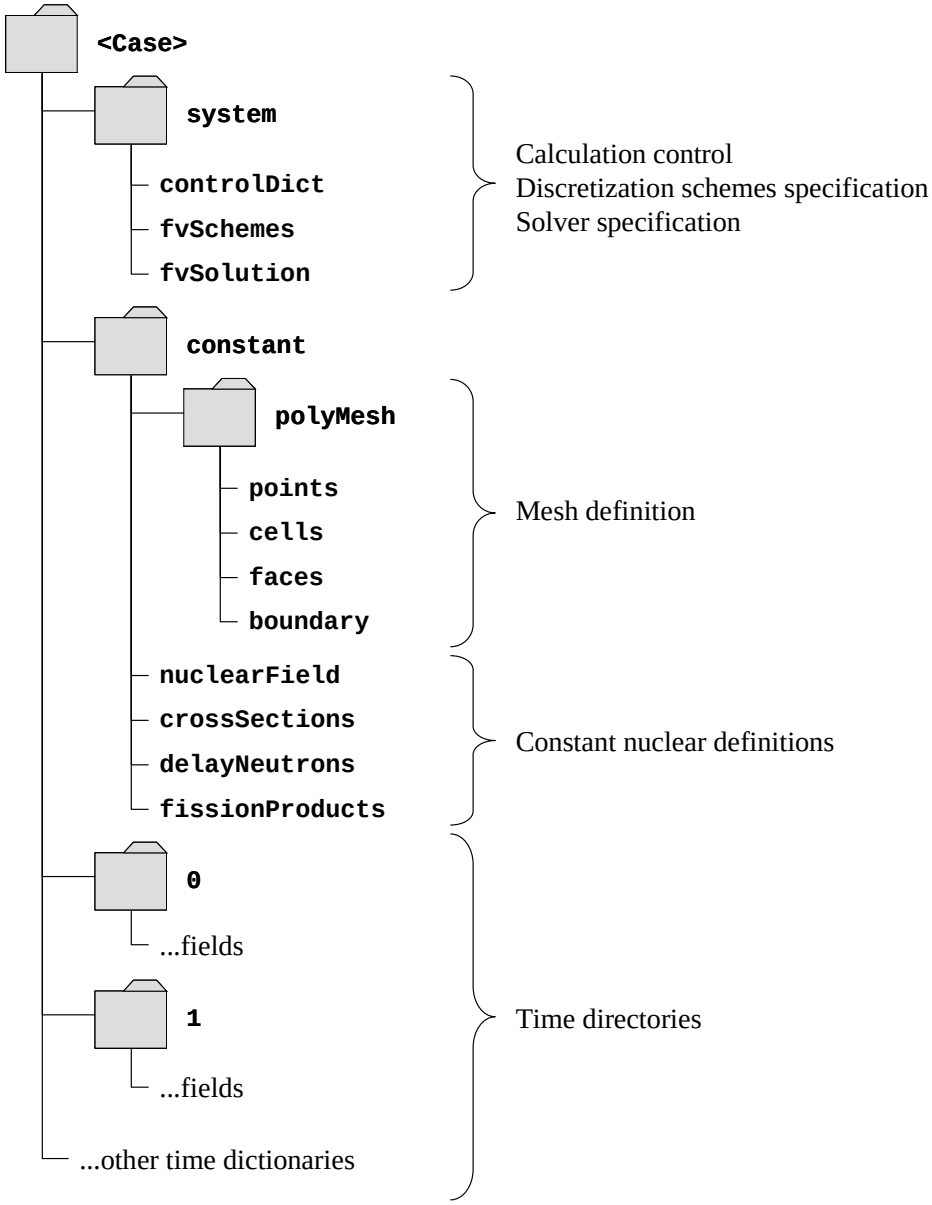



***Figure 9: Structural Layout of a Typical diffusionFoam Case***

## 5.4 *Closure*

In this chapter, the implementation for the OpenFOAM-based diffusion solver, called diffusionFoam, was described. During solver development, significant effort was devoted towards ensuring that an object-oriented approach was followed. This solver is known to be unstable for time-dependent multi-group calculations. A number of test calculations and their results, using the diffusionFoam solver, are given in chapter 6. Chapter 6 also includes further discussion which is based on the knowledge gained in this and previous chapters.

# 6. RESULTS AND FURTHER DISCUSSION

In order to test the diffusionFoam implementation of chapter 5, numerical solutions to a number of test cases have been obtained using the code. The test cases have been chosen so as to envelop the main features of the code, and numerical solutions are compared with analytical or other numerical solutions. These comparisons are presented in this chapter. Section 6.1 includes initial steady-state comparisons for simple one-group reactor models. This is then extended to more advanced non-homogenous two-group reactor models in 6.2. In section 6.3, short term and medium term dynamics are tested for the cases of step reactivity insertion and load-following. Additional discussions around the known issues of 5.3 as well as around questions 2, 3 and 4 of Chapter 1 are included in section 6.4 of this chapter.

## 6.1 *Steady-State Analytical Comparisons*

Analytical criticality conditions are readily available for a number of simple geometries, including spherical, block and cylindrical reactors (Stacey 2001), for the case of fixed uniform cross-sections. The criticality conditions are given in terms of a geometric buckling $B_g^{\ 2}$ as follows.

$$k = 1 = \frac{\frac{\nu\Sigma_f}{\Sigma_a}}{1 + L^2 B^2} \tag{6.1}$$

where $L = \sqrt{\frac{D}{\Sigma_a}}$ is the diffusion length.

These analytical benchmarks formed the basis of initial tests carried out using the diffusionFoam implementation. The geometric bucklings and flux profiles for the three simple geometries mentioned above, as well as chosen critical dimensions for typical PWR cross sections are given in Table 7.

Numerical steady-state solutions were obtained using diffusionFoam for each of these cases, where the analytical reactor is critical ($k = 1$). The results are summarized in Table 8. From the results shown, it is clear that the steady-state solver is operating correctly for simple cases, with zero flux at the boundaries. In all cases, the difference in k-effective between the analytical and numerical solutions is sufficiently small that it can be attributed to numerical discretization error.

***Table 7: Criticality Conditions for Some Simple Bare Reactors***

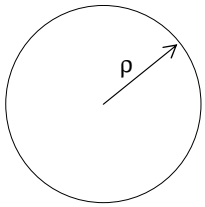
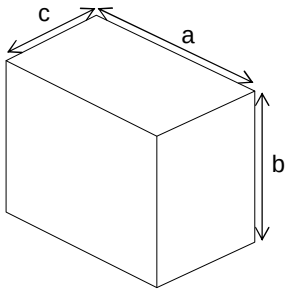
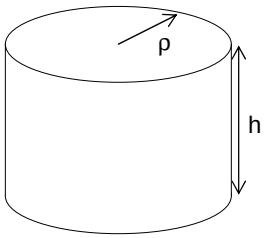
| | **Sphere** | **Block** | **Finite Cylinder** |
|---|---|---|---|
| Geometry |  |  |  |
| Geometric Buckling | $\left(\frac{\pi}{\rho}\right)^2$ | $\left(\frac{\pi}{a}\right)^2+\left(\frac{\pi}{b}\right)^2+\left(\frac{\pi}{c}\right)^2$ | $\left(\frac{2.405}{\rho}\right)^2+\left(\frac{\pi}{h}\right)^2$ |
| Flux Profile | $\frac{1}{r}\sin\frac{\pi r}{\rho}$ | $\cos\frac{\pi x}{a}\cos\frac{\pi y}{b}\cos\frac{\pi z}{c}$ | $J_0\frac{2.405r}{\rho}\cos\frac{\pi z}{h}$ |
| Diffusion Length $D$ | 10 cm | 10 cm | 10 cm |
| Absorption Cross Section $\Sigma_a$ | 0.15 $cm^{-1}$ | 0.15 $cm^{-1}$ | 0.15 $cm^{-1}$ |
| Nu-fission Cross Section $\nu\Sigma_f$ | 0.16 $cm^{-1}$ | 0.16 $cm^{-1}$ | 0.16 $cm^{-1}$ |
| Critical Dimensions | 99.35 cm | 200 x 150 x 177.1 cm | ρ=120, h=128.43 |

***Table 8: Summary of diffusionFoam Results for Steady-state Analytical Benchmarks***

| | **Sphere** | **Block** | **Finite Cylinder** |
|---|---|---|---|
| Mesh dimensions | 50 radial | 30 x 30 x 30 | 50 radial, 50 axial |
| k-effective | 0.99999 | 1.00006 | 1.00001 |
| Error [ $\Delta k \times 10^5$ ] | -1 | +6 | +1 |

## 6.2 *Steady-State Benchmark Comparisons*

### 6.2.1 The Dodds Benchmark

The Dodds benchmark problem (ANL-7416 1977) is a set of pure neutronic calculations for a two-dimensional axisymmetric (r-z) reactor model. The benchmark is intended to test two-dimensional neutron kinetics solutions, and consists of an initial steady-state eigenvalue calculation followed by a supercritical transient with six-group delayed neutron feedback. Relevant reactor parameters for the steady-state calculation are given in Table 9 and the layout of the reactor is depicted in Figure 10.

***Table 9: Dodds Benchmark Steady-State Parameters***

| Parameter | Value |
|---|---|
| Number of radial meshes | 18 (equally spaced) |
| Number of axial meshes | 28 (equally spaced) |
| Number of broad energy groups | 2 |
| Reactor width | 235.61 cm |
| Reactor height | 524.87 cm |
| Boundary conditions | Zero-flux |
| Number of material types | 9 |
| Number of material regions | 16 |
| Benchmark k-effective | 0.867053 |

A steady-state solution to this benchmark using the TINTE code is available (Strydom 2004). Since the underlying theory of diffusionFoam is based on the TINTE code theory, the results are expected to match closely.

For reasons discussed in chapter 5, the time-dependent solution to this two-group problem could not be obtained with the currently implemented predictor-corrector algorithm. The steady-state solution was, however, calculated using diffusionFoam, using a mesh refinement of six fine meshes per coarse mesh in both the radial and axial directions. Comparisons of steady-state results with both the TINTE code and the benchmark reference result are given in

Table 10 and Figure 11. The results of TINTE and diffusionFoam compare very well. There is an eigenvalue difference of less than $1\times10^{5}$ between the two codes, and the flux profiles show negligible differences.

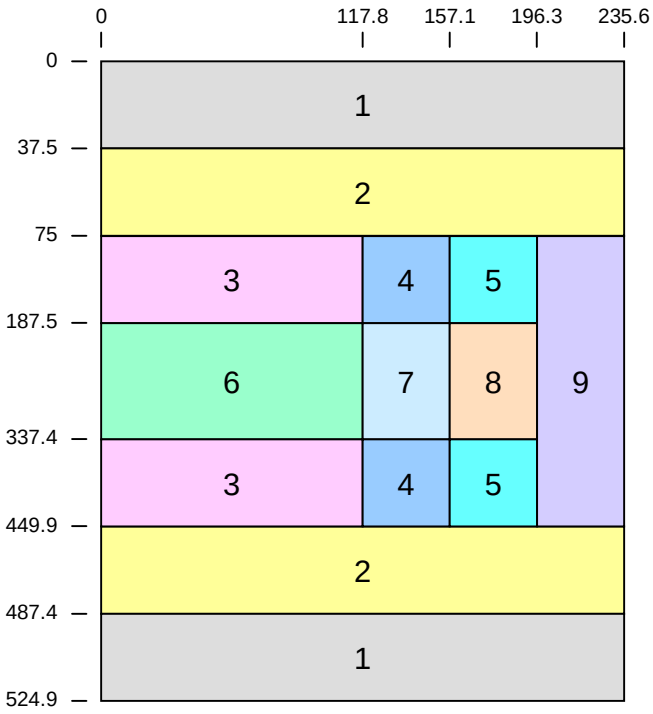
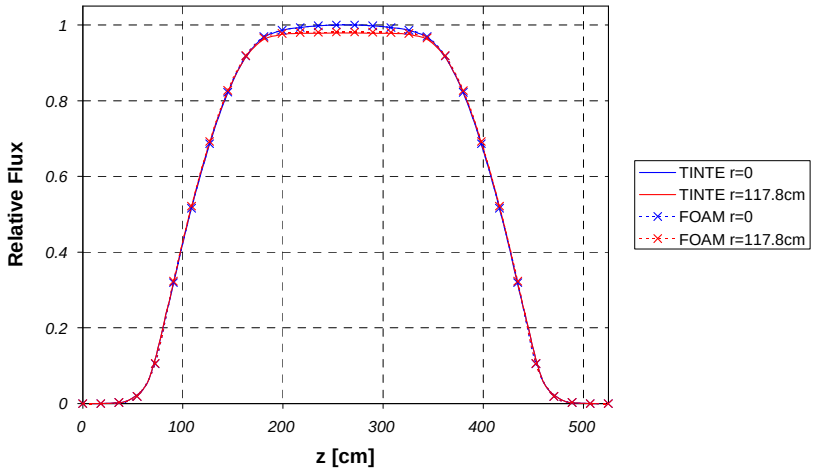


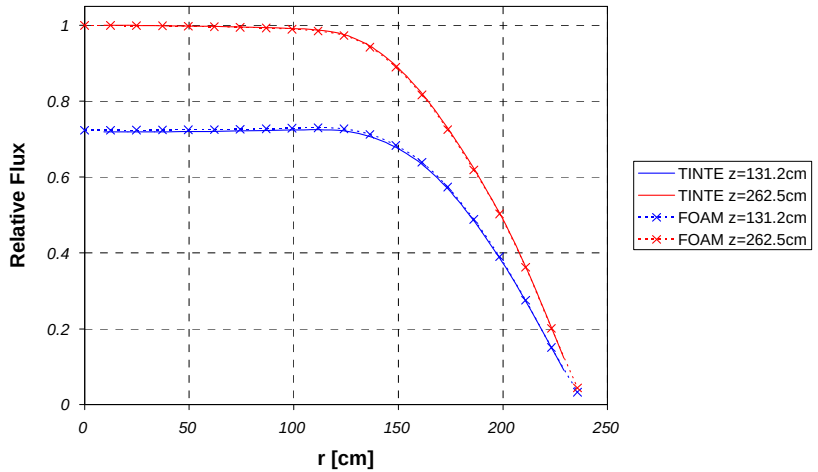

***Figure 10: Dodds Benchmark Steady-State Reactor Layout***

***Table 10: K-effective Comparison for the Dodds Benchmark***

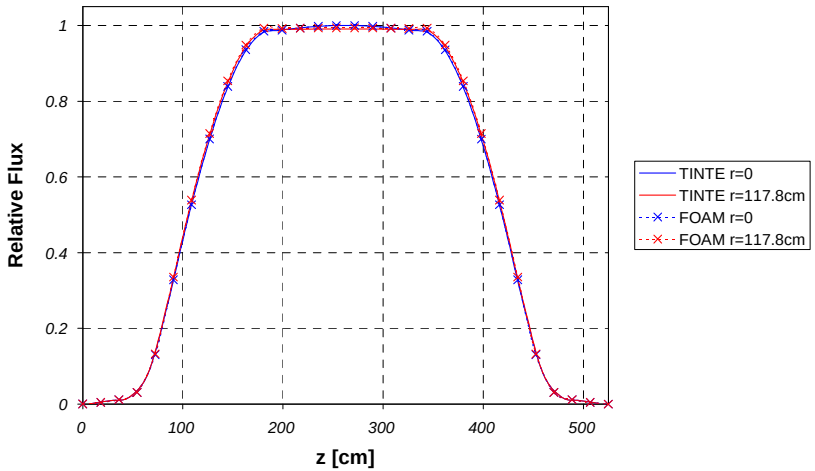
| | **Reference** | **TINTE** | **diffusionFoam** |
|---|---|---|---|
| k-effective | 0.867053 | 0.867433 | 0.867442 |
| Difference [ $\Delta k \times 10^{5}$ ] | - | +38 | +39 |



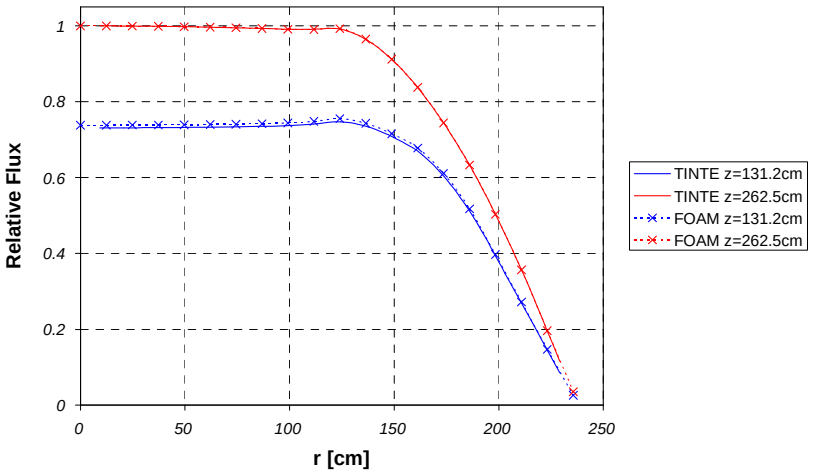

*(a) Fast Flux Axial Profiles*




*(b) Fast Flux Radial Profiles*




*(c) Thermal Flux Axial Profiles*




*(d) Thermal Flux Radial Profiles*

***Figure 11: diffusionFoam and TINTE Steady-State Flux Profile Comparisons for the Dodds Benchmark***

### 6.2.2 The OECD PBMR Benchmark

The Nuclear Energy Agency, within the Organization for Economic Co-operation and Development (OECD), has published the OECD PBMR benchmark (Reitsma et. al. 2004), in which a set of steady-state and transient calculations for the PBMR HTR are defined. The reactor is modeled in two-dimensional axisymmetric (r-z) geometry. A total of 190 nuclear materials are defined in 580 nuclear calculation regions. The model layout is shown in Figure 12. A two-group structure is defined. The benchmark defines the reactor geometry on a structured rectangular coarse mesh, indicated by softer lines in Figure 12.

In this section, case 1 of the benchmark is considered. This is a pure neutronic steady-state calculation using fixed cross-section sets. A model for the case was created using a mesh refinement of four fine meshes per coarse mesh in both the radial and axial directions. The steady-state k-effective for this model was compared with TINTE results for the same case, using the same mesh structure. These comparisons were also done for the case of eight fine meshes per coarse mesh. The results of these comparisons are given in Table 11.

***Table 11: K-effective Comparison for the OECD PBMR Benchmark***

| Case | Parameter | TINTE | diffusionFoam |
|---|---|---|---|
| Four fine meshes per coarse mesh | K-effective | 0.99821 | 0.99745 |
| | Difference [ $\Delta k \times 10^5$ ] | - | -76 |
| Eight fine meshes per coarse mesh | K-effective | 0.99869 | 0.99803 |
| | Difference [ $\Delta k \times 10^5$ ] | - | -66 |

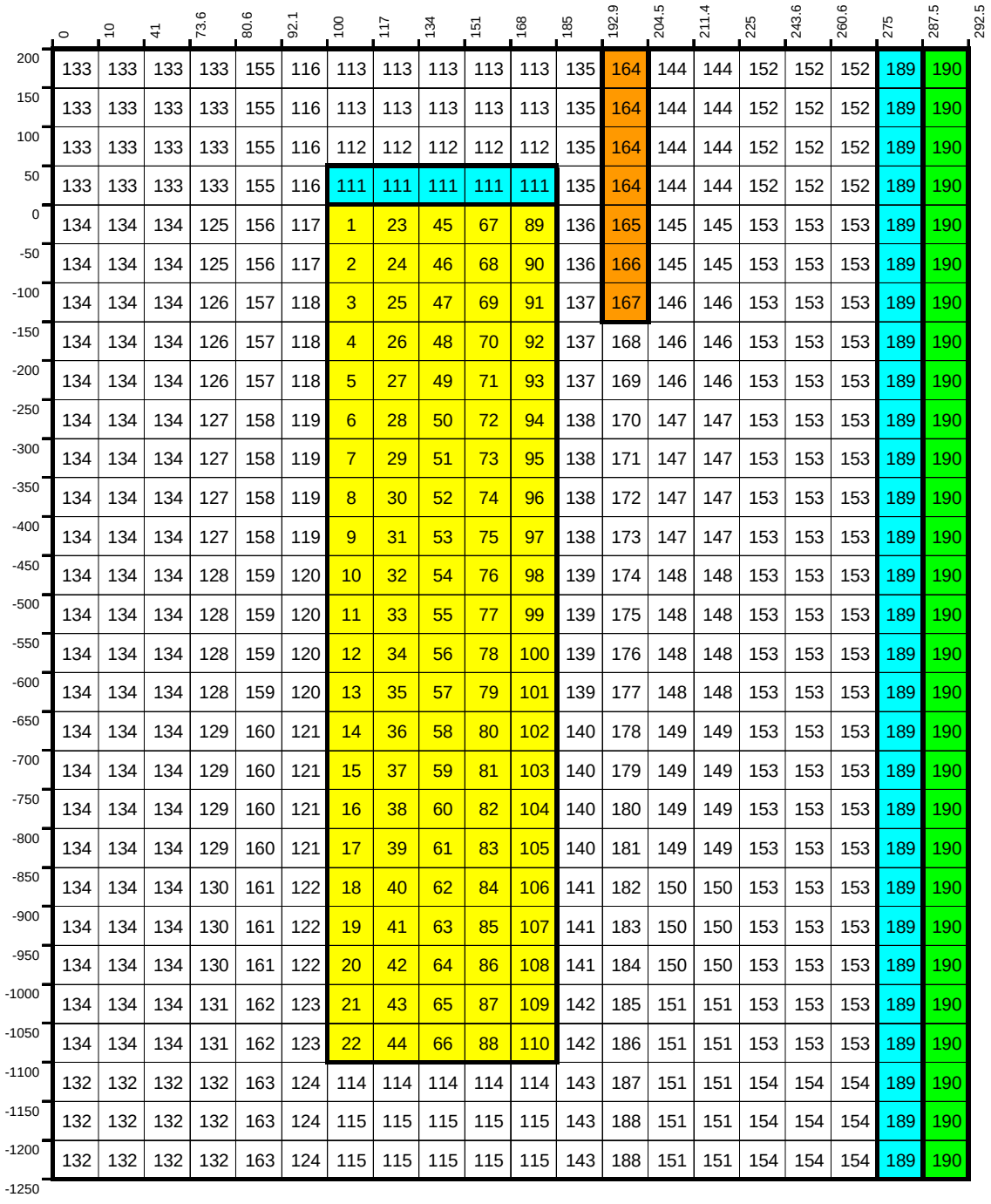



***Figure 12: PBMR OECD Benchmark Steady-State Reactor Layout***

From these results it is clear that there is a difference of approximately 70 pcm between TINTE and diffusionFoam results. Possible reasons for these differences include:

- Differences between the discretization methods employed by each solver. FOAM employs finite-volume discretization while TINTE employs a variant of the finite-difference discretization.

- The current diffusionFoam implementation does not support directional diffusion constants. For this reason, non-directional diffusion constants in the void regions were approximated, based on the specified benchmark values.

## 6.3 *Time-Dependent Comparisons*

As was discussed in section 5.3, the simple predictor-corrector solution algorithm of section 4.4.1.3 was found to be unable to ensure solution stability for time-dependent multi-group cases. In the absence of a block-coupled solver, no time-dependent multi-group solutions could be obtained using the current diffusionFoam implementation. In order to demonstrate the potential of the modern multi-physics approach to these classes of problems, however, a number of time-dependent one-group calculations were carried out.

### 6.3.1 Short Term Dynamics - Reactivity Insertion

The bare sphere model of section 6.1 was modified to include delayed neutrons. The delayed neutron parameters of Table 12 were assumed, and a mean neutron velocity $v$ of $10^6$ cm/s was assumed. Calculations were then carried out for the first 10 s of reactivity insertion events. Both positive and negative step reactivity insertions of 100 pcm and 200 pcm were considered. Additionally, each calculation was repeated for the case of constant precursor concentrations, so that the prompt jump could be shown without any delayed neutrons influences. No supercritical insertion was considered because no reactivity feedback model has currently been implemented in diffusionFoam. The results of all reactivity insertion cases are summarized in Figure 13. The numerical results are compared with analytical prompt jump approximation (PJA) solutions, which are derived in the next section. These analytical

solutions describe the initial jump and power gradient immediately after the reactivity insertion. The analytical PJA solutions overlaid on Figure 13 therefore only indicate the initial response of the reactor. No comparisons were made for later times.

***Table 12: Delayed Neutron Parameters for Reactivity Insertion Calculations***

| **Group** $l$ | **Decay Constant** $\lambda_l$ **[s$^{-1}$]** | **Fission Fraction** $\beta_l \times 10^3$ † |
|---|---|---|
| 1 | 3.87 | 0.179504 |
| 2 | 1.4 | 0.883712 |
| 3 | 0.311 | 2.809928 |
| 4 | 0.116 | 1.297952 |
| 5 | 0.03174 | 1.470552 |
| 6 | 0.01272 | 0.262352 |
| All | - | 6.904 |

The prompt jump is clearly visible in all cases, followed by the slower response of the six delayed neutron groups. For all cases of constant delayed neutron precursor concentrations, the prompt jump is clearly visible and, as expected, the power remains constant after this prompt jump. The results around the initial jump compare well with the analytical approximations obtained in the next section. Differences are seen beyond 0.5 s because the prompt jump approximation solution of the next section describes only the initial reactor response.

† Values used are taken from Table 3 and Table 4 for $U^{235}$.

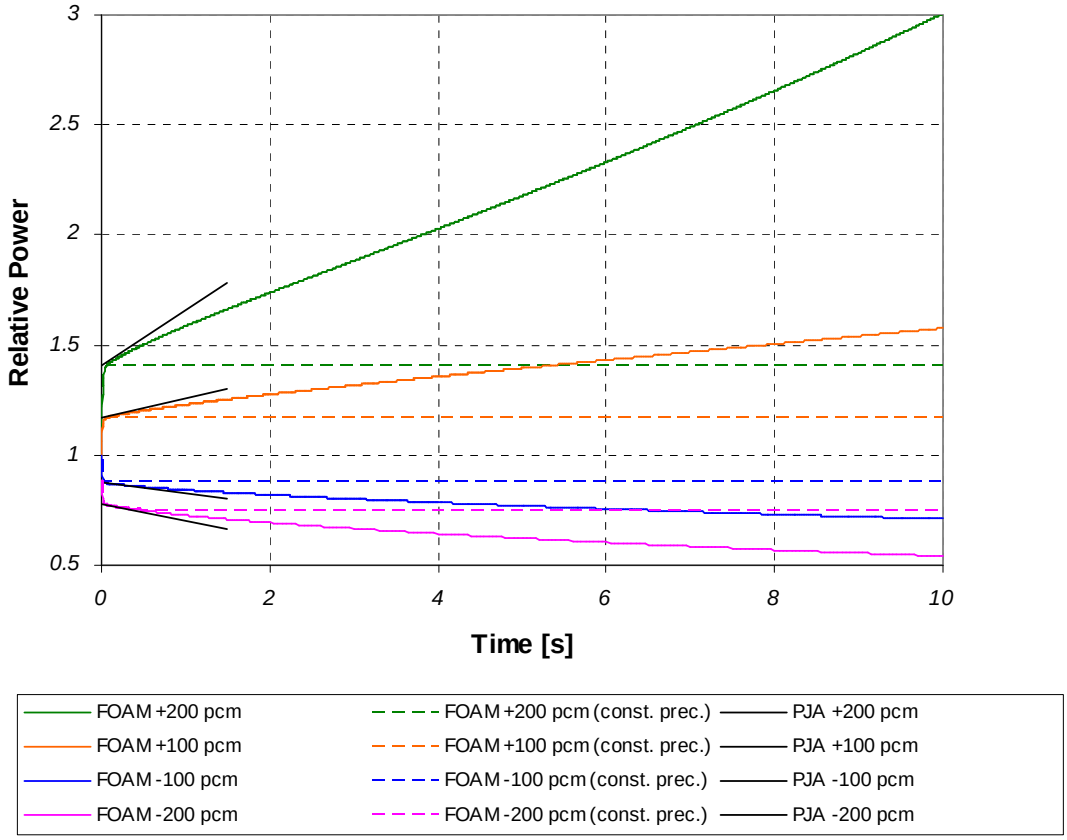



***Figure 13: Time Plot of Relative Power for Subprompt-critical Reactivity Insertions***

#### 6.3.1.1 Analytical Comparisons

Analytical approximations of the initial power response can be obtained using the prompt jump approximation (PJA) (Ott and Neuhold 1985). What follows is the calculation of the expected response, based on this approximation, for the reactivity insertion cases.

The neutron generation time $\Lambda$ for the model is calculated as

$$\Lambda = \frac{1}{v\nu\Sigma_f} = \frac{1}{10^6 \times 0.16} = 6.25 \times 10^{-6}\, s$$

The simplified point kinetics equation, independent of external sources is written

$$\frac{dQ}{dt} = \frac{\rho - \beta}{\Lambda} Q + \frac{1}{\Lambda} \sum_l \lambda_l \zeta_l$$

where $Q$ is the total reactor power and $\rho$ in the inserted reactivity. The precursor balance equation is written

$$\frac{d\zeta_l}{dt} = -\lambda_l \zeta_l + \beta_l Q$$

The prompt jump approximation (PJA) may be applied to determine the initial jump after a step reactivity insertion.

$$\left(\frac{Q_0^*}{Q_0}\right)_{PJA} = \frac{\beta}{\beta - \rho} \tag{6.2}$$

The rate of change of power, following the reactivity insertion and based on the point jump approximation, may be calculated according to

$$\left(\frac{dQ}{dt}\right)_{PJA} = \frac{\bar{\lambda}\rho}{\beta - \rho} Q_0^* \tag{6.3}$$

where the single group decay constant $\bar{\lambda}$ is calculated according to

$$\bar{\lambda} = \frac{1}{\beta}\sum_l \beta_l \lambda_l$$

For the data of Table 12, $\bar{\lambda} = 0.435\, s^{-1}$.

Solutions for the prompt jump and initial rate of change of power following the prompt jump, Equations (6.2) and (6.3), are given in Table 13. These initial power curves are superimposed on Figure 13 for comparison with the diffusionFoam results.

***Table 13: Point Jump Approximation Applied to the Reactivity Insertion Cases***

| **Inserted Reactivity** $\rho$ | **Prompt Jump** $\left(\frac{Q_0^*}{Q_0}\right)_{PJA}$ | **Initial Power Slope** $\left(\frac{dQ}{dt}\right)_{PJA}$ |
|---|---|---|
| +200 pcm | 1.408 | 0.25 |
| +100 pcm | 1.169 | 0.0862 |
| -100 pcm | 0.873 | -0.0481 |
| -200 pcm | 0.775 | -0.0758 |

### 6.3.2 Medium Term Dynamics – Load Follow

The OECD PBMR benchmark model of section 6.2.2 was collapsed to a single energy group model for the purposes of running transients using diffusionFoam. Case 4a of the OECD PBMR benchmark (Reitsma et. al. 2004) was run using this single group model. In this case, the $Xe^{135}$ behaviour is modelled for a typical 100%-40%-100% load follow. The reactor, initially at a steady-state power of 400 MW (100%), is ramped down to 160 MW (40%). After three hours of operation at this level, the reactor is then ramped back to full power. The control rods are kept at a constant position for the duration of the transient and the global reactivity is monitored. The benchmark calculation includes temperature feedback. This feedback was not modelled in diffusionFoam.

At each time-interval in the calculation, the k-effective was updated according to equation (4.31) and, based on this, an effective global reactivity was calculated. The time behaviour of global reactivity, as calculated using diffusionFoam, is shown in Figure 14, compared with the reference TINTE solution for this case.

The time-scales of the reactivity response compare well, i.e. the maximum reactivity occurs 6 h after the return to full power in both cases. There are significant differences (150 pcm) in the magnitudes calculated by diffusionFoam and TINTE. These can be attributed to modelling differences. The diffusionFoam solution was obtained by assuming a step change in reactor power, rather than the six minute ramp specified in the benchmark. No temperature feedback was modelled. A single energy group was assumed for the diffusionFoam calculation, which was obtained by collapsing from the two-group steady-state solution of section 6.2.2. There are also potentially differences in the $^{135}Xe$ and $^{135}I$ yields because TINTE does not allow custom values to be specified.

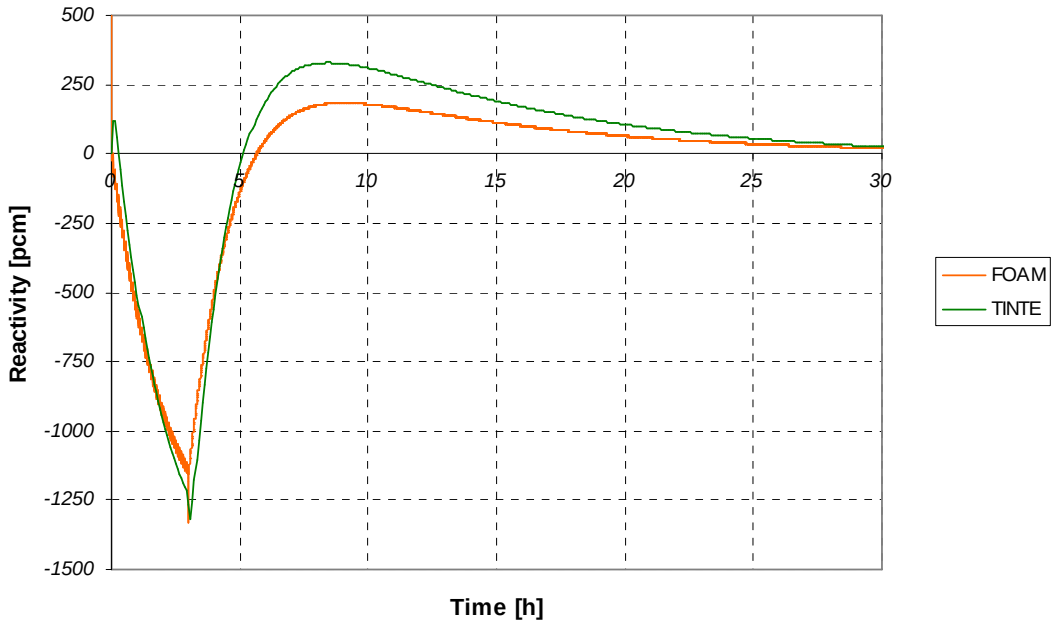



***Figure 14: Time Plot of Global Reactivity for Load Follow Transients***

## 6.4 *Further Discussion*

The results of sections 6.1 through 6.3 have shown that the diffusionFoam implementation, although still in an early stage of development is capable of solving a number of general reactor analysis problems. It is clear from this that the FOAM toolkit can successfully be applied to the solution of the spatial- and time-dependent neutron diffusion equation. In this section, we now turn towards answering questions 2, 3 and 4 of section 1.1, which relate to the advantages provided by a multi-physics toolkits such as FOAM and to implementing more advanced functionality in the toolkit. Also included in this section are discussions around particular issues which were encountered during the development of diffusionFoam.

### 6.4.1 Theoretical Modeling

Chapter 4 includes extensive derivations and descriptions of the necessary equations and algorithms for a FOAM-based multi-group diffusion solver, based on the TINTE code. In almost all cases, there is no significant difference from the TINTE equations. Any differences

have resulted from the multi-group assumption used, whereas TINTE uses a two-group assumption.

Of importance is that no expressions for modeling the spatial discretization were necessary. FOAM is responsible for handling the basic finite-volume discretization. This is quite an advantage. If one considers the TINTE theoretical description (Clifford 2007), a substantial portion of this is devoted to the spatial discretization and the matrix solver based on this discretization. It is clear from this that an object-oriented framework allows the developer to approach the problem from a higher level than does traditional code development.

Further, if the same approach is applied to the time-discretization of the delayed neutrons and saturation fission products, these too may also be approached from a higher level. In these cases it will, of course, be necessary to introduce suitable higher order time-discretization schemes as options in FOAM. This will not necessarily simplify the theoretical description but it will separate the task of implementing a higher order time-discretization scheme from that of implementing the global solution algorithm, i.e. a first-order assumption may initially be made and therefore the overall development of the solver is not held back until such time as this higher-order scheme is fully implemented and tested. The theoretical descriptions of some higher order time discretization schemes are available (Ferziger and Peric 2001), and have been successfully implemented in other finite-volume codes (Star-CD 2007). The FOAM implementation of typical higher order time differencing schemes such as the GAKIN and $\theta$ methods (Stacey 2001), used in reactor analysis, should be relatively straightforward tasks.

### 6.4.2 Block Coupled Solutions

For reasons discussed previously, the coupling of the group diffusion equations requires an implicit coupling. This requirement has a number of implications when considering development on any framework. This discussion is not limited to multi-physics toolkits alone; these considerations must be taken into account in any new solver.

The multi-group diffusion equations represent a block-point implicit set of PDEs, i.e. the group fluxes depend on each other in the same computational point but each group flux depends only on the neighbouring value of the same energy group. We therefore have a block matrix with many dense $G \times G$ matrices along the diagonal, and spatial coupling vectors scattered in the lower and upper matrices, as depicted in Figure 15.

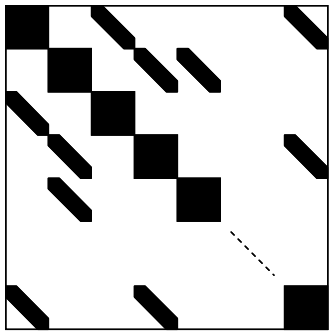


***Figure 15: Typical Block Matrix Layout for a Block-Point Implicit set of PDEs***

By structuring the matrix in this manner, matrix preconditioning remains effective. In the context of the FOAM and similar frameworks, however, the construction and solution of this block matrix requires additional effort. In particular, one must look at the method of parallelization employed by the toolkit. FOAM uses domain decomposition for parallelization of the solver and, in this case, it will be necessary to invest additional effort into extending this parallelization to block solutions.

Implicit equation coupling and block matrices are currently areas of development in FOAM (Jasak 2007). The coupled solution of vector and tensor equations is currently supported, and block-point implicit coupling is likely to be available in the future.

### 6.4.3 Higher Order Transport Methods

The diffusionFoam implementation is based on the diffusion approximation. Up to now the more advanced neutron transport methods or their applicability to general multi-physics toolkits has yet to be discussed. In exploring the potential for the deterministic solution of the neutron transport equation using multi-physics toolkits we will restrict ourselves to the

discrete-ordinates (S-N) methods. Discussions around the spherical harmonics methods are excluded because of their relative mathematical complexity, but the principles discussed still apply. The discrete-ordinates methods essentially discretize the angular domain (direction of neutron flow) into a number of fixed directions or ordinates. This is not dissimilar to the multiple energy group approach; the number of coupled equations now becomes multiplied by the number of discrete ordinates. A simplified representation of the steady-state discrete-ordinates equation is given below (Stacey 2001).

$$\mathbf{\Omega}_k \bullet \nabla \psi_{k,g} + \sigma_{t,g}\psi_{k,g} = \sum_{g'=1}^{G} \sigma_s^{g' \to g} \sum_{k'=1}^{K} w_{k'} \psi_{k',g'} + S_{k,g} \ , \quad k = 1,\ldots,K \ , \quad g = 1,\ldots,G$$

where

$\mathbf{\Omega}_k$ is the k[th] ordinate unit vector

$\psi_{k,g}$ is the angular flux for the k[th] ordinate and g[th] energy group

$\sigma_{t,g}$, $\sigma_s^{g' \to g}$ are the g[th] energy group microscopic total and in-scattering cross-sections

$w_k$ is the k[th] ordinate quadrature weight describing the between ordinate scattering dependency

$S_{k,g}$ is the source term including fission and fixed sources.

In the general neutron transport equation, the diffusion term is replaced by a streaming operator $\mathbf{\Omega} \bullet \nabla \psi(\mathbf{\Omega}, \mathbf{r}, t)$. The angular domain is discretized into discrete values $\mathbf{\Omega}_k$, chosen such that they correspond with the angular fluxes $\psi_k$. The streaming operator is written as $\mathbf{\Omega}_k \bullet \nabla \psi_k(\mathbf{r}, t)$. The operator is now in a form suitable for applying any of the finite-difference, finite-volume, finite-element, etc. formulations. FOAM does not currently include this particular operator, however the implementation of this operator will be a relatively straightforward task after a suitable finite-volume formulation is derived.

Methods are also required to simplify the scattering integral in the transport equation. A common approach is to approximate the scattering source using Legendre polynomials (Stacey 2001). Using this approach, the integral reduces to a sum involving explicit

coefficients (quadrature weights $w_k$). No special treatment is required for these terms when applying the finite-volume or other methodologies.

It is clear that, provided a suitable discretized formulation for the streaming term can be obtained, the discrete-ordinates method is readily applicable to any general multi-physics toolkit.

### 6.4.4 Higher Order Spatial Discretization Schemes

The relatively large computational requirements of deterministic reactor analysis have led researchers to study methods of improving the computational accuracy on coarse meshes. This has led to the development of a number of higher-order spatial discretization schemes. Of these, the nodal (Wagner 1979) (Lawrence and Dorning 1979) (Shober et. al. 1986) (Hutt and Knight 1990) (Turinsky et. al. 1994) and finite-element methods (Kang and Hansen 1973) (Ciarlet 1978) (Lautard 1994) (Van Criekingen 2007) are in common use. Sutton and Aviles provide a good general overview of the higher order methods available for solving the time-dependent group diffusion equation (Sutton and Aviles 1996).

The nodal methods are ideally suited to lattice-type calculations where representative cross-sections for each large node are obtained using an assembly calculation. In this respect they are used particularly in light-water reactor analysis, where a node can be defined for each rectangular fuel assembly. The nodal methods have generally been restricted to structured rectangular meshes in the past. Recent development has been made into hexagonal nodal methods for the cases of hexagonal lattice structures such as those found in VVER reactors and block HTRs (Jin and Chang 1998) (Bangyang and Zhongsheng 2006). This development is, however, for the case of structured orthogonal meshes.

For reactor analysis calculations using unstructured meshes, the finite-element formulation has more commonly been used (Lucas et. al. 2004). This is not to say that the finite-volume method is not suited to reactor analysis problems. Rather, if one takes the point of view that the nodal, finite-volume and finite-element methods can be written in mathematically

equivalent forms, it is then possible to represent the finite-element and nodal methods using a higher order finite-volume discretization. A number of examples of this generalized point of view are available. For example, Grossman and Hennart consider the finite-element formulation to be a general discretization technique, and as such have successfully applied it to the nodal methods (Grossman and Hennart 2007). Similarly, Chavent combined the advantages of the finite-volume and finite-element methods into a single numerical procedure, by using mixed-hybrid finite-element and Godunov's methods (Chavent et. al. 1997). Numerous other studies have also been carried out into higher order finite-volume methods and their relation to finite element methods (Baranger et. al. 1996) (Aboubacar and Webster 2000).

These studies are generally not aimed at reactor analysis problems, however, and research will most likely be necessary to derive these higher order finite-volume discretizations for implementation in a finite-volume toolkit such as FOAM. FOAM provides many of the features necessary for implementing such higher-order discretization schemes. The FOAM toolkit allows fields of values to be defined at cell-centers, mesh faces and at mesh vertices. The stressFemFoam application is an example of a finite-element implementation in FOAM.

### 6.4.5 Other Numerical Issues

Apart from the diffusion equation coupling problems discussed in previous sections, numerous other numerical problems were encountered during testing and execution of diffusionFoam.

- Steady-state convergence of the more complex models is slow. The TINTE code system generally provides a converged steady-state solution with 50 iterations. The diffusionFoam code currently requires significantly more (several hundred) iterations than this for convergence. This is largely because the maximum pseudo-transient time interval is limited by solution stability in diffusionFoam. More attention should be paid towards optimizing the numerics of the steady-state solution. In particular, the

pseudo-transient algorithm used by the TINTE code could be replaced with the a more traditional fixed-source iteration method.

- The convergence of the inner iteration during time-dependent calculations requires optimization. During the diffusion equation solution FOAM, by default, adds a portion of the prompt neutron production term as an explicit source to ensure stability during the matrix inversion. A properly converged solution therefore requires iteration outside the matrix solution. This is done in the inner iteration loop of diffusionFoam. A more advanced method to linearise the prompt neutron production could potentially improve the rate of convergence.

- The time-dependent and pseudo-transient steady-state calculations were carried out based on user-specified time interval values. The introduction of a time interval controller, which optimizes the time intervals based on the reactor period and other parameters, will assist in reducing the number of time intervals required for a given calculation.

The above problems are not specific to FOAM or any other multi-physics toolkits. It is likely that any new implementation, on any platform, will require significant effort to optimize the numerics of the problem.

## 6.5 *Closure*

In this chapter, numerical solutions to a number of test cases were obtained using the diffusionFoam code. The test cases were chosen so as to test the main features of the code, from simple steady-state solutions to more complex time-dependent solutions involving short and medium term dynamics. The numerical solutions were compared to analytical or other numerical solutions. In all cases the solver performed adequately. Based on this we can conclude that the FOAM implementation of a time-dependent diffusion solver was successful.

The main numerical issues surrounding the diffusionFoam code were then discussed. The potential for and issues surrounding the implementation of a block solver in FOAM were

discussed, as was the potential for implementing more advanced transport calculations and higher order discretization schemes. In these discussions the potential for applying multi-physics toolkits to other, more advanced, classes of reactor analysis problems is shown. In chapter 7 the main conclusions from this and previous chapters are summarized.

# 7. CONCLUSIONS

The basic implementation of a time-dependent diffusion solver was created using the FOAM toolkit. This new implementation, called diffusionFoam, includes models for delayed neutrons as well as fission product poisoning by saturation fission products such as $^{135}$Xe. Fixed value cross-sections were assumed. This solver was shown to function well for two-group steady-state calculations and one-group time-dependent calculations.

In the development of this solver, a subset of the theoretical basis for the TINTE code was rederived in such a way as to be compatible with the FOAM framework. Based on this theoretical description, a data structure was defined and a number of container classes were then created. The resulting implementation is an example of an object-oriented, multi-physics approach to reactor analysis solver development. While there is still scope for improvement and outstanding issues, the key benefits and disadvantages of such an approach have been explored to some depth.

The FOAM toolkit has shown great potential for the solution of general reactor analysis problems. The initial literature survey showed FOAM to be a general numerical toolkit, which had the potential for solving reactor analysis classes of problems. Further research has shown, rather, that the greatest benefit of using such a framework is through the software design approach applied. When creating a solver using such a framework, the developer inherently seeks to modularize the code. FOAM includes a fixed number of container types; `scalar`, `dimensionedScalar`, `Field` of `scalar` values and a `Field` of `dimensionedScalar` values. Inherent to each of these objects is the functionality to read and write data to/from file, for mathematical expression evaluation and full error handling. The code developer is therefore responsible for identifying how these variables interact with each other, and structuring them so as to take advantage of these interactions. It is clear that this object-oriented approach to coding does provide advantages in terms of the development and maintenance of complex reactor analysis codes.

The theoretical description of chapter 4 and further discussions in section 6.4.1 have shown that the approach followed in deriving suitable equations for the FOAM framework is virtually indistinguishable from the approach followed in the case of the TINTE code. A distinguishing feature of the object-oriented approach is that physical equation derivations are carried out independently from those for the spatial and time discretization schemes. Thus we can see that in order to take advantage of the object-oriented structure of the framework, it is necessary to modularize the theoretical basis.

A number of test calculations were carried out to validate the accuracy of the diffusionFoam solver.

- Steady-state eigenvalue comparisons were made for three simple bare reactors, namely spherical, block and finite-cylinder reactors, in section 6.1. The numerical results compared very well with the analytical criticality conditions for these simple reactor shapes.
- A steady-state eigenvalue comparison was made for the Dodds benchmark problem, in section 6.2.1. Here, both the calculated k-effective and flux profiles were shown to closely match the TINTE results for this benchmark. The k-effective also compared well with the reference benchmark value (39 pcm difference).
- A steady-state eigenvalue comparison was made for case 1 of the OECD PBMR benchmark in section 6.2.2. Here small k-effective differences (70 pcm) were seen between diffusionFoam and TINTE solutions.
- Short term dynamics comparisons were made by modeling a number of simple reactivity insertion transients based on the one-group bare sphere reactor model of section 6.1. Positive and negative step reactivity insertions of 100 and 200 pcm were considered. The initial power response of the reactor compared well with analytical solutions, which were based on the point jump approximation.
- The medium term reactor dynamics was tested for a simple load follow calculation. The two energy group model of section 6.2.2 was collapsed to a single group model. Based on this new model, a 100%-40%-100% load follow (case 4a of the OECD

PBMR benchmark) was calculated using diffusionFoam. The results were considered adequate in comparison with TINTE results for the same calculation.

Based on these tests, the diffusionFoam implementation has been shown to perform satisfactorily, although a number of issues do need to be addressed. The structure is currently in place for multi-group time-dependent solutions, but the fully time-dependent solution for multiple energy groups will require the resolution of several numerical issues. These same issues would need to be resolved for the implementation of a more advanced neutron transport solver.

Of particular importance would be the need for a block-point implicit solver, as was discussed in section 6.4.2. The fact that FOAM currently excludes such a coupled solver may initially seem to be a disadvantage, but one must consider that an efficient block solver would need to be created or sourced for any new implementation, regardless of the underlying framework. The lack of this functionality in FOAM should therefore not lead to the conclusion that an object-oriented multi-physics approach is not suited to reactor analysis applications. Rather, one should note that such functionality will need to be implemented as it would for any other framework. For this implementation, an object-oriented design provides a number of advantages. In particular, the implementation of new functionality in an object-oriented framework will have little to no impact on the already existing functionality. Additional features may be developed in parallel without the need to continuously ensure that the final code is synchronized, and that the new features are compatible with each other.

The potential for more advanced transport solutions was discussed in section 6.4.3. Here it was determined that such solutions are feasible, provided that equivalent finite-volume expressions for the spatial coupling can be derived. In section 6.4.4, the potential for implementing higher order spatial discretization schemes was discussed. This, again, is dependent on the derivation of an equivalent finite-volume representation for such schemes.

From the discussions and conclusions above, the objectives stated in section 1.1 have been met. It has been shown that a modern object-oriented multi-physics toolkit can effectively be applied to the solution of spatial reactor dynamics problems, and the potential exists for their application to other classes of reactor analysis problems.

## 7.1 *Future Work*

It is proposed that certain additional research be carried out to further investigate a number of topics.

- Existing block solvers used in reactor analysis, as well as those specialized block solvers already implemented in FOAM should be investigated further. The aim of such an investigation would be to fully block couple the diffusion equation solution in the current diffusionFoam implementation in a manner consistent with the existing structure of the toolkit.

- There is significant scope for the development of higher-order discretization schemes using the finite-volume approach. Developments in both time and spatial discretization schemes should be considered. Such schemes would provide advantages for any number of classes of engineering problems in addition to reactor analysis problems.

- A finite-volume implementation of the neutron transport equation, specifically using the discrete-ordinates method, would serve to illustrate the flexibility of the methodology. Such an illustration would further assist in breaking down the existing barriers between the reactor analysis classes of problems and other classes of engineering problems. Integral to this research would be the derivation of the equivalent streaming operator and required boundary conditions for the neutron transport equation using the finite-volume approach.

- The extension of the current diffusionFoam implementation to include the feedback effects of temperature and coolant density could potentially serve to illustrate the

primary advantages of utilizing multi-physics frameworks. Such an extension would require the close coupling of neutronic and thermal-hydraulic fields. As discussed in the introductory sections of this text, substantial research is currently directed towards the topic of close coupling in reactor analysis. Such a coupling would contribute valuable knowledge towards this topic.

- The storage, retrieval and processing of raw nuclear data, based on libraries such as the ENDF/B libraries, is a topic which requires substantial attention in the future. In particular, research into efficient and optimal storage, retrieval and processing of raw nuclear data using object-based data formats such as HDF5 and the FOAM file format is recommended. Such storage formats are in line with the current object-oriented approach to code development.

# 8. REFERENCES


ANL-7416 1977, "Benchmark Problem Book", ANL-7416 Suppl. 2, Argonne Code Centre, Argonne National Laboratory

Aboubacar, M. and Webster, M.F. 2000, "Development of an optimal hybrid finite volume/element method for viscoelastic flows", Int. J. Numer. Meth. Fluids, John Wiley & Sons

Ansys CFX 2007, "Ansys CFX", Ansys Inc., [Online]. Available from: <http://www.ansys.com/products/cfx.asp> [Accessed September 2007]

Ansys MP 2007, "Ansys Multiphysics 11.0", Ansys Inc., [Online]. Available from: <http://www.ansys.com/products/multiphysics.asp> [Accessed September 2007]

Bangyang, X. and Zhongsheng, X. 2006, "Flux expansion nodal method for solving multigroup neutron diffusion equations in hexagonal-z geometry", Annals of Nuclear Energy, Volume 33, Issue 4, pp. 370-376

Baranger, J., Maitre, J. and Oudin, F. 1996, "Connection between finite volume and mixed finite element methods", Mathematical Modelling and Numerical Analysis, Vol. 30, Issue 4, pp 445-465

Brainerd, W. S., Goldberg, C. H. and Adams, J. C. 1996, "Programmer's Guide to FORTRAN 90", Springer Verlag

Burns, G., Daoud, R. and Vaigl, J. 1994, "LAM: An Open Cluster Environment for MPI", Proceedings of the Supercomputing Symposium, pp. 379-386

CFD-ACE+ 2007, "CFD-ACE+", CFD Research Corporation, [Online]. Available from: <http://www.cfdrc.com/serv_prod/cfd_multiphysics/software/ace/> [Accessed September 2007]

CFD-FASTRAN 2007, "CFD-FASTRAN", CFD Research Corporation, [Online]. Available from: <http://www.cfdrc.com/serv_prod/cfd_multiphysics/software/fastran/> [Accessed September 2007]

Chadwick, M.B., Oblozinsky, P. and Herman, M. et al. 2006, "ENDF/B-VII.0: Next Generation Evaluated Nuclear Data Library for Nuclear Science and Technology", Nuclear Data Sheets, Volume 107, pp. 2931-3060

Chavent, G., Jaffré, J. and Roberts, J.E. 1997, "Generalized cell-centered finite volume methods: application to two phase flow in porous media", Computational Science for the 21st century, Tours (France), John Wiley & Sons

Ciarlet, P.G. 1978, "The finite element method for elliptic problems", North-Holland, Amsterdam

Clifford, I.D. 2007, "TINTE Nuclear Calculation Theory Description Report", IMAN 052322/A, PBMR (Pty.) Ltd.

Edenius, M. and Forssen, B. 1989, "CASMO-3, A Fuel Assembly Burnup Program, User's Manual", STUDSVIK/NFA-89/3, Studsvik Energiteknik AB

Ferziger, J.H. and Peric, M. 2001, "Computational Methods for Fluid Dynamics", 3rd Revision, ISBN 3540420746, Springer-Verlag

Filippone, S., Colajanni, M. and Pascucci, D. 1999, "An object-oriented environment for sparse parallel computation on adaptive grids", Proceedings of the 13th International Parallel Processing Symposium

Fletcher, C.A.J. 1990, "Computational Techniques for Fluid Dynamics", Volume I, Second Edition, Springer-Verlag

Fluent 2007, "Fluent 6.3", Fluent Inc., [Online]. Available from: <http://www.fluent.com/software/fluent/index.htm> [Accessed September 2007]

Gerwin, H. 1987, "The Two-Dimensional Reactor Dynamics Programme TINTE, Part 1: Basic Principles and Methods of Solution", Forschungzentrum (FZ) Juelich, ISSN 0366-0885

Gerwin, H., Scherer, W. and Teuchert, E. 1989, "The TINTE modular code system for computational simulation of transient processes in the primary circuit of a pebble-bed high-temperature gas-cooled reactor", Nuclear Science and Engineering, Vol. 103:3, pp. 302-312

Grossman, L.M. and Hennart, J. 2007, "Nodal diffusion methods for space-time neutron kinetics", Progress in Nuclear Energy, Vol. 49, Issue 3

HDF5 2007, "Introduction to HDF5 Release 1.6.6", The HDF Group, The National Center for Supercomputing Applications, [Online]. Available from: <http://hdf.ncsa.uiuc.edu/HDF5> [Accessed October 2007]

Hutt, P.K. and Knight, M.P. 1990, "The Development of a Transient Nodal Flux Solution in the PANTHER Code", Nuclear Electric (U.K.) report TD/FCB/MEM/3018

Ivanov, K.N. 2007, "Progress and challenges in the development and qualification of multi-Level multi-physics coupled methodologies for reactor analysis", International Congress on Advances in Nuclear Power Plants (ICAPP)

Jasak, H. 1996, "Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows", PhD Thesis, Imperial College, University of London

Jasak, H. 2006, "Multi-physics Simulations in Continuum Mechanics", 5th International Congress of Croatian Society of Mechanics

Jasak, H. 2007, "New Developments in OpenFOAM", Wikki Ltd. (United Kingdom), FSB, University of Zagreb (Croatia), [Online]. Available from: <http://powerlab.fsb.hr/ped/kturbo/OpenFOAM/slides/PolyMilano_14Feb2007.pdf> [Accessed November 2007]

Jin, Y.C. and Chang, H.K 1998, "Higher order polynomial expansion nodal method for hexagonal core neutronics analysis", Annals of Nuclear Energy, Volume 25, Issue 13, pp 1021-1031

Joo, H. G., Barber, D., Jiang, G. and Downar, T. J. 1998, "PARCS, A Multi-Dimensional Two-Group Reactor Kinetics Code Based on the Nonlinear Analytic Nodal Method", PU/NE-98-26, Purdue University

Kang, C.M. and Hansen, K.F. 1973, "Finite Element Methods for Reactor Analysis", Nucl. Sci. Engineering, 51

Kruger, J.H. 2004, "Object-oriented software development applied to adaptive resolution control in two-fluid models", North-West University

Lautard, J. 1994, "Minos : a nodal method; approximation by mixed dual finite elements in the Cronos code", CEA-N - 2763, Commissariat à l'énergie atomique

Lawrence, R.D. and Dorning, J.J 1979, "New Coarse-Mesh Diffusion and Transport Theory Methods for the Efficient Numerical Calculation of Multidimensional Reactor Power Distributions", in Proc. OECD/NEACRP Specialists' Mtg. Calculation of Three-Dimensional Rating Distributions in Operating Reactors, OECD

Lethbridge, P. 2004/2005, "Multi-physics Analysis", The Industrial Physicist, Dec 2004/Jan 2005

Lucas, D.S., Gougar, H.D., Roth, P.A, Wareing, T., Failla, G., McGhee, J. and Barnett, A. 2004, "Applications of the 3-D Deterministic Transport Attila® for Core Safety Analysis", Americas Nuclear Energy Symposium (ANES)

Meyer, B. 1988, "Object-oriented Software Construction", Prentice-Hall

Oden, J.T., Belytschko, T., Babuska, I. and Hughes, T.J.R. 2002, "Research directions in computational mechanics", Elsevier Science B.V.

OpenFOAM PG 2005, "OpenFOAM Programmer's Guide v1.2", OpenCFD Ltd., [Online]. Available from: <http://www.opencfd.co.uk/openfoam/index.html> [Accessed June 2007]

OpenFOAM UG 2005, "OpenFOAM User's Guide v1.2", OpenCFD Ltd., [Online]. Available from: <http://www.opencfd.co.uk/openfoam/index.html> [Accessed June 2007]

Ott, K.O. and Neuhold, R.J. 1985, "Introductory Nuclear Reactor Dynamics", American Nuclear Society

Peric, M. 1985, "A finite volume method for the prediction of three-dimensional fluid flow in complex ducts", University of London

Ragusa, J. 2006, "Overview of Reactor Core Neutron Transport Codes", Texas A&M DOE-NPRCAFC Meeting

Reitsma, F., Strydom, G., de Haas, J.B.M, Ivanov, K., Tyobeka, B., Mphahlele, R., Downar, T.J., Seker, V., Gougar, H.D., Da Cruz, D.F. and Sikik, U.E. 2004, "The PBMR steady-state and coupled kinetics core thermal-hydraulics benchmark test problems", Proceedings of the Conference on High Temperature Reactors, Beijing, China, September 22-24

Rumbaugh, J.R., Blaha, M.R., Lorensen, W., Eddy, F. and Premerlani, W. 1991, "Object-Oriented Modeling and Design", Prentice-Hall

Shober, R.A., Simms, R.C. and Henry, A.F. 1986, "The Nodal Methods for Solving Time Dependent Group Diffusion Equations", IIMAS-UNAM

Smith, K. 2003, "Reactor core methods", Mathematics and Computation (M&C) Topical Meeting

Stacey, W.M. 2001, "Nuclear Reactor Physics", John Wiley & Sons Inc.

Star-CD 2007, "Star-CD v4", CD-Adapco, [Online]. Available from: <http://www.cd-adapco.com/products/STAR-CD/index.html> [Accessed September 2007]

Strydom, G. 2004, "TINTE V&V: Modelling of the Dodds Neutronics Benchmark", T000120/A, PBMR Pty. Ltd.

Sutton, T.M. and Aviles, B.N. 1996, "Diffusion Theory Methods for Spatial Kinetics Calculations", Progress in Nuclear Energy, Vol. 30, No. 2, pp. 119-182

Turinsky, P.J., Al-Chalabi, R.M.K., Engrand, P., Sarsour, H.N., Faure, F.X. and Guo, W 1994, "NESTLE: A Few Group Neutron Diffusion Equation Solver Utilizing the Nodal Expansion Method (NEM) for Eigenvalue, Adjoint, and Fixed-Source Steady-State and Transient Problems", EGG_NRE-11406

U.S. National Committee on Theoretical and Applied Mechanics (NCTAM), Manufacturing Studies Board, Commission of Engineering and Technical Systems and National Research Council 1991, "Research directions in computational mechanics", National Academy of Sciences

Van Criekingen, S. 2007, "A 2-D/3-D cartesian geometry non-conforming spherical harmonic neutron transport solver", Annals of Nuclear Energy, Vol. 34, Issue 3, pp 177-187

Versteeg, H.K. and Malalasekera, W. 1995, "An introduction to computational fluid dynamics: The finite volume method", Second Edition, Prentice Hall

Wagner, M.R. 1979, "A Nodal Discrete Ordinates Method for the Numerical Solution of the Multidimensional Transport Equation", in Proc. Topl. Mtg. Comp. Meth. in Nucl. Eng., Williamsburg, VA, Vol. 2, American Nuclear Society

Waterman, P.J. 2004, "Moving up to multiphysics", Desktop Engineering Magazine, October 2004

Weller, H.G., Tabora, G., Jasak, H. and Fureby, C. 1998, "A Tensorial Approach to Computational Continuum Mechanics using Object-Oriented Techniques", American Institute of Physics