Subgrid Methods for Resolving Axial Heterogeneity in Planar Synthesis Solutions for the Boltzmann Transport Equation

by

Aaron M. Graham

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Nuclear Engineering and Radiological Sciences and Scientific Computing)
in the University of Michigan
2017

Doctoral Committee:

Adjunct Assistant Professor Benjamin S. Collins, Co-Chair
Professor Thomas J. Downar, Co-Chair
Assistant Professor Brian C. Kiedrowski
Professor William R. Martin
Assistant Professor Shravan Veerapaneni

For by Him all things were created, both in the heavens and on earth, visible and invisible, whether thrones or dominions or rulers or authorities – all things have been created through Him and for Him. He is before all things, and in Him all things hold together.

– Colossians 1:16-17

...all of this through the Creator, Lord and Savior Christ Jesus

Aaron M. Graham

aarograh@umich.edu

ORCID iD: 0000-0002-3245-8441

For my late father Marc

Thank you for your love and sacrifice for me for 25 years

Acknowledgments

Many people have beem instrumental both academically and personally to me during this process, so I would like to thank those who have helped me reach this goal.

First, I want to thank my co-advisors, Dr. Benjamin Collins and Prof. Thomas Downar. Your guidance and teaching these last four years have been invaluable. Working and learning under both of you has been an amazing opportunity for which I will always be grateful.

I would also like to thank the other members of my committe: Prof. Brian Kiedrowski, Prof. William Martin, and Prof. Shravan Veerapaneni. I appreciate your participation in this process and value the feedback I have received from each of you.

Next, I would like to thank the MPACT development team. Since joining the team, I have learned a great deal working with each of you and have been fortunate to be part of what has become a world-class project. I would like to thank the Department of Energy Office of Nuclear Energy for supporting this work under an Integrated University Program Graduate Fellowship, as well as members of the Consortium for Advanced Simulation of Light Water Reactors, who have supported parts of this work under U.S. Department of Energy contract number DE-AC05-00OR22725.

My officemates at both the University of Michigan and Oak Ridge National Laboratory (ORNL) have also been of great help throughout this process. I would like to express gratitude to all of you for help in the office and your friendship outside of it. I would also specifically like to thank Andrew Godfrey at ORNL for his assistance in generating reference solutions with KENO-VI.

Finally, I would like to thank my churches (in both Michigan and Tennessee) and my family for their support, encouragement, and prayers throughout these last four years. I want to express my great appreciation to my parents, who have guided and taught me not only throughout this process, but for the last 26 years. Thank you for everything; this would not have been possible without you. Most importantly, I want to thank my Lord and Savior Jesus Christ, who created this marvelous world and enabled me to study it in this way. All glory for these efforts goes to Him.

TABLE OF CONTENTS

Dedication . . . . **ii**

Acknowledgments . . . . **iii**

List of Figures . . . . **vii**

List of Tables . . . . **ix**

List of Appendices . . . . **x**

Abstract . . . . **xi**

Chapter 1 Introduction **1**

1.1 Motivation 2
1.2 Previous Work in 2D/1D 3
1.3 Dissertation Overview 4

Chapter 2 Neutron Transport Methodology **7**

2.1 Boltzmann Equation 7
2.1.1 Multigroup Approximation 10
2.1.2 Discrete Ordinates Approximation 11
2.1.3 Scattering Approximations 11
2.1.3.1 P_N Scattering 12
2.1.3.2 Transport Correction 12
2.1.4 Diffusion Approximation 13
2.2 Numerical Methods 15
2.2.1 Method of Characteristics 15
2.2.2 Coarse Mesh Finite Difference 18
2.2.3 Nodal Methods 21
2.2.3.1 Simplified Spherical Harmonics 21
2.2.3.2 Nodal Expansion Method 23
2.2.4 Collision Probabilities Method 24

Chapter 3 2D/1D Framework **26**

3.1 Background 26
3.2 Derivation 28
3.2.1 Radial Equations 28
3.2.2 Axial Equations 29
3.3 Implementation 30

3.3.1 3D Subplane CMFD 31
3.3.1.1 Homogenization 33
3.3.1.2 Coupling Coefficients 33
3.3.1.3 Projection 34
3.3.2 1D NEM-P_3 34
3.3.3 2D MOC. 35
3.3.3.1 Ray Tracing 35
3.3.3.2 Sweeping Algorithm 36
3.4 Parallel Decomposition 37
3.5 Sources of Error 40
3.5.1 Axial Homogenization 40
3.5.2 Axial Transverse Leakage Source 40
3.5.3 Radial Currents 41
3.5.4 Other Sources of Error 41

Chapter 4 Rod Cusping 43

4.1 Background 43
4.2 Decusping Methods History 45
4.2.1 Nodal Codes 45
4.2.1.1 Tabulation Methods 45
4.2.1.2 Polynomial Flux Expansions 46
4.2.1.3 Collector-Predictor Method 46
4.2.1.4 Bi-Linear Weighting Method 47
4.2.1.5 Nodal Expansion Method Modification 47
4.2.1.6 Equivalent-Node Method 48
4.2.1.7 Inverse Spectral Index Method 48
4.2.1.8 CIAMA Nodal Method 48
4.2.2 2D/1D Codes 49
4.2.2.1 Neighbor Spectral Index Method 49
4.2.2.2 Subplane Decusping 50
4.2.2.3 Approximate Flux Weighting Method 50
4.3 Summary 51

Chapter 5 Subgrid Decusping Methods 52

5.1 Polynomial Decusping 52
5.1.1 Correction Data Generation 52
5.1.2 Correction Application 54
5.2 Subplane Collision Probabilities 55
5.2.1 Axial Correction. 56
5.2.2 Radial Correction 56
5.2.3 MOC Correction 58
5.3 Subray Method of Characteristics 59
5.3.1 1D MOC Code and Problem Description 59
5.3.2 1D MOC Investigation 61
5.3.2.1 Specified Total Source 61

5.3.2.2 Fixed Fission Source 64
5.3.2.3 Summary 66
5.3.3 Subray MOC Description 66
5.3.4 MPACT Implementation 68
5.3.4.1 MOC Sweeper Modifications 68
5.3.4.2 Subregion and Recombination Selection 70
5.3.4.3 Subplane CMFD and P_3. 71
5.4 Summary 72

Chapter 6 Results 73

6.1 VERA Problem 4 74
6.1.1 Homogeneous Control Rods 75
6.1.2 Heterogeneous Control Rod 76
6.1.3 Differential Rod Worth Curve 77
6.1.4 KENO-VI Comparisons. 79
6.2 VERA Problem 5 80
6.3 C5G7 Results 81
6.3.1 1D Subray 81
6.3.2 2D C5G7 84
6.3.2.1 UO_2 Assembly 86
6.3.2.2 2D Core – Rodded Center Assembly. 90
6.3.2.3 2D Core – Rodded Corner Assembly 92
6.3.3 3D C5G7 94
6.3.3.1 3D C5G7 UO_2 Assembly 94
6.3.3.2 3D Core – Rodded Center Assembly. 96
6.3.3.3 3D Core – Rodded Corner Assembly 97

Chapter 7 Conclusions 100

7.1 Summary 100
7.2 Future Work 102
7.2.1 Subgrid Method Applications 102
7.2.2 Polynomial Decusping Improvements 103
7.2.3 Subplane Collision Probabilities 103
7.2.4 Subray MOC 104

Appendices 105

Bibliography 147

LIST OF FIGURES

1.1 Illustration of reactor geometry and mesh 1

2.1 Illustration of the multigroup discretization scheme 10

3.1 2D/1D Illustration 27
3.2 Calculation flow for 2D/1D scheme 31
3.3 Calculation flow for 3D subplane CMFD 32
3.4 Calculation flow for 1D axial calculations in MPACT 34
3.5 Modular Ray Tracing 35
3.6 Calculation flow for 2D MOC calculation in MPACT 37
3.7 Calculation flow for 2D MOC sweep of a single plane 38

4.1 Rod Cusping Illustration 43
4.2 Rod Cusping in MPACT 44

5.1 Polynomial decusping example data generation 53
5.2 Calculation flow for 2D/1D with subplane collision probabilities 55
5.3 Subplane Collision Probabilities Illustration 57
5.4 Illustration of 1D MOC 3x17 Geometry 61
5.5 1D MOC Group 7 Scalar Flux for Fixed Total Source 62
5.6 Angular flux comparisons for a fixed fission and scattering source calculation . 63
5.7 Group 7 angular flux comparisons for 25% and 75% mixtures 64
5.8 Group 1 angular flux comparisons for 25% and 75% mixtures 65
5.9 Subray MOC Illustration 67
5.10 Calculation flow for 2D MOC sweep of a single plane with subray 69
5.11 Pin cells in 17×17 assembly using subray MOC for recombination flags of 0 (red), 1 (blue), 2 (green), and 3 (white). 70

6.1 VERA Problem 4 radial (left) and axial (right) layouts 74
6.2 Differential rod worth curves for MPACT decusping techniques. 78
6.3 VERA Problem 5 radial geometry (left) and rod bank positions (right) 80
6.4 Illustration of 1D MOC 10 Pin Geometry 82
6.5 Eigenvalue comparisons for 1D rod withdrawal 82
6.6 Scalar flux distributions for 1D subray MOC calculations 83
6.7 Descriptiong of C5G7 Core 85
6.8 Description of C5G7 Assembly 86

A.1 Cylindrical Geometry Collision Probabilities 111
A.2 Cylindrical Geometry Collision Probabilities Self-Transport 112

LIST OF TABLES

6.1 VERA Problem 4 Homogeneous Rod Results 75
6.2 VERA Problem 4 Heterogeneous Rod Results 77
6.3 Average Differences between MPACT and KENO-VI for VERA Problem 4 . . 79
6.4 VERA Problem 5 Decusping Results 81
6.5 2D C5G7 UO_2 Assembly Recombination Sensitivity 87
6.6 2D C5G7 UO_2 Assembly Results 87
6.7 Long Ray Data for 2D C5G7 UO_2 Rodded Assembly 89
6.8 Average Performance and Convergence Data for C5G7 Rodded UO_2 Assembly 89
6.9 2D C5G7 Core Recombination Sensitivity, Center Assembly 90
6.10 2D C5G7 Core Results, Center Assembly 90
6.11 Long Ray Data for C5G7 Core 91
6.12 2D C5G7 Core Performance Data, Center Assembly 91
6.13 2D C5G7 Core Recombination Sensitivity, Corner Assembly 92
6.14 2D C5G7 Core Results, Corner Assembly 92
6.15 2D C5G7 Core Performance Data, Corner Assembly 93
6.16 3D C5G7 UO_2 Assembly Recombination Sensitivity 94
6.17 3D C5G7 UO_2 Assembly Performance 95
6.18 3D C5G7 Core with Rodded Center Assembly Results 96
6.19 3D C5G7 Core with Rodded Center Assembly Performance 97
6.20 3D C5G7 Core with Rodded Corner Assembly 98
6.21 3D C5G7 Core with Rodded Corner Assembly Performance 99

A.1 Four-Point Gaussian Quadrature 115

LIST OF APPENDICES

Chapter A Supplemental Derivations 105

Chapter B C5G7 Cross-Sections 116

Chapter C 1D MOC Code 119

ABSTRACT

As computing power has increased in recent years, there has been a strong interest in using direct whole-core transport calculations for reactor analysis to improve on the traditional nodal methods used by industry. Because 3D transport is still prohibitively expensive for practical calculations, a new class of planar synthesis methods has been developed. These methods decompose the reactor into a stack of 2D planes, performing high fidelity transport on each 2D plane before using a lower order method to couple the planes. These methods are faster than 3D transport and more accurate than nodal methods. Despite the more palatable runtimes compared with 3D transport, these methods are still much slower than nodal methods. Thus, reducing computational expense is still important to make these methods useful.

One means of accomplishing this is to reduce the number of 2D planes used in the transport calculations, since these are the most expensive parts of the calculations. Doing this improves the efficiency of the calculation, but also increases the likelihood of subgrid heterogeneity within a 2D plane, introducing error into the solution. To eliminate this error while maintaining the efficiency improvements from the coarse axial mesh, a subgrid method must be employed to resolve the subgrid information.

In this work, three new subgrid methods are presented in the context of the 2D/1D method. The polynomial method uses pregenerated sixth-order polynomials to correct the homogenization of the subgrid heterogeneity. The subplane collision probabilities (CP) method uses 1D CP calculations to capture the radial effects of the subgrid heterogeneity, then uses subplane coarse mesh finite difference (CMFD) and 1D spherical harmonics (P_N) to capture the axial effects of the heterogeneity. The axial flux profiles produced by these calculations are then used to generate improved homogenized cross sections for the 2D MOC calculations. Finally, the subray method of characteristics (MOC) performs a modified transport sweep of each of the 2D planes to directly resolve the subgrid flux shape during the transport calculations while making use of the subplane scheme to improve the CMFD and PN calculations.

Each of these methods was applied to the rod cusping problem, the most severe type of axial heterogeneity in planar synthesis methods. The polynomial methods reduced the errors in 3D power distribution from greater than 25% to around 10% for most problems. The subplane CP method performed much better, reducing the errors in power distribution to less than 5% for most problems. Finally, subray MOC reduced the errors to less than 2% for all problems it was tested on.

CHAPTER 1

Introduction

Predicting the neutron flux distribution is of primary importance in reactor analysis. The power distribution in the reactor is directly related to the neutron flux and directly impacts the economics and safety of the reactor. Economically, accurate prediction of the power distribution enables core designers to determine the optimal schemes for fuel loading and shuffling to make efficient use of the fuel and minimize the likelihood of fuel failures. The power distribution is also important to both steady-state and transient operation of the reactor, especially during severe accident scenarios. Because of these requirements, the codes used in reactor analysis must be capable of providing highly accurate, detailed information concerning the neutron flux distribution.

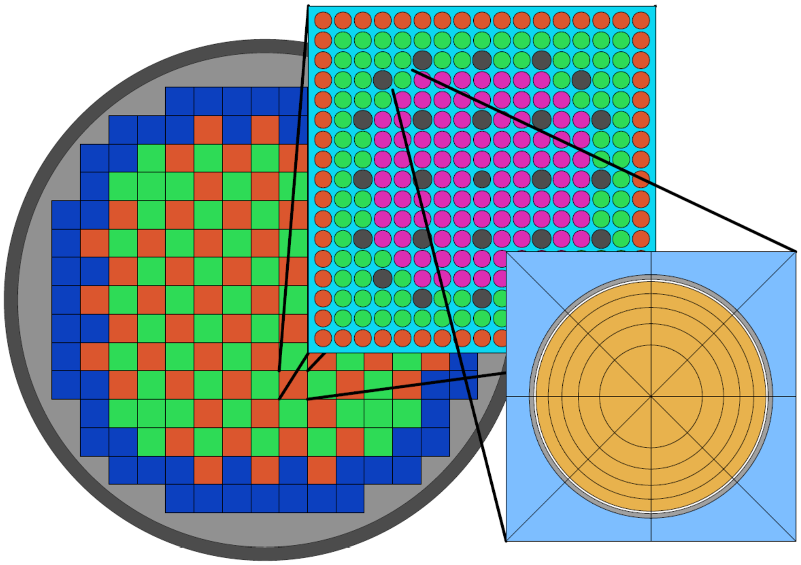


Figure 1.1: Illustration of reactor geometry and mesh

Figure 1.1 shows the geometry of a reactor. The full core is shown on the left as an array of fuel assemblies; a fuel assembly is shown in the center as an array of fuel pins; a meshed fuel pin is shown on the right. Reactor analysis has historically used a two-step approach to model reactors. The first step consists of lattice calculations. These calculations are typically 2D transport calculations done on a single fuel assembly using explicitly meshed pins to obtain a local shape for the flux distribution in that assembly. This local distribution is used to homogenize cross sections in larger nodes, usually a quarter assembly. This coarser mesh is then used in a nodal diffusion calculation to generate the global neutron flux distribution for the entire reactor. The global diffusion solution and the local transport solutions are then combined to reconstruct a global fine mesh solution that can be used for the necessary analysis.

While these methods have been sufficient for reactor design and operation to this point, the homogenization and reconstruction steps inherently introduce some uncertainty in the accuracy of the solution and are known to be inaccurate for some situations. This uncertainty forces analysts to be conservative to ensure an acceptable power distribution. This conservatism combined with a rapid increase in computing power in recent years has driven a push toward direct whole-core transport calculations. These solutions require more computing resources to obtain, but are able to correctly model all the physics in the reactor in a single calculation and provide greater confidence in the solutions.

There are two categories of methods that can be used for transport calculations: Monte Carlo and deterministic. Monte Carlo calculations use random sampling of continuous cross section data to determine the average behavior of neutrons throughout the reactor core. Given enough samples, a Monte Carlo calculation will give the exact solution. However, the number of particles required to obtain a full core power distribution with acceptably low statistical uncertainty is prohibitive even with today's computing resources. Deterministic calculations do not have to deal with statistical uncertainty and can provide solutions much more quickly than Monte Carlo calculations because they solve the equation directly instead of using random sampling. However, deterministic methods are limited by certain approximations and discretizations that adversely affect the accuracy of the solution. Making improvements to these approximations is crucial to improving deterministic transport calculations and making them viable for practical reactor analysis.

1.1 Motivation

One category of whole-core deterministic transport methods that has gained popularity is planar synthesis methods. Two types of planar synthesis methods under active research today are 2D/1D and 2D/3D. In both of these methods, the reactor is modeled as a stack of 2D planes, which are then coupled through a second calculation. Each 2D plane is solved using a high-fidelity transport method such as the method of characteristics (MOC) [1]. In 2D/3D, these calculations are used to generate homogenized cross sections for a 3D discrete ordinates (S_N) calculation. Because the MOC calculations are performed across the entire 2D plane, the homogenized cross sections are more accurate than any that can be produced by the traditional 2-step method. If 2D/1D is used, the stack of 2D MOC solutions is coupled axially through a lower order diffusion or transport calculation in the axial direction. This takes advantage of the fact that in many reactors, most heterogeneity is in the radial direction. The solution changes shape much more slowly in the axial direction, so the same resolution is not needed as in the radial direction.

These planar synthesis methods are orders of magnitude faster than a full core Monte Carlo calculation, but they are still much slower than traditional nodal methods used by industry. To rectify this, it is important to improve the efficiency of these calculations to make them more useful for practical calculations. Some of these efficiency gains can be realized by modifications to the methods and algorithms used in the codes. However, a simpler way to reduce the runtime fo these calculations is to simply use a coarser mesh. Because the most expensive part of the calculation is the 2D planar transport sweep, reducing the number of 2D planes can significantly reduce the computing resources required for the 2D/1D or 2D/3D calculations.

When reducing the number of 2D planes, care must be taken to ensure that no significant approximations are made. These methods implicitly assume that there is no axial change in materials for a given 2D plane. If this assumption is violated, these materials will be homogenized and introduce error in the solution. For some models, explicitly meshing all axial components results in a relatively fine mesh, preventing any runtime reduction by coarsening the mesh. To circumvent this issue, subgrid methods must be developed which are able to account for axial heterogeneity within a plane and preserve the accuracy of transport calculations without requiring a fine axial mesh. The work presented here focuses on the development of such methods.

1.2 Previous Work in 2D/1D

Many different axial heterogeneities can be thought of, but by far the most troublesome in most reactors is control rods. Control rods move axially throughout reactor operation to control the power shape in the core, and they can move rapidly at times to maintain safe conditions during a transient scenario. Furthermore, they tend to be strong neutron absorbers with relatively large cross sections. Attempting to explicitly mesh every control rod position can result in an unusably fine axial mesh, but homogenizing the control rods axially within a plane results in unacceptably large errors in the flux distribution called rod cusping. The application of subgrid methods developed in this work will be focused on the problem of rod cusping since it is the most severe axial heterogeneity for planar synthesis methods.

Rod cusping was first identified in [2] in 1977 in the context of nodal methods. In generating homogenized cross sections for the nodal calculation, it was discovered that simply having a rodded and unrodded version of the node cross section and interpolating them was insufficient and produced cusping errors. Since then, a myriad of researchers have endeavored to resolve this issue [3–24]. The rise of whole-core transport methods, including planar synthesis methods, addresses many shortcomings of the traditional reactor analysis methods, but not rod cusping.

Methods to deal with control rod cusping up to this point have generally fallen into one of two categories: fast or accurate. The fastest methods usually have pregenerated data from a separate set of calculations which can be used to quickly adjust cross sections for partially rodded nodes and improve the answers. The most accurate methods reduce the rod cusping errors to acceptable levels through changes in the global iteration scheme, but enough additional calculation is required to achieve this accuracy that the methods are not much more beneficial than simply refining the axial mesh to eliminate the cusping altogether. While some methods have bridged this gap between speed and accuracy better than others, none have succeeded at eliminating cusping altogether. This work seeks to fundamentally resolve the control rods in the transport calculations in such a way that rod cusping errors are greatly reduced or eliminated without a significant increase in computational burden.

1.3 Dissertation Overview

This dissertation is organized into 5 main parts:

1. Approximations and numerical methods used to solve the Boltzmann transport equation,
2. An in-depth description of the 2D/1D implementation used for this work,
3. A detailed look at rod cusping and the history of decusping methods research,
4. Descriptions of methods developed for this work, and
5. Results and analysis of the newly developed methods.

The first part will introduce the Boltzmann transport equation in its most general form. The equation is continuous in space, energy, angle and time, so discretizations are required to solve the equation deterministically. After discussing some of the common discretization schemes, some important numerical methods relevant to this work are discussed. These include MOC, coarse mesh finite difference (CMFD), the spherical harmonics (P_N) approximation, the Nodal Expansion Method (NEM), and the collision probabilities (CP) method. Detailed derivations are presented for those methods which will be modified as part of the new subgrid methods; other derivations are deferred to the appendices or external references.

After presenting the basic numerical methods, the following chapter will provide an in-depth look at the 2D/1D implementation in the MPACT code used for this research. Some history of the method will be described first, followed by a derivation of the radial and axial equations used as a basis for the 2D/1D method. The numerical methods and iteration scheme used by MPACT to solve these equations will then be described in detail.

With the foundation of 2D/1D laid, the third part will provide a more thorough discussion of rod cusping. First, the cause and effects of cusping errors will be described for both nodal and 2D/1D codes. Second, a history of decusping methods will be discussed. These methods will be grouped based on whether they were developed for nodal codes or 2D/1D codes. This discussion will be used to more thoroughly motivate the need for advanced multigrid methods to address this problem.

Chapter 5 will describe the three new methods that have come out of this work. These methods are described in increasing accuracy, complexity, and generality. The first is polynomial decusping. This method relies on pregenerated data to correct the cross sections in a partially rodded node prior to the beginning of the 2D/1D calculations. This was developed primarily out of immediate need to address the rod cusping issue in MPACT. The second method is called subplane collision probabilities. This method modifies the axial part of the 2D/1D calculation to use a refined grid, then uses the CP method for the rodded and unrodded regions to generate improved flux shapes and cross sections. This method is more general in its ability to handle multiple control rod types and other simple heterogeneities. The final method presented is a new version of MOC called subray MOC. This method seeks to address axial heterogeneity during the 2D MOC sweeps, which should improve the solution at the most fundamental level instead of attempting to improve homogenized cross sections.

The final section will present results and analysis of the methods. First, two problems will be presented to test the polynomial and subplane collision probabilities. Both accuracy and runtime will be compared to reference cases with no rod cusping errors. Second, results and analysis of subray MOC will be presented using another set of problems. Accuracy, convergence, and runtime will be analyzed compared to reference cases. Comparisons of subray MOC with the polynomial and subplane collision probabilities methods will also be made to show improvements in accuracy achieved by subray MOC.

The dissertation will close with a summary and brief discussion of future research topics related to the methods presented here.

CHAPTER 2

Neutron Transport Methodology

This chapter is dedicated to introducing the neutron transport equation and numerical methods commonly used to solve it. First, the transport equation will be introduced and all its terms defined. Some common approximations that are needed to make the transport equation more practically solvable will also be introduced. Second, numerical methods will be discussed and, when appropriate, derived from the transport equation. The theory is covered in detail in a variety of textbooks [25], and other texts discuss numerical methods to solve the transport equation [26]. This chapter will not attempt to reiterate all that can be found in these references, but rather to highlight the theory and methods which are relevant to this work.

2.1 Boltzmann Equation

The time-dependent Boltzmann equation for neutron transport is shown below:

$$\begin{aligned}
&\frac{1}{v}\frac{\partial\varphi}{\partial t} + \boldsymbol{\Omega}\cdot\boldsymbol{\nabla}\varphi + \Sigma_t(\boldsymbol{x},E,t)\varphi(\boldsymbol{x},E,\boldsymbol{\Omega},t) \\
&\quad = \frac{1}{4\pi}\int_0^\infty\int_{4\pi}\Sigma_s(\boldsymbol{x},E'\to E,\boldsymbol{\Omega}'\to\boldsymbol{\Omega})\varphi(\boldsymbol{x},E',\boldsymbol{\Omega}')\,d\Omega' dE' \\
&\qquad + \frac{\chi_p(\boldsymbol{x},E)}{4\pi}\int_0^\infty\int_{4\pi}(1-\beta(\boldsymbol{x},E'))\nu\Sigma_f(\boldsymbol{x},E',t)\varphi(\boldsymbol{x},E',\boldsymbol{\Omega}',t)\,d\Omega' dE' \\
&\qquad + \sum_{j=1}^{N_d}\frac{\chi_{d,j}(\boldsymbol{x},E)}{4\pi}\lambda_j C_j(\boldsymbol{x},t) + Q(\boldsymbol{x},E,\boldsymbol{\Omega},t)\ ,
\end{aligned} \tag{2.1a}$$

$$\varphi(\boldsymbol{x}_b,E,\boldsymbol{\Omega},t) = \varphi^b(\boldsymbol{x}_b,E,\boldsymbol{\Omega},t)\ , \quad \boldsymbol{\Omega}\cdot\boldsymbol{n} < 0\ . \tag{2.1b}$$

Before addressing the methods used to solve Equation 2.1, we will briefly define each of the terms in the Equation 2.2. The first term is the time derivative term, shown in , which accounts for the change in the angular flux over time in dV about $\boldsymbol{x}$, dE about E, and dΩ about $\boldsymbol{\Omega}$.

$$\frac{1}{v}\frac{\partial\varphi}{\partial t} \; . \tag{2.2}$$

The second term (Equation 2.3 is the streaming term. This describes neutrons with energy E traveling out of the volume dV in the direction $\boldsymbol{\Omega}$.

$$\boldsymbol{\Omega}\cdot\boldsymbol{\nabla}\varphi \; . \tag{2.3}$$

The third term is the total reaction rate (Equation 2.4). This describes the total number of collisions experienced in dV by neutrons with energy E and direction $\boldsymbol{\Omega}$. Equations 2.2-2.4 together make up the total loss of neutrons.

$$\Sigma_t(\boldsymbol{x},E)\varphi(\boldsymbol{x},E,\boldsymbol{\Omega}) \; . \tag{2.4}$$

Equation 2.5 shows the scattering source written in a simplified form. This is the total number of neutrons scattering into energy E and direction $\boldsymbol{\Omega}$ from all other energies and directions E' and $\boldsymbol{\Omega}'$ in dV. Because scattering is symmetric around the incident angle, the scattering cross section depends only on the dot product $\boldsymbol{\Omega}'\cdot\boldsymbol{\Omega}$ rather than each of the two angles independently.

$$\int_0^\infty\int_{4\pi}\Sigma_s(\boldsymbol{x},E'\to E,\boldsymbol{\Omega}'\cdot\boldsymbol{\Omega})\varphi(\boldsymbol{x},E',\boldsymbol{\Omega}')d\Omega' dE' \; . \tag{2.5}$$

Equation 2.6 shows the prompt fission source, neutrons entering dV with energy and direction E and $\boldsymbol{\Omega}$ directly from a fission event. Fission is an isotropic process, so the total fission source is calculated then distributed evenly across 4π. Furthermore, the energy distribution of fission is practically independent of incident neutron energy, so the fission neutron distribution $\chi_p(E)$ can be outside the integral over energy. A small fraction of fission neutrons are considered “delayed,” meaning they are emitted by the radioactive decay of a fission product. The prompt fission source must be adjusted by the factor $(1-\beta(\boldsymbol{x},E'))$ to account for this. Typically β is less than 1% and different for each fissionable isotope.

$$\frac{\chi_p(\boldsymbol{x},E)}{4\pi}\int_0^\infty\int_{4\pi}(1-\beta(\boldsymbol{x},E'))\nu\Sigma_f(\boldsymbol{x},E',t)\varphi(\boldsymbol{x},E',\boldsymbol{\Omega}',t)d\Omega' dE' \; . \tag{2.6}$$

Equation 2.7 shows the fission source due to delayed neutrons. The precursors, fission products which emit delayed neutrons, are divided into N_d groups based on the magnitude of their decay constant λ_j. Like the prompt neutrons, delayed neutrons are isotropic and distributed in energy with some distribution $\chi_{d,j}(\boldsymbol{x},E)$ based on which precursors are produced.

$$\sum_{j=1}^{N_d} \frac{\chi_{d,j}(\boldsymbol{x},E)}{4\pi}\lambda_j C_j(\boldsymbol{x},t) \ . \tag{2.7}$$

Equation 2.8 shows external source term. This term accounts for all neutrons entering dV with energy and direction E and $\boldsymbol{\Omega}$ from sources other than scatter and fission. Equation 2.8 shows the source term as a function of angle, but it is often considered to be isotropic like the fission source.

$$Q(\boldsymbol{x},E,\boldsymbol{\Omega}) \ . \tag{2.8}$$

Finally, Equation 2.1b shows the boundary condition for the transport equation. The dot product $\boldsymbol{\Omega}\cdot\boldsymbol{n}$ is between the direction of flight and the outward normal vector on the boundary of the problem. The angular flux boundary condition defines the angular flux entering the problem domain. for the entire surface $\boldsymbol{x}_b$, all energies, and all times.

Many problems of interest are steady-state, allowing Equation 2.1 to be simplified significantly. The time derivative becomes 0, eliminating the term in 2.2, and the precursor concentrations are unchanging in time, allowing the fission source terms in 2.6 and 2.7 to be lumped into a single term. To ensure balance between the loss and source terms without a time derivative, the equation is reformulated as an eigenvalue equation. The fission source is multiplied by the eigenvalue $\lambda = \frac{1}{k_{eff}}$, allowing the equation to be balanced. The cross sections can then be adjusted until $\lambda = 1$ is achieved, providing a physically meaningful steady-state solution to the equation. The steady-state form of the Boltzmann equation is shown below:

$$\begin{aligned}\boldsymbol{\Omega}\cdot\boldsymbol{\nabla}\varphi + \Sigma_t(\boldsymbol{x},E)\varphi(\boldsymbol{x},E,\boldsymbol{\Omega}) \ &= \frac{1}{4\pi}\int_0^\infty\int_{4\pi}\Sigma_s(\boldsymbol{x},E'\to E,\boldsymbol{\Omega}'\to\boldsymbol{\Omega})\varphi(\boldsymbol{x},E',\boldsymbol{\Omega}')d\Omega' dE' \\ &+ \frac{1}{k_{eff}}\frac{\chi(E)}{4\pi}\int_0^\infty\int_{4\pi}\nu\Sigma_f(\boldsymbol{x},E')\varphi(\boldsymbol{x},E',\boldsymbol{\Omega}')d\Omega' dE' \ ,\end{aligned} \tag{2.9a}$$

$$\varphi(\boldsymbol{x}_b,E,\boldsymbol{\Omega}) = \varphi^b(\boldsymbol{x}_b,E,\boldsymbol{\Omega}) \ , \quad \boldsymbol{\Omega}\cdot\boldsymbol{n} < 0 \ . \tag{2.9b}$$

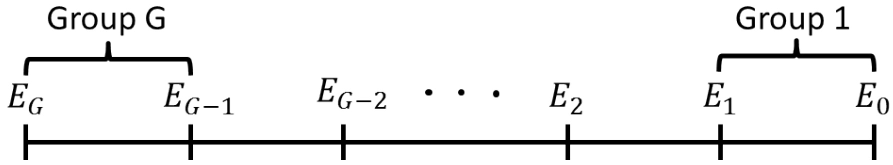


Figure 2.1: Illustration of the multigroup discretization scheme

2.1.1 Multigroup Approximation

One important approximation that is commonly made to the transport equation is the multigroup approximation. To make this approximation, an appropriate energy range for the problem of interest is selected. This energy range is divided up into N energy groups, with each group going from E_g up to E_{g-1}. For light-water reactor problems, it is common to select 0 eV for E_N and 20 MeV for E_0.

We first define the multi-group flux, cross sections, and chi distribution in Equations 2.10. These definitions ensure that the total reaction rates are preserved in each energy interval g.

$$\varphi_g(\boldsymbol{x},\boldsymbol{\Omega}) = \int\limits_{E_n}^{E_{n-1}} \varphi(\boldsymbol{x},E,\boldsymbol{\Omega})\,dE\ , \tag{2.10a}$$

$$\begin{aligned}\Sigma_{x,g}(\boldsymbol{x},\boldsymbol{\Omega})\varphi_g(\boldsymbol{x},\boldsymbol{\Omega}) &= \int\limits_{E_n}^{E_{n-1}} \varphi(\boldsymbol{x},E,\boldsymbol{\Omega})\Sigma_x(\boldsymbol{x},E,\boldsymbol{\Omega})\,dE \\ \Rightarrow \Sigma_x,g(\boldsymbol{x},\boldsymbol{\Omega}) &= \frac{\int_{E_n}^{E_{n-1}} \varphi(\boldsymbol{x},E,\boldsymbol{\Omega})\Sigma_x(\boldsymbol{x},E,\boldsymbol{\Omega})\,dE}{\int_{E_n}^{E_{n-1}} \varphi(\boldsymbol{x},E,\boldsymbol{\Omega})\,dE}\ .\end{aligned} \tag{2.10b}$$

Using the definitions above, Equation 2.9 can be operated on by $\int_{E_n}^{E_{n-1}}(\cdot)\,dE$ to obtain the multi-group transport Equation 2.11.

$$\begin{aligned}\boldsymbol{\Omega}\cdot\boldsymbol{\nabla}\varphi_g + \Sigma_{t,g}(\boldsymbol{x})\varphi_g(\boldsymbol{x},\boldsymbol{\Omega}) &= \frac{1}{4\pi}\sum_{g'=1}^{G}\int\limits_{4\pi}\Sigma_{s,g'\to g}(\boldsymbol{x},\boldsymbol{\Omega}'\to\boldsymbol{\Omega})\varphi_{g'}(\boldsymbol{x},\boldsymbol{\Omega}')\,d\Omega' \\ &+ \frac{1}{k_{eff}}\frac{\chi_g}{4\pi}\sum_{g'=1}^{G}\int\limits_{4\pi}\nu\Sigma_{f,g'}(\boldsymbol{x})\varphi_{g'}(\boldsymbol{x},\boldsymbol{\Omega}')\,d\Omega'\ ,\end{aligned} \tag{2.11a}$$

$$\varphi_g(\boldsymbol{x}_b,\boldsymbol{\Omega}) = \int\limits_{E_n}^{E_{n-1}} \varphi^b(\boldsymbol{x}_b,E,\boldsymbol{\Omega})\,dE\ , \quad \boldsymbol{\Omega}\cdot\boldsymbol{n} < 0\ . \tag{2.11b}$$

2.1.2 Discrete Ordinates Approximation

In addition to energy discretization, it is also useful to discretize the transport equation in angle. The angular variable $\boldsymbol{\Omega}$ is made up of a polar angle (μ) and an azimuthal angle (α), where the polar angle is defined with respect to the z-axis and the azimuthal angle is defined with respect to the x-axis in the x-y plane. Both angles can be discretized as follows:

$$\boldsymbol{\Omega} = \cos(\alpha)\sqrt{1-\mu^2}\boldsymbol{i} + \sin(\alpha)\sqrt{1-\mu^2}\boldsymbol{j} + \mu\boldsymbol{k} \tag{2.12a}$$

$$\Rightarrow \boldsymbol{\Omega}_n = \cos(\alpha_n)\sqrt{1-\mu_n^2}\boldsymbol{i} + \sin(\alpha_n)\sqrt{1-\mu_n^2}\boldsymbol{j} + \mu_n\boldsymbol{k}\ . \tag{2.12b}$$

For each $\boldsymbol{\Omega}_n$, there is an associated weight w_n. These weights and angles together make up an angular quadrature set which simplifies the integrals in Equations 2.13.

$$\int d\Omega = \sum_{n=1}^{N} w_n = 4\pi\ , \tag{2.13a}$$

$$\int \boldsymbol{\Omega} d\Omega = \sum_{n=1}^{N} \boldsymbol{\Omega}_n w_n = 0\ , \tag{2.13b}$$

$$\int_{4\pi} f(\boldsymbol{\Omega})\, d\Omega \approx \sum_{n=1}^{N} f_n w_n\ . \tag{2.13c}$$

Applying this discretization to the multi-group transport Equation 2.11, we obtain the following discrete ordinates (S$_N$) equations:

$$\begin{aligned}\boldsymbol{\Omega}_n \cdot \boldsymbol{\nabla}\varphi_{g,n} + \Sigma_{t,g}(\boldsymbol{x})\varphi_{g,n}(\boldsymbol{x}) = {} & \frac{1}{4\pi}\sum_{g'=1}^{G}\sum_{n'=1}^{N}\Sigma_{g'\to g,n'\to n}(\boldsymbol{x})\varphi_{g',n'}(\boldsymbol{x})w_{n'} \\ & + \frac{1}{k_{eff}}\frac{\chi_g}{4\pi}\sum_{g'=1}^{G}\sum_{n'=1}^{N}\nu\Sigma_{f,g'}(\boldsymbol{x})\varphi_{g',n'}(\boldsymbol{x})w_{n'}\ ,\end{aligned} \tag{2.14a}$$

$$\varphi_{g,n}(\boldsymbol{x}_b) = \varphi_g^b(\boldsymbol{x}_b, \boldsymbol{\Omega}_n)\ , \quad \boldsymbol{\Omega}_n \cdot \boldsymbol{n} < 0\ . \tag{2.14b}$$

2.1.3 Scattering Approximations

One of the biggest challenges to solving the transport equation is angle dependence of the scattering cross sections and angular flux. To simplify the scattering cross sections, there are two different types of approximations that can be made in MPACT.

2.1.3.1 P$_N$ Scattering

The first scattering approximation that can be made is P$_N$ scattering. To make this approximation, the $\mathbf{\Omega}' \cdot \mathbf{\Omega}$ is rewritten as a single angular variable μ_s, the cosine between the incoming and outgoing scattering angles. The scattering cross section can then be expanded in terms of Legendre polynomials, defined by Equations 2.15.

$$P_{n+1}(\mu_s) = \frac{(2n+1)\mu_s P_n(\boldsymbol{\mu}_s) - nP_{n-1}(\mu_s)}{n+1} \ , \tag{2.15a}$$

$$P_0(\mu_s) = 1 \ , \tag{2.15b}$$

$$P_1(\mu_s) = \mu_s \ , \tag{2.15c}$$

$$\int\limits_{-1}^{1} P_n(\mu_s) P_m(\mu_s) d\mu_s = \frac{2}{2n+1}\delta_{n,m} \ . \tag{2.15d}$$

Equations 2.16 show the expansion of the scattering cross section using Legendre polynomials. Using more terms in the expansion improves the accuracy. For most reactor problems, $N \leq 3$ is sufficient. Problems such as shielding and others may require many more terms to be kept to obtain sufficient accuracy.

$$\Sigma_s(\boldsymbol{x}, \mu_s) = \sum_{n=0}^{N} \frac{2n+1}{4\pi} P_n(\mu_s) \Sigma_{sn}(\boldsymbol{x}) \ , \tag{2.16a}$$

$$\Sigma_{s,n}(\boldsymbol{x}) = 2\pi \int\limits_{-1}^{1} \Sigma_s(\boldsymbol{x}, \mu_s) P_n(\mu_s) d\mu_s \ . \tag{2.16b}$$

2.1.3.2 Transport Correction

A second simplification of the scattering source that can be used is transport-corrected isotropic (TCP$_0$) scattering. When using this approximation, the equation is solved using only the zeroth order term in 2.16. The cross section data used to develop the multi-group scattering cross sections is modified beforehand to still preserve some of the higher order scattering physics.

Currently, the MPACT code uses [27] uses a combination of three transport-correction methods for its TCP$_0$ scattering treatments [28, 29]. The Neutron Leakage Conservation (NLC) method [30] is used for H-1; the nTRACER in-scatter method [31] is used for B-11, C-12, and O-16; and the out-scatter method [32] is used for all remaining isotopes. The out-scatter approximation is defined as

$$\Sigma_{s0,g\to g} = \Sigma_{s0,g\to g} - \sum_{g'=1}^{G} \Sigma_{s1,g\to g'} \ . \tag{2.17}$$

The in-scatter approximation is defined as

$$\Sigma_{s0,g\to g} = \Sigma_{s0,g\to g} - \frac{1}{\phi_{1,g}} \sum_{g'=1}^{G} \Sigma_{s1,g'\to g}\phi_{1,g'} \ , \tag{2.18}$$

where $\phi_{1,g}$ is the first moment of the angular flux. Finally, the NLC correction is defined as follows:

$$\Sigma_{s0,g\to g} = \Sigma_{s0,g\to g} + \frac{1}{3D_g} - \Sigma_{t,g} \ , \tag{2.19}$$

where D_g is a diffusion coefficient determined from leakage calculations in a 1D fixed-source transport calculation. For each of these corrections, the same modification made to the self-scatter cross sections in the above equations is also made to the total cross section $\Sigma_{t,g}$ to obtain the transport cross section $\Sigma_{tr,g}$.

2.1.4 Diffusion Approximation

Another approximation that can be made to the angular dependence of the transport equation is the diffusion approximation. To derive this approximation, we again begin with the multi-group transport equation from Equation 2.11. First, we obtain the P$_1$ form of the equation by operating on it by $\int(\cdot)\,d\Omega$ and $\int(\cdot)\,\Omega_i d\Omega$. To simplify these integrals, we make use of the following identities for integrating over $\mathbf{\Omega}$:

$$\int_{4\pi} d\Omega = 4\pi \ , \tag{2.20a}$$

$$\int_{4\pi} \Omega_i d\Omega = 0 \ , \tag{2.20b}$$

$$\int_{4\pi} \Omega_i\Omega_j d\Omega = \frac{4\pi}{3}\delta_{i,j} \ , \tag{2.20c}$$

$$\int_{4\pi} \Omega_i\Omega_j\Omega_k d\Omega = 0 \ . \tag{2.20d}$$

We must also assume that the angular flux is linearly anisotropic. This leads to the following expression of the angular flux as a function of the scalar flux ϕ and current $\boldsymbol{J}$:

$$\varphi_g(\boldsymbol{x},\boldsymbol{\Omega}) \approx \frac{1}{4\pi}\left(\phi_g(\boldsymbol{x}) + 3\boldsymbol{\Omega}\cdot\boldsymbol{J}(\boldsymbol{x})\right) , \tag{2.21a}$$

$$\phi_g(\boldsymbol{x}) = \int\limits_{4\pi} \varphi_g(\boldsymbol{x},\boldsymbol{\Omega})\, d\Omega , \tag{2.21b}$$

$$J_g(\boldsymbol{x}) = \int\limits_{4\pi} \varphi_g(\boldsymbol{x},\boldsymbol{\Omega})\, \boldsymbol{\Omega} d\Omega . \tag{2.21c}$$

Applying this assumption and integrating, we obtain a coupled set of four equations for the scalar flux and the three components of the current vector.

$$\frac{dJ_{x,g}}{dx}+\frac{dJ_{y,g}}{dy}+\frac{dJ_{z,g}}{dz}+\Sigma_{t,g}(\boldsymbol{x})\phi_g(\boldsymbol{x}) = \sum_{g'=1}^{G}\Sigma_{s0,g'\to g}(\boldsymbol{x})\phi_{g'}(\boldsymbol{x})+\frac{1}{k_{eff}}\frac{\chi_g}{4\pi}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\boldsymbol{x})\phi_{g'}(\boldsymbol{x}) , \tag{2.22a}$$

$$\frac{d\phi_g}{dx} + \Sigma_{t,g}(\boldsymbol{x})J_{x,g}(\boldsymbol{x}) = \sum_{g'=1}^{G}\Sigma_{s1,g'\to g}(\boldsymbol{x})J_{x,g'}(\boldsymbol{x}) , \tag{2.22b}$$

$$\frac{d\phi_g}{dy} + \Sigma_{t,g}(y)J_{y,g}(\boldsymbol{x}) = \sum_{g'=1}^{G}\Sigma_{s1,g'\to g}(\boldsymbol{x})J_{y,g'}(\boldsymbol{x}) , \tag{2.22c}$$

$$\frac{d\phi_g}{dz} + \Sigma_{t,g}(\boldsymbol{x})J_{z,g}(\boldsymbol{x}) = \sum_{g'=1}^{G}\Sigma_{s1,g'\to g}(\boldsymbol{x})J_{z,g'}(\boldsymbol{x}) . \tag{2.22d}$$

Solving Equations 2.22b-2.22d for the components of the current, we obtain a relationship between the current and scalar flux, shown in Equation 2.23. Equation 2.22 naturally leads to the out-scatter approximation for the transport cross section, but either of the other methods from Section 2.1.3.2 can be used instead.

$$\boldsymbol{J}_g(\boldsymbol{x}) = -\boldsymbol{D}_g(\boldsymbol{x})\nabla\phi_g(\boldsymbol{x}) , \tag{2.23a}$$

$$\boldsymbol{D}_g(\boldsymbol{x}) = \frac{1}{3}\left(\Sigma_{tr,g}\right)^{-1} . \tag{2.23b}$$

Substituting 2.23a into 2.22a gives us the diffusion form of the transport equation, which has only the scalar flux ϕ and eigenvalue $\frac{1}{k_{eff}}$ as unknowns:

$$-\boldsymbol{\nabla}\cdot\boldsymbol{D}_g(\boldsymbol{x})\boldsymbol{\nabla}\phi_g(\boldsymbol{x})+\Sigma_{t,g}(\boldsymbol{x})\phi_g(\boldsymbol{x})=\sum_{g'=1}^{G}\Sigma_{s0,g'\to g}(\boldsymbol{x})\phi_{g'}(\boldsymbol{x})+\frac{1}{k_{eff}}\frac{\chi_g}{4\pi}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\boldsymbol{x})\phi_{g'}(\boldsymbol{x})\,. \tag{2.24}$$

To derive boundary conditions, Equation 2.11b is used, making the same linear anisotropy assumption as in the derivation of the diffusion equation itself:

$$\begin{aligned}\int\limits_{\boldsymbol{\Omega}\cdot\boldsymbol{n}<0}\left|\boldsymbol{\Omega}\cdot\boldsymbol{n}\right|\varphi_g(\boldsymbol{x}_b,\boldsymbol{\Omega})\,d\Omega&=\int\limits_{\boldsymbol{\Omega}\cdot\boldsymbol{n}<0}\left|\boldsymbol{\Omega}\cdot\boldsymbol{n}\right|\varphi_g^b(\boldsymbol{x}_b,\boldsymbol{\Omega})\,d\Omega\\ \int\limits_{\boldsymbol{\Omega}\cdot\boldsymbol{n}<0}\left|\boldsymbol{\Omega}\cdot\boldsymbol{n}\right|\frac{1}{4\pi}\left[\phi_g(\boldsymbol{x}_b)+3\boldsymbol{\Omega}\cdot\boldsymbol{J}_g(\boldsymbol{x}_b)\right]d\Omega&=J^-(\boldsymbol{x}_b)\\ -\frac{1}{4\pi}\int\limits_{\boldsymbol{\Omega}\cdot\boldsymbol{n}<0}\boldsymbol{\Omega}\cdot\boldsymbol{n}\left[\phi_g(\boldsymbol{x}_b)+3\boldsymbol{\Omega}\cdot\boldsymbol{J}_g(\boldsymbol{x}_b)\right]d\Omega&=J^-(\boldsymbol{x}_b)\\ \frac{1}{4}\phi_g(\boldsymbol{x}_b)-\frac{1}{2}\boldsymbol{n}\cdot\boldsymbol{J}(\boldsymbol{x}_b)&=J^-(\boldsymbol{x}_b)\ .\end{aligned} \tag{2.25}$$

This is the Marshak boundary condition, which preserves the total incident flux at every point on the boundary of the system while assuming linear anisotropy. This boundary condition can be further simplified so that the only unknown is the scalar flux ϕ_g by substituting Equation 2.23a:

$$\frac{1}{4}\phi_g(\boldsymbol{x}_b)+\frac{\boldsymbol{D}_g(\boldsymbol{x})}{2}\cdot\boldsymbol{\nabla}\phi_g(\boldsymbol{x}_b)=J^-(\boldsymbol{x}_b)\ . \tag{2.26}$$

2.2 Numerical Methods

This section will present some of basic numerical methods used to solve the transport equation. There are many different numerical methods that can be used, but only those which are important to this work will be described here. For the most extensively used ones, detailed descriptions or derivations will be included. Others will simply be described at a high level with details left to the appendices and references

2.2.1 Method of Characteristics

One method that is commonly used to solve the transport equation is the Method of Characteristics (MOC) [1, 33]. This method allows the transport equation to be solved along a characteristic direction with limited approximations, making it useful for complicated geometries such as those found in nuclear reactors. Doing this along many lines for a variety of angles and integrating allows an accurate calculation of the scalar flux for large, geometrically complex problems, making MOC useful for many reactor calculations. This section focuses on a detailed derivation of MOC.

To derive MOC, we make use of both the multi-group and discrete ordinates approximations by beginning with Equation 2.14. First, the right-hand side is lumped into a single source term for a given energy group g and direction n:

$$\boldsymbol{\Omega}_n \cdot \boldsymbol{\nabla}\varphi_{g,n} + \Sigma_{t,g}(\boldsymbol{x})\varphi_{g,n}(\boldsymbol{x}) = q_{g,n}(\boldsymbol{x}) \ , \tag{2.27a}$$

$$q_{g,n}(\boldsymbol{x}) = \frac{1}{4\pi}\sum_{g'=1}^{G}\sum_{n'=1}^{N}\Sigma_{s,g'\to g,n'\to n}(\boldsymbol{x})\varphi_{g',n'}(\boldsymbol{x})w_{n'} + \frac{1}{k_{eff}}\frac{\chi_g}{4\pi}\sum_{g'=1}^{G}\sum_{n'=1}^{N}\nu\Sigma_{f,g'}(\boldsymbol{x})\varphi_{g',n'}(\boldsymbol{x})w_{n'} \ . \tag{2.27b}$$

Now we introduce a characteristic direction $\boldsymbol{r}$ in the direction $\boldsymbol{\Omega}_n$

$$\boldsymbol{r} = \boldsymbol{r}_0 + s\boldsymbol{\Omega}_n \Rightarrow \begin{cases} x(s) = x_0 + s\Omega_{n,x} \\ y(s) = y_0 + s\Omega_{n,y} \\ z(s) = z_0 + s\Omega_{n,z} \end{cases} . \tag{2.28}$$

Substituting this variable into Equation 2.27 gives the characteristic form of this equation:

$$\frac{\partial\varphi_{g,n}}{\partial s} + \Sigma_{t,g}(\boldsymbol{r}_0 + s\boldsymbol{\Omega}_n)\varphi_{g,n}(\boldsymbol{r}_0 + s\boldsymbol{\Omega}_n) = q_{g,n}(\boldsymbol{r}_0 + s\boldsymbol{\Omega}_n) \ , \tag{2.29a}$$

$$\begin{aligned} q_{g,n}(\boldsymbol{r}_0 + s\boldsymbol{\Omega}_n) \ &= \frac{1}{4\pi}\sum_{g'=1}^{G}\sum_{n'=1}^{N}\Sigma_{s,g'\to g,n'\to n}(\boldsymbol{r}_0 + s\boldsymbol{\Omega}_n)\varphi_{g',n'}(\boldsymbol{r}_0 + s\boldsymbol{\Omega}_n)w_{n'} \\ &+ \frac{1}{k_{eff}}\frac{\chi_g}{4\pi}\sum_{g'=1}^{G}\sum_{n'=1}^{N}\nu\Sigma_{f,g'}(\boldsymbol{r}_0 + s\boldsymbol{\Omega}_n)\varphi_{g',n'}(\boldsymbol{r}_0 + s\boldsymbol{\Omega}_n)w_{n'} \ . \end{aligned} \tag{2.29b}$$

This equation is easily solved with the integrating factor

$$\exp\left(-\int_0^s \Sigma_{t,g}(\boldsymbol{r}_0 + s'\boldsymbol{\Omega}_n)ds'\right) \tag{2.30}$$

giving the following solution:

$$\begin{aligned}\varphi_{g,n}(\boldsymbol{r}_0 + s\boldsymbol{\Omega}_n) \ &= \varphi_{g,n}(\boldsymbol{r}_0)\exp\left(-\int\limits_0^s \Sigma_{t,g}(\boldsymbol{r}_0 + s'\boldsymbol{\Omega}_n)\,ds'\right) \\ &+ \int\limits_0^s q_{g,n}(\boldsymbol{r}_0 + s'\boldsymbol{\Omega}_n)\exp\left(-\int\limits_0^{s'} \Sigma_{t,g}(\boldsymbol{r}_0 + s''\boldsymbol{\Omega}_n)\,ds''\right)ds' \ .\end{aligned} \tag{2.31}$$

Now if $\boldsymbol{r}_0$ is on the boundary of the region of interest, the incoming flux $\varphi(\boldsymbol{r}_0, E, \boldsymbol{\Omega})$ is equal to the specified boundary condition φ^{in}. This allows the angular flux to be calculated at any point s to be calculated along the characteristic direction $\boldsymbol{r}$.

Up to this point, no approximations have been made in deriving the method of characteristics. One approximation that is necessary to use MOC for practical problems is to assume some spatial shape for the source term q. The simplest of these is the flat source approximation. In this approximation, q is assumed to be flat along $\boldsymbol{r}$. Additionally, it is assumed that the cross sections along each segment of the ray will be constant. This simplifies Equation 2.31 to the following:

$$\begin{aligned}\varphi_{g,n}(\boldsymbol{r}_0 + s\boldsymbol{\Omega}_n) &= \varphi_{g,n}(\boldsymbol{r}_0)\,e^{-\int_0^s \Sigma_{t,g} ds'} + \int\limits_0^s q_{g,n} e^{-\int_0^{s'} \Sigma_{t,g} ds''} ds' \\ &= \varphi^{in} e^{-\Sigma_{t,g} s} + \int\limits_0^s q_{g,n} e^{-\Sigma_{t,g} s'} ds' \\ &= \varphi^{in} e^{-\Sigma_{t,g} s} + \frac{q_{g,n}}{\Sigma_{t,g}}\left(1 - e^{-\Sigma_{t,g} s}\right) \ .\end{aligned} \tag{2.32}$$

For a segment of length A beginning on the boundary of region i, traveling in direction n along track j, Equation 2.32 allows us to solve for the outgoing angular flux at the end of track j (2.33a), as well as the average angular flux along the characteristic direction for a specific track j (2.33b).

$$\varphi^{out}_{g,i,n,j} = \varphi_g\left(\boldsymbol{r}_0 + A_j\boldsymbol{\Omega}_n\right) = \varphi^{in}_{g,i,n,j} e^{-\Sigma_{t,g,i} A_j} + \frac{q_{g,i,n}}{\Sigma_{t,g,i}}\left(1 - e^{-\Sigma_{t,g,i} A_j}\right) \ , \tag{2.33a}$$

$$
\begin{aligned}
\overline{\varphi}_{g,i,n,j} &= \frac{\int_0^{A_j} \varphi_g(\boldsymbol{r}_0 + s\boldsymbol{\Omega}_n)\, ds}{\int_0^{A_j} ds} \\
&= \frac{1}{A_j}\int_0^{A_j} \varphi^{in}_{g,i,n,j} e^{-\Sigma_{t,g,i}s} + \frac{q_{g,n,i}}{\Sigma_{t,g,i}}\left(1 - e^{-\Sigma_{t,g,i}s}\right) ds \\
&= \frac{1}{A_j}\int_0^{A_j} \frac{q_{g,n,i}}{\Sigma_{t,g,i}} + \left(\varphi^{in}_{g,i,n,j} - \frac{q_{g,n,i}}{\Sigma_{t,g,i}}\right) e^{-\Sigma_{t,g,i}s} ds \\
&= \frac{q_{g,n,i}}{\Sigma_{t,g,i}} + \frac{1}{A_j \Sigma_{t,g,i}}\left(\varphi^{in}_{g,i,n,j} - \frac{q_{g,n,i}}{\Sigma_{t,g,i}}\right)\left(1 - e^{-\Sigma_{t,g,i}A_j}\right) . \qquad (2.33b)
\end{aligned}
$$

If the transport cross section is constant along s for the region of interest, then the only approximation that has been made in this derivation is that the source is constant. It is possible to assume higher order shapes for the source (linear, quadratic, etc.) to improve the accuracy of MOC [34], but these will not be discussed here.

Typically multiple tracks are laid down across a region in each direction and integrated, with some spacing δx between them. To calculate the region average angular flux for region i, the average angular flux from Equation 2.33b for each track must be area-averaged, as shown below:

$$
\overline{\varphi}_{g,i,n} = \frac{\sum_j \overline{\varphi}_{g,i,n,j} \delta x A_j}{\sum_j \delta x A_j} . \qquad (2.34)
$$

Using this, a quadrature can be used to integrate the average region angular flux for each angle to obtain a region-averaged angular flux.

Using the outgoing flux in Equation 2.33a along the edge of a region boundary as the incoming angle flux along a neighboring region allows MOC to be used to solve across entire domains regardless of geometry. Doing this in many directions and applying Equations 2.33b and 2.34.

2.2.2 Coarse Mesh Finite Difference

Because transport calculations can take many iterations to solve, it is important to accelerate the convergence of the iteration scheme when possible. The most common way of doing this is the Coarse-Mesh Finite Difference (CMFD) method [35]. This method involves solving a diffusion problem on a coarse grid to get the average magnitude of the flux in each coarse cell. This is then used to scale the flux solution on the fine grid. In order to ensure consistency between the transport solution and the lower order diffusion solution, coupling coefficients are calculated on the boundaries of the coarse mesh cells to correct the leakage between the cells.

To describe the CMFD method, the scalar flux, current, and cross sections on the coarse grid must be related to the same quantities on the fine grid. We assume the fine mesh has already been spatially discretized and solved so that quantities such as volume, cell-averaged flux, cell-averaged cross sections, and cell-averaged source terms are known for each fine cell. With this assumption, we can define the needed coarse mesh values for each coarse cell i in terms of the fine cells it owns. N_f is the total number of fine cells owned by a coarse cell, (x_i, y_i, z_i) are the coordinates for the center of coarse cell i, and $x_{i+\frac{1}{2}}$, $x_{i-\frac{1}{2}}$, $y_{i+\frac{1}{2}}$, $y_{i-\frac{1}{2}}$, $z_{i+\frac{1}{2}}$, and $z_{i-\frac{1}{2}}$ define the edges of the coarse cell in each direction.

$$A_{x,i} = \left(y_{i+\frac{1}{2}} - y_{i-\frac{1}{2}}\right)\left(z_{i+\frac{1}{2}} - z_{i-\frac{1}{2}}\right) , \tag{2.35a}$$

$$A_{y,i} = \left(x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}\right)\left(z_{i+\frac{1}{2}} - z_{i-\frac{1}{2}}\right) , \tag{2.35b}$$

$$A_{z,i} = \left(x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}\right)\left(y_{i+\frac{1}{2}} - y_{i-\frac{1}{2}}\right) , \tag{2.35c}$$

$$V_i = \sum_{j=1}^{N_f} V_j , \tag{2.35d}$$

$$\phi_{g,i} = \frac{1}{V_i} \sum_{j=1}^{N_f} \phi_{g,j} V_j , \tag{2.35e}$$

$$\Sigma_{t,g,i} = \frac{1}{\phi_{g,i} V_i} \sum_{j=1}^{N_f} \Sigma_{tr,g,j} \phi_{g,j} V_j , \tag{2.35f}$$

$$\Sigma_{s0,g'\to g,i} = \frac{1}{\phi_{g',i} V_i} \sum_{j=1}^{N_f} \Sigma_{s0,g'\to g,j} \phi_{g',j} V_j , \tag{2.35g}$$

$$\nu\Sigma_{f,g,i} = \frac{1}{\phi_{g,i} V_i} \sum_{j=1}^{N_f} \nu\Sigma_{f,g,j} \phi_{g,j} V_j , \tag{2.35h}$$

$$\chi_{g,i} = \frac{\sum_{j=1}^{N_f} \chi_{g,j} \sum_{g'=1}^{G} \nu\Sigma_{f,g',j} \phi_{g',j} V_j}{\sum_{j=1}^{N_f} \sum_{g'=1}^{G} \nu\Sigma_{f,g',j} \phi_{g',j} V_j} , \tag{2.35i}$$

$$Q_{g,i} = \frac{1}{V_i} \sum_{j=1}^{N_f} Q_{g,j} V_j . \tag{2.35j}$$

Furthermore, currents must be tallied on the coarse mesh cell boundaries by the fine mesh transport calculations. Using these definitions, we now operate on the multi-group diffusion equation (2.22a) by Equation 2.36a to obtain Equation 2.36b.

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \int_{y_{i-\frac{1}{2}}}^{y_{i+\frac{1}{2}}} \int_{z_{i-\frac{1}{2}}}^{z_{i+\frac{1}{2}}} (\cdot)\, dzdydz \,, \tag{2.36a}$$

$$\begin{gathered} A_{x,i}\left(J_{x,g}\left(x_{i+\frac{1}{2}}\right) - J_{x,g}\left(x_{i-\frac{1}{2}}\right)\right) + A_{y,i}\left(J_{y,g}\left(y_{i+\frac{1}{2}}\right) - J_{y,g}\left(y_{i-\frac{1}{2}}\right)\right) \\ + A_{z,i}\left(J_{z,g}\left(z_{i+\frac{1}{2}}\right) - J_{z,g}\left(z_{i-\frac{1}{2}}\right)\right) + \Sigma_{tr,g,i}\phi_{g,i}V_i = \sum_{g'=1}^{G} \Sigma_{s0,g'\to g,i}\phi_{g',i}V_i \\ + \frac{1}{k_{eff}}\frac{\chi_{g,i}}{4\pi}\sum_{g'=1}^{G} \nu\Sigma_{f,g',i}\phi_{g',i}V_i \ . \end{gathered} \tag{2.36b}$$

Using Equations 2.23, we can define the interface currents in terms of the diffusion constants on the positive (p) and negative (n) sides of the interface (Equation 2.37a. The interface diffusion coefficients are defined in 2.37b-2.37d.

$$J_g = -D_g\left(\phi_{g,p} - \phi_{g,n}\right) \,, \tag{2.37a}$$

$$D_g = \frac{1}{3}\left(\left(\Delta x_p \Sigma_{tr,g,i+1} + \Delta x_n \Sigma_{tr,g,i}\right)\right)^{-1} \,, \tag{2.37b}$$

$$D_{g,\frac{1}{2}} = \alpha\left(1 + 3\Delta x_p \alpha \Sigma_{tr,g,1}\right)^{-1} \,, \tag{2.37c}$$

$$D_{g,N+\frac{1}{2}} = \alpha\left(3\Delta x_n \alpha \Sigma_{tr,g,N} + 1\right)^{-1} \,, \tag{2.37d}$$

$$\alpha = \begin{cases} 0 & \text{, reflecting} \\ 0.5 & \text{, vacuum} \end{cases} \,. \tag{2.37e}$$

Defining the interface currents as in Equation 2.37a will cause inconsistency between the high-order transport solver and the diffusion-based CMFD solver. To account for this, a correction factor can be defined in terms of the transport solution and added to the CMFD currents to enforce consistency between the two solutions.

$$J_{CMFD,g} = -D_g\left(\phi_{g,p} - \phi_{g,n}\right) + \hat{D}_g\left(\phi_{g,p} + \phi_{g,n}\right) \,, \tag{2.38a}$$

$$\hat{D}_g = \frac{J_{trans,g} + D_g\left(\phi_{g,p} - \phi g, n\right)}{\left(\phi_{g,p} + \phi_{g,n}\right)} \,. \tag{2.38b}$$

Using this definition of the current, all terms in Equation 2.36b are defined. This gives a system of $N \times G$ equations, where N is the number of coarse nodes and G is the number of energy groups in the problem, which can be written in matrix form and solved as an eigenvalue problem:

$$\underline{\underline{M}}\underline{\phi} = \frac{1}{k}\underline{\underline{F}}\underline{\phi} \, . \tag{2.39}$$

Using the homogenized transport solution as an initial guess on the right-hand side of Equation 2.39, this equation can be easily solved using a standard eigenvalue solver. MPACT generally uses either power iteration or generalized Davidson methods to solve this problem, but these methods will not be discussed in detail here.

After obtaining a solution Equation 2.39, the solution must be used to update the fine mesh solution for the next transport calculation. To do this, a simple scaling is applied to the previous transport solution as shown in Equation 2.40. This scaling is applied to all fine mesh cells j in each coarse mesh cell i and ensures preservation of reaction rates between the CMFD and transport solutions prior to the next transport sweep.

$$\phi^{k}_{trans,g,j} = \frac{\phi^{k}_{CMFD,g,i}}{\phi^{k-1}_{CMFD,g,i}} \phi^{k-1}_{trans,g,j} \, . \tag{2.40}$$

Performing the next transport calculation and continuing to iterate between the fine mesh transport calculations and CMFD will provide a converged solution far more quickly than performing iterations consisting only of fine mesh transport calculations.

2.2.3 Nodal Methods

2.2.3.1 Simplified Spherical Harmonics

One method that can be used to solve the transport equation is the Spherical Harmonics (P$_n$) method [36]. This method is able to resolve the angle dependence of the angular flux by expanding the flux using spherical harmonic functions. These functions are defined in Equation 2.41a, with the Associated Legendre Functions and Legendre Polynomials defined in Equations 2.41b and 2.15, respectively.

$$Y_n^m(\mathbf{\Omega}) = \left[\frac{2n+1}{4\pi}\frac{(n-|m|)\,!}{(n+|m|)\,!}\right]^{\frac{1}{2}} P_n^{|m|}(\mu)\, e^{im\gamma}, \quad 0 \le |m| \le n < \infty \, , \tag{2.41a}$$

$$P_n^m(\mu) = \left(1-\mu^2\right)^{\frac{m}{2}} \frac{d}{d\mu}^{m} P_n(\mu), \quad 0 \le m \le n < \infty \, . \tag{2.41b}$$

The angular flux can then be expanded in terms of the spherical harmonic functions, as shown in Equation 2.42. When using the P$_n$ approximation, some integer N is specified.

The first summation in Equation 2.42 is then truncated at N. This gives $(N+1)^2G$ expansion coefficients that must be determined. To solve for these coefficients, the transport equation must be multiplied by $Y_n^{-m}(\boldsymbol{\Omega})$ and integrated over $\boldsymbol{\Omega}$ for each combination of m, n, and g.

$$\varphi_g(\boldsymbol{x},\boldsymbol{\Omega}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\varphi_{n,m,g}(\boldsymbol{x})\,Y_n^m(\boldsymbol{\Omega}) \ . \tag{2.42}$$

While the P$_n$ method can be used to solve the transport equation, this is rarely done in practice due to the increasing complexity of the equations and the quadratic growth in the number of unknowns as N increases. Instead, the more common method used in practice is the Simplified Spherical Harmonics (SP$_n$) method. In 1D planar geometry, the angular variable γ vanishes, leaving only μ. In this case, only the spherical harmonic functions for $m=0$ are needed, since these are the functions independent of γ. Thus, the flux expansion reduces to a Legendre polynomial expansion:

$$\varphi_g(x,\mu) = \sum_{m=0}^{\infty}\frac{2m+1}{2}\varphi_{m,g}(x)\,P_m(\mu) \ , \tag{2.43a}$$

$$\varphi_{m,g}(x) = \int_{-1}^{1} P_m(\mu')\,\varphi_g(x,\mu')\,d\mu' \ . \tag{2.43b}$$

To solve, we apply the expansion in Equation 2.43 to the transport equation, multiply by Legendre polynomials $P_n(\mu)$ for $0 \le n \le N$, and integrate over μ. Following this procedure results in the 1D planar geometry P$_1$ equations, shown in Equation 2.44:

$$\frac{d\varphi_{1,g}}{dx} + \Sigma_{t,g}(x)\,\varphi_{0,g}(x) = \sum_{g'=1}^{G}\sigma_{s0,g'\to g}(x)\,\varphi_{0,g'}(x) + \frac{1}{k_{eff}}\frac{\chi_g}{4\pi}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(x)\,\varphi_{0,g'}(x) \ , \tag{2.44a}$$

$$\frac{d\varphi_{0,g}}{dx} + \Sigma_{t,g}(x)\,\varphi_{1,g}(x) = \sum_{g'=1}^{G}\Sigma_{s1,g'\to g}(x)\,\varphi_{1,g'}(x) \ . \tag{2.44b}$$

It is clear that if $\frac{d}{dx}$ in Equation 2.44 is replaced by $\boldsymbol{\nabla}$, then the 3D P$_1$ equations are obtained exactly as shown in Equation 2.22. For any odd $N > 1$, following this same process and making the same observation about the derivative terms will result in the 3D SP$_n$ equations. For $N = 1$, these equations are exactly equal to the P$_n$ equations. However, for $N > 1$, this results in a simplified system of equations which still preserve much of the transport physics that are not preserved by the P$_1$ or diffusion approximations. Appendix A contains a detailed derivation of the SP$_3$ equations, but the final 3D SP$_3$ equations are shown in Equation 2.45.

$$\begin{aligned}-\boldsymbol{\nabla}\cdot D_{0,g}(\boldsymbol{x})\boldsymbol{\nabla}\Phi_{0,g}(\boldsymbol{x}) + \left[\Sigma_{tr,g}(\boldsymbol{x}) - \Sigma_{s0,g}(\boldsymbol{x})\right]\Phi_{0,g}(\boldsymbol{x}) \\ = Q_g(\boldsymbol{x}) + 2\left[\Sigma_{tr,g}(\boldsymbol{x}) - \Sigma_{s0,g}(\boldsymbol{x})\right]\Phi_{2,g}(\boldsymbol{x})\ ,\end{aligned} \tag{2.45a}$$

$$\begin{aligned}-\boldsymbol{\nabla}\cdot D_{2,g}(\boldsymbol{x})\boldsymbol{\nabla}\Phi_{2,g}(\boldsymbol{x}) + \left[\Sigma_{tr,g}(\boldsymbol{x}) - \Sigma_{s2,g}(\boldsymbol{x})\right]\Phi_{2,g}(\boldsymbol{x}) \\ = \frac{2}{5}\left\{\left[\Sigma_{tr,g}(\boldsymbol{x}) - \Sigma_{s0,g}(\boldsymbol{x})\right]\left[\Phi_{0,g}(\boldsymbol{x}) - 2\Phi_{2,g}(\boldsymbol{x})\right] - Q_g(\boldsymbol{x})\right\}\ .\end{aligned} \tag{2.45b}$$

Each of these equations is similar in form to the P$_1$ equation and can be solved by iterating between the the 0th moment equation (2.45a) and the 2nd moment equation (2.45b). This results in an accurate 3D flux distribution without the expense and complexity of performing a 3D P$_N$ calculation.

In this work, the SP$_N$ equations are used only for 1D calculations. In 1D, SP$_N$ is exactly equivalent to P$_N$ for all values of N. Because of this, this method is generally referred to as P$_N$ in later chapters of this dissertation and other recent publications related to the MPACT code.

2.2.3.2 Nodal Expansion Method

While SP$_N$ can be used to capture angular moments of the flux, the Nodal Expansion Method (NEM) [2] is used to capture intra-nodal flux shapes to calculate accurate currents at the interface between two nodes. This is done by expanding the source and flux using quadratic and quartic Polynomials, respectively:

$$Q(\xi) = \sum_{i=0}^{2} q_i P_i(\xi)\ , \tag{2.46a}$$

$$\phi(\xi) = \sum_{i=0}^{4} \phi_i P_i(\xi)\ , \tag{2.46b}$$

where the variable ξ is simply the spatial variable normalized so the problem is being solved on the interval $[-1,1]$. To solve for these moments, five equations are required. The first three are the moment balance equations, obtained by multiplying the diffusion equation by $P_n(\xi)$ for moments 0-2 and integrating (Equation 2.47a). The other two equations are found by enforcing flux and current continuity at the interfaces between nodes (Equations

2.47b-2.47c).

$$\int_{-1}^{1} P_n(\xi)\left(-\frac{D}{h^2}\frac{d^2}{d\xi^2}\phi(\xi) + \Sigma_r\phi(\xi) - Q(\xi)\right)d\xi = 0,\ n = 0,1,2\ , \tag{2.47a}$$

$$\phi_L(1) = \phi_R(-1)\ , \tag{2.47b}$$

$$J_L(1) = J_R(-1)\ . \tag{2.47c}$$

The source moments q_i are constructed using the flux moments from the previous iteration. Using this method, an intra-nodal flux shape can be calculated within each node, and interface currents can be calculated at the boundaries of each node.

2.2.4 Collision Probabilities Method

One method that can be used to calculate flux spectra inside a pin cell is the method of Collision Probabilities (CP) [37]. This method is used in MPACT as part of the control rod decusping methods described in Chapter 4. The details of the derivation will be left to Appendix A, but a brief overview of the method will be presented here that focuses on a 1D radial calculation for a single pin cell.

To begin, the multi-group transport Equation 2.11 can be rewritten so that the entire right-hand side is a single source term:

$$\mathbf{\Omega}\cdot\mathbf{\nabla}\varphi_g + \Sigma_{tr,g}(\boldsymbol{x})\varphi_g(\boldsymbol{x},\mathbf{\Omega}) = q_g(\boldsymbol{x},\mathbf{\Omega})\ , \tag{2.48a}$$

$$\begin{aligned} q_g(\boldsymbol{x},\mathbf{\Omega}) = &\frac{1}{4\pi}\sum_{g'=1}^{G}\int_{4\pi}\Sigma_{s,g'\to g}(\boldsymbol{x},\mathbf{\Omega}'\to\mathbf{\Omega})\varphi_{g'}(\boldsymbol{x},\mathbf{\Omega}')d\Omega' \\ &+ \frac{1}{k_{eff}}\frac{\chi_g}{4\pi}\sum_{g'=1}^{G}\int_{4\pi}\nu\Sigma_{f,g'}(\boldsymbol{x})\varphi_{g'}(\boldsymbol{x},\mathbf{\Omega}')d\Omega'\ . \end{aligned} \tag{2.48b}$$

This form of the equation assumes no source of neutrons except scatter and fission. It also assumes isotropic (or transport-corrected) scattering, implying that the CP method cannot resolve highly anisotropic scattering. This gives way to a simplified isotropic source, shown in Equation 2.49:

$$q_g(\boldsymbol{x},\mathbf{\Omega}) \approx q_g(\boldsymbol{x}) = \frac{1}{4\pi}\sum_{g'=1}^{G}\Sigma_{s,g'\to g}(\boldsymbol{x})\phi_{g'}(\boldsymbol{x}) + \frac{1}{k_{eff}}\frac{\chi_g}{4\pi}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\boldsymbol{x})\phi_{g'}(\boldsymbol{x})\ , \tag{2.49}$$

where ϕ_g is the scalar flux in group g. At this point, we assume the problem is discretized into R regions and that the cross sections and fluxes are flat in each region, which leads to a flat source as well for each region r:

$$q_{g,r} = \frac{1}{4\pi} \sum_{g'=1}^{G} \Sigma_{s,g' \to g,r} \phi_{g',r} + \frac{1}{k_{eff}} \frac{\chi_{g,r}}{4\pi} \sum_{g'=1}^{G} \nu \Sigma_{f,g',r} \phi_{g'} \ . \tag{2.50}$$

Now the total source of neutrons in each region r can be calculated by multiplying by the region volume V_r. If the probability $T_{g,r' \to r}$ of a neutron born in region r' reaching region r is known for all regions r', then the scalar flux can be calculated as follows:

$$\phi_{g,r} = \sum_{r'=1}^{R} T_{g,r' \to r} q_{g,r'} V_{r'} \ . \tag{2.51}$$

Equation 2.51 gives the general solution to any problem using the CP method. The only remaining work that must be done is to calculate the transport matrix $\underline{\underline{T_g}}$ for each group, which is done in detail for a pin cell in the appendix.

Several important details about this method should be noted. First, it is common to use a buffer region in the calculation. When applying the CP method to a fuel pin, the buffer region is not necessary because the fuel pin has its own fission source to drive the problem. However, when using this method on a different pin cell such as a control rod, it is useful to place a homogenized mixture of fuel and moderator around the outside of the pin cell. This provides a source to drive the problem. Second, because the calculation is a 1D radial calculation, the moderator region in the rectangular pin cell must be cylindricized. This is done by transforming the moderator region into a ring with the same inner radius and volume as the rectangular moderator region. Finally, the boundary conditions assumed on the edge of the problem are “white” boundary conditions. A reflecting boundary condition assumes that all neutron exiting the problem return at the same energy and traveling in a reflected direction. However, because the boundary of this problem is a circle, it is possible for a neutron to be born at such an angle that it continuously reflects off the boundary without ever traveling toward the fuel. The white boundary condition takes all exiting neutrons and returns them isotropically to prevent this error.

CHAPTER 3

2D/1D Framework

3.1 Background

The Boltzmann transport equation can be solved directly in 3D to obtain 3D flux and power distributions. One method to do this is the 3D Method of characteristics, which is implemented in MPACT [38]. However, performing these 3D transport calculations becomes too computationally burdensome to be of practical use, even with today's improved computing resources. Because LWRs have most of their material heterogeneity in the radial direction with very little change in the axial direction, it was recognized that approximations could be made in the axial direction to increase the efficiency of the calculations while still performing high-fidelity transport calculations in the radial direction. Two different groups of researchers pursued this concept and developed two different methods of solving the transport equation for LWR problems.

The first of these methods was the "2D/1D Fusion" technique, developed by researchers at Korea Advanced Institute of Science and Technology (KAIST) and implemented in codes such as CRX [22, 39, 40]. In this method, the 3D problem is decomposed into a stack of 2D planes. These planes are solved using 2D MOC, with incoming angular fluxes on the top and bottom boundaries of the plane as source terms. To couple the planes, the problem domain is integrated in the x- and y-directions for each pin cell. The angular fluxes at the radial edges are obtained from the 2D MOC calculations and used as source terms. The angular fluxes are then solved in the axial direction using the Diamond Difference method. These results, in turn, are fed back into the radial calculations. Iterating between the radial and axial calculations then produces a full 3D solutions.

The second group of researchers was at Korea Atomic Energy Research Institute (KAERI). They developed what is known more simply as the "2D/1D" scheme, first implemented in the DeCART code [41–43]. This employs very similar technique to the 2D/1D Fusion method described above. However, rather than using angular fluxes from each

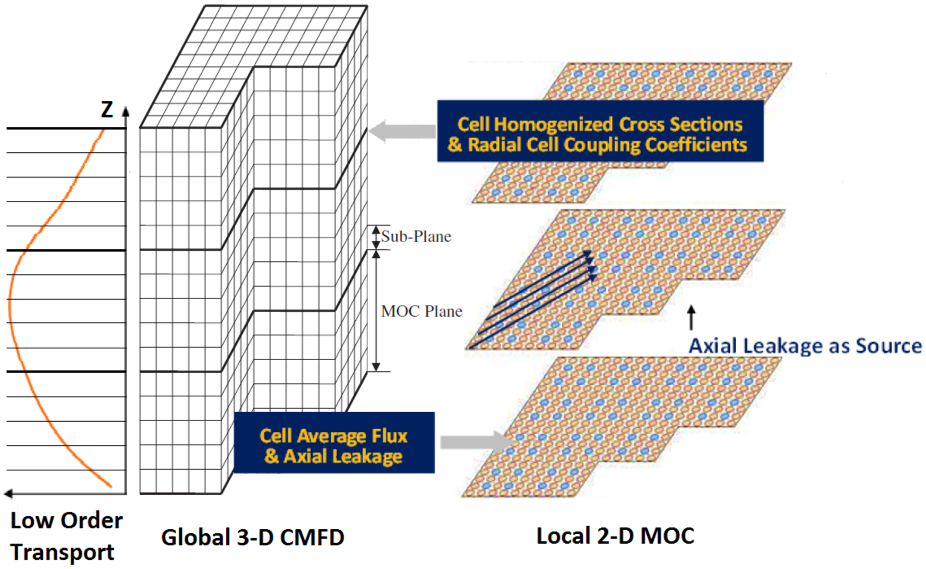

Z
Cell Homogenized Cross Sections & Radial Cell Coupling Coefficients
Sub-Plane
MOC Plane
Axial Leakage as Source
Cell Average Flux & Axial Leakage
Low Order Transport
Global 3-D CMFD
Local 2-D MOC

Figure 3.1: The 2D/1D Method illustrated with the subplane scheme for the axial and CMFD calculations

solver as a source term, currents are tallied on each of the six faces of each pin cell. The currents can then be used to compute axial and radial "transverse leakage" sources for the radial and axial solvers, respectively. This change allows for the storage of group-wise currents at each interface instead of storing the group-wise angular fluxes for each angle, significantly reducing the memory burden of the calculation.

After some development, the DeCART code was forked into several different versions for different institutions, one of them being the University of Michigan (UM). After some development, it was determined that there would be no further development of the DeCART code at UM and a new 2D/1D implementation would be put in MPACT [44, 45]. In MPACT's implementation of 2D/1D (shown in Figure 3.1), 2D MOC is used for each of the radial planes, as with earlier 2D/1D codes. The axial calculation done on a pin-homogenized mesh usually with P_3 wrapper in an NEM kernel. A variety of other solvers are available, such as SENM, P_1, P_3, and S_N, but these will not be used in this work. Finally, MPACT also uses 3D CMFD on the same pin-homogenized mesh to provide convergence acceleration to the calculations. The remainder of this chapter will look at the derivation of the 2D/1D equations, the details of how they are implemented in MPACT, and some of the approximations and sources of errors related to this method.

3.2 Derivation

3.2.1 Radial Equations

To derive the radial equations, we begin with the multigroup approximation in Equation 2.11 and integrate in the z-direction over some range $\Delta z_i = z_{k+\frac{1}{2}} - z_{k-\frac{1}{2}}$. To do this, we assume the cross sections are all constant in the interval $z \in \left[z_{k-\frac{1}{2}}, z_{k+\frac{1}{2}}\right]$. With this assumption, we obtain the following equation:

$$
\begin{aligned}
&\Omega_x \frac{\partial \varphi_g^Z}{\partial x} + \Omega_y \frac{\partial \varphi_g^Z}{\partial y} + \frac{\Omega_z}{\Delta z_k}\left(\varphi_{g,z_{k+\frac{1}{2}}} - \varphi_{g,z_{k-\frac{1}{2}}}\right) + \Sigma_{t,g}(x,y)\,\varphi_g^Z(x,y,\mathbf{\Omega}) \\
&= \frac{1}{4\pi}\sum_{g'=1}^{G}\int_{4\pi} \Sigma_{s,g'\to g}^Z(x,y,\mathbf{\Omega}'\cdot\mathbf{\Omega})\,\varphi_{g'}^Z(x,y,\mathbf{\Omega}')\,d\Omega' \\
&\quad + \frac{1}{k_{eff}}\frac{\chi_g^Z}{4\pi}\sum_{g'=1}^{G}\int_{4\pi} \nu\Sigma_{f,g'}^Z(x,y)\,\varphi_{g'}^Z(x,y,\mathbf{\Omega}')\,d\Omega' \ ,
\end{aligned}
\tag{3.1a}
$$

$$
\varphi_g^Z(x,y,\mathbf{\Omega}) = \frac{1}{\Delta z_k}\int_{z_{k-\frac{1}{2}}}^{z_{k+\frac{1}{2}}} \varphi_g^Z(x,y,z,\mathbf{\Omega})\,dz \ ,
\tag{3.1b}
$$

where a superscript Z indicates the average of a quantity over a given plane. The z-component of the streaming can now be moved to the right-hand side of the equation and treated as a source term, giving a 2D transport problem which could be solved with a variety of methods:

$$
\Omega_x \frac{\partial \varphi_g^Z}{\partial x} + \Omega_y \frac{\partial \varphi_g^Z}{\partial y} + \Sigma_{t,g}(x,y)\,\varphi_g^Z(x,y,\mathbf{\Omega}) = q_g^Z(x,y) + L_g^Z(x,y,\Omega_z) \ ,
\tag{3.2a}
$$

$$
\begin{aligned}
q_g^Z(x,y) = &\frac{1}{4\pi}\sum_{g'=1}^{G}\int_{4\pi} \Sigma_{s,g'\to g}^Z(x,y,\mathbf{\Omega}'\cdot\mathbf{\Omega})\,\varphi_{g'}^Z(x,y,\mathbf{\Omega}')\,d\Omega' \\
&+ \frac{1}{k_{eff}}\frac{\chi_g^Z}{4\pi}\sum_{g'=1}^{G}\int_{4\pi} \nu\Sigma_{f,g'}^Z(x,y)\,\varphi_{g'}^Z(x,y,\mathbf{\Omega}')\,d\Omega' \ ,
\end{aligned}
\tag{3.2b}
$$

$$
L_g^Z(x,y,\Omega_z) = \frac{\Omega_z}{\Delta z_k}\left(\varphi_{g,z_{k-\frac{1}{2}}} - \varphi_{g,z_{k+\frac{1}{2}}}\right) ,
\tag{3.2c}
$$

where $L_g^Z(x,y,\Omega_z)$ is the axial transverse leakage source term for plane z. To simplify the source term, the axial transverse leakage term is often handled isotropically. This is done by averaging over angle:

$$L_g^Z(x,y) = \frac{1}{4\pi}\int L_g^Z(x,y,\Omega_z)\,d\Omega \approx \frac{J_{g,z_{k-\frac{1}{2}}} - J_{g,z_{k+\frac{1}{2}}}}{4\pi\Delta z_k}\ , \tag{3.3}$$

where $J_{z_{i\pm\frac{1}{2}}}$ is the current at the top (+) or bottom (−) of the plane. This eliminates the need for storing all the angluar fluxes on the top and bottom of every plane. Other methods exist that allow the axial transverse leakage source to maintain its angular dependence without storing the angular fluxes [46], but these methods are not discussed here since they were not used by this work.

3.2.2 Axial Equations

The axial equations can be derived in a manner similar to the radial equations. Again, we begin with the multi-group approximation shown in Equation 2.11. This time, we integrate in both the x- and y-directions over intervals $x \in \left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right]$ and $y \in \left[y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}\right]$, giving the following equations in the axial direction, which are analogous to the radial equations in the previous section:

$$\Omega_z \frac{\partial \varphi_g^{XY}}{\partial z} + \Sigma_{t,g}^{XY}(z)\,\varphi_g^{XY}(z,\mathbf{\Omega}) = q_g^{XY}\left(z,\Omega_x,\Omega_y\right) + L_g^{XY}\left(z,\Omega_x,\Omega_y\right)\ , \tag{3.4a}$$

$$\begin{aligned} q_g^{XY}\left(z,\Omega_x,\Omega_y\right) = {} & \frac{1}{4\pi}\sum_{g'=1}^{G}\int_{4\pi}\Sigma_{s,g'\to g}^{XY}(z,\mathbf{\Omega}'\cdot\mathbf{\Omega})\,\varphi_{g'}^{XY}(z,\mathbf{\Omega}')\,d\Omega' \\ & + \frac{1}{k_{eff}}\frac{\chi_g^{XY}}{4\pi}\sum_{g'=1}^{G}\int_{4\pi}\nu\Sigma_{f,g'}^{XY}(z)\,\varphi_{g'}^{XY}(z,\mathbf{\Omega}')\,d\Omega'\ , \end{aligned} \tag{3.4b}$$

$$L_g^{XY}\left(z,\Omega_x,\Omega_y\right) = \frac{\Omega_x}{\Delta y_i}\int_{y_{i-\frac{1}{2}}}^{y_{i+\frac{1}{2}}}\left(\varphi_{g,x_{i-\frac{1}{2}}}(y) - \varphi_{g,x_{i+\frac{1}{2}}}(y)\,dy\right) + \frac{\Omega_y}{\Delta x_i}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}\left(\varphi_{g,y_{i-\frac{1}{2}}}(x) - \varphi_{g,y_{i+\frac{1}{2}}}(x)\,dx\right), \tag{3.4c}$$

$$\varphi_g^{XY} = \frac{1}{\Delta_i\Delta_j}\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}\varphi_g(x,y,z,\mathbf{\Omega})\,dxdy\ , \tag{3.4d}$$

where a superscript XY now corresponds to a particular x- and y-region which extends the full height of the problem in the z-direction. Again, it is assumed that the cross sections are constant in the x- and y-directions inside the region of integration. How this is accomplished will be discussed in more detail when discussing MPACT's implementation of P_3 and CMFD.

As with the radial equations, we can treat the transverse leakage source isotropically by averaging over angle:

$$\begin{aligned} L_g^{XY}(z) &= \frac{1}{4\pi}\int_{4\pi} L_g^{XY}\left(z,\Omega_x,\Omega_y\right) \\ &\approx \frac{1}{4\pi\Delta x_i\Delta y_j}\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}\left(J_g\left(x_{i-\frac{1}{2}},y,z\right)-J_g\left(x_{i+\frac{1}{2}},y,z\right)\right)dy \\ &\quad+\frac{1}{4\pi\Delta x_i\Delta y_j}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}\left(J_g\left(x,y_{j-\frac{1}{2}},z\right)-J_g\left(x,y_{j+\frac{1}{2}}z\right)\right)dx\,. \end{aligned} \tag{3.5}$$

As an additional approximation we assume that the currents are tallied across an entire transverse surface, meaning that the current terms in the integrals above are constant over the direction of integration. With this simplification, the final form of the radial TL is

$$L_g^{XY}(z)=\frac{J_{g,x_{i-\frac{1}{2}},y_j}(z)-J_{g,x_{i+\frac{1}{2}},y_j}(z)}{4\pi\Delta x_i}+\frac{J_{g,x_i,y_{j-\frac{1}{2}}}(z)-J_{g,x_i,y_{j+\frac{1}{2}}}(z)}{4\pi\Delta y_j}\,. \tag{3.6}$$

Again, methods have been developed to handle the angle-dependence of the radial transverse leakage source [47], but this work used only isotropic radial leakage.

3.3 Implementation

Now that the general 2D/1D scheme has been described, some attention should be given to the details of its implementation in MPACT. Figure 3.2 shows the calculation flow used by MPACT. The first step is to perform a global 3D CMFD calculation to obtain pin-averaged flux and interface currents between each cell. Next, the axial solver uses the radial currents calculated by CMFD as a radial transverse leakage source to obtain an axial transverse leakage source for the radial solver. Finally, 2D MOC is used as the radial solver to obtain a solution with sub-pin resolution in each plane.

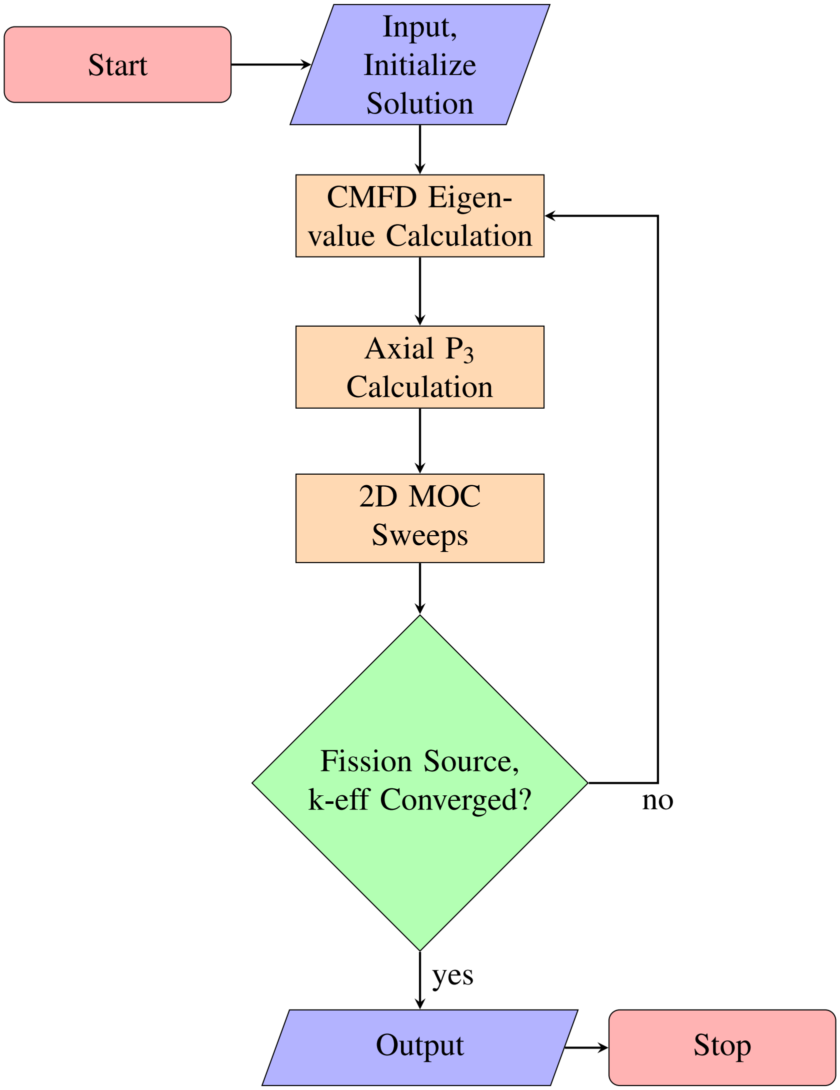

Start
Input, Initialize Solution
CMFD Eigen-value Calculation
Axial P_3 Calculation
2D MOC Sweeps
Fission Source, k-eff Converged?
no
yes
Output
Stop

Figure 3.2: Calculation flow for 2D/1D scheme

3.3.1 3D Subplane CMFD

The CMFD method was originally implemented in MPACT just as described in Section 2.2.2. To do this, each pin cell is homogenized using the quantities defined in Equation 2.35 in every plane in the model. The radial coupling coefficients defined in Equation 2.38 are obtained by calculating the current at the interface between each pair of pin cells using the 2D MOC sweeper, while the axial coupling coefficients are obtained from the axial currents calculated by the axial solve during the previous iteration. The matrix for the 3D multi-group system can then be set up and solved, typically using the generalized Davidson eigenvalue solver.

As part of this work, the traditional CMFD capability was extended to use the subplane scheme, first developed by Cho et al. for the DeCART code [48] and later used in the nTRACER code [49]. Thin MOC planes are capable of causing instability in the 2D/1D scheme, but are sometimes required to maintain accuracy. The subplane scheme allows users to increase the thickness of the 2D planes while still maintaining the accuracy of a fine axial mesh. While DeCART used the subplane scheme primarily to allow for thicker MOC planes, nTRACER also uses the subplane scheme as part of its rod decusping methods [23]. This section will focus only on the subplane scheme as a means of using fewer MOC planes, while Chapter 4 will discuss how the subplane scheme has been modified to be used in decusping methods.

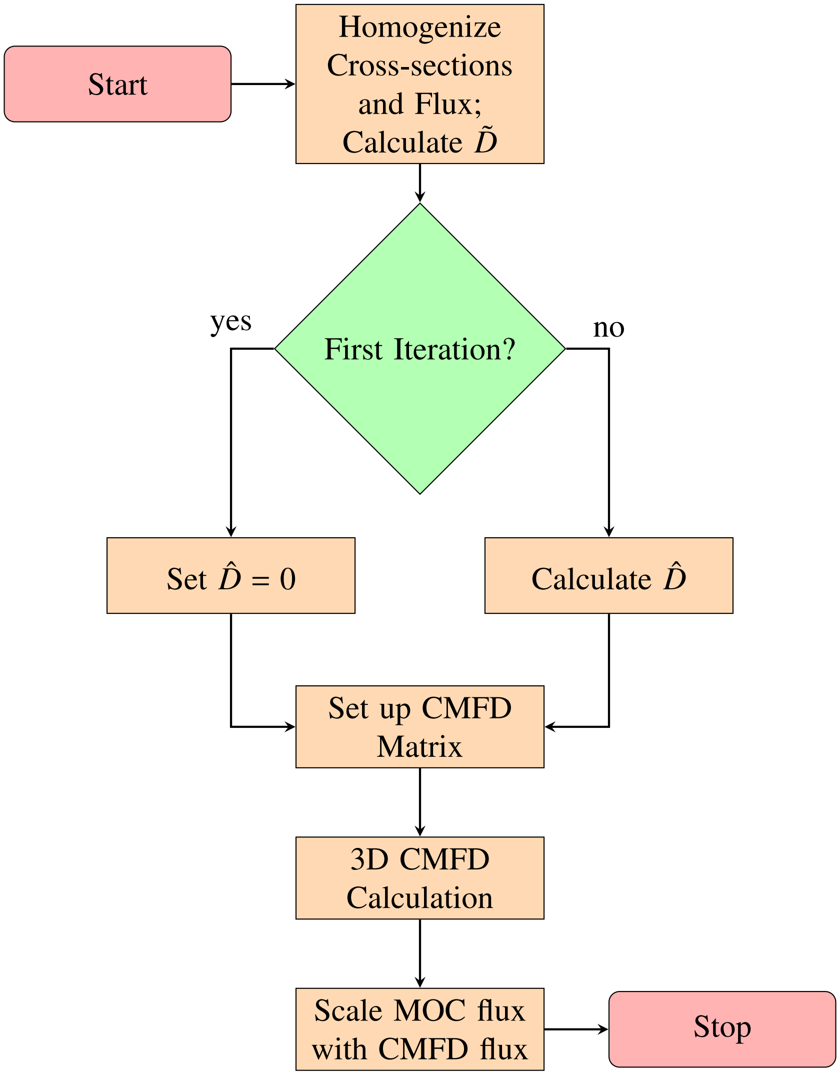


Figure 3.3: Calculation flow for 3D subplane CMFD

3.3.1.1 Homogenization

For the traditional CMFD calculations, each pin cell in an MOC plane is homogenized into a single CMFD cell. When using the subplane scheme, the homogenized pin cell is divided axially into a stack of cells. This causes the CMFD system to have multiple planes per MOC plane, allowing CMFD to capture subplane axial flux shapes that would otherwise be ignored. To do this, a subplane scaling factor is introduced which will be used to provide an axial shape within a 2D plane:

$$\begin{aligned} c_{g,i}^{k} &= \frac{\phi_{g,i}^{k-1}}{\overline{\phi}_{g,i}^{k-1}} \\ &= \frac{\phi_{g,i}^{k-1} \sum_{i'=1}^{N_{sp}} V_{i'}}{\sum_{i'=1}^{N_{sp}} \phi_{g,i'}^{k-1} V_{i'}} \,, \end{aligned} \tag{3.7}$$

where superscripts indicate which iteration the values are taken from and N_{sp} is the number of subplanes for the pin cell of interest. Now when the homogenized values are calculated from the 2D transport solution using Equation 2.35, the fine mesh flux is multiplied by this subplane scaling factor everywhere it appears. Because the 2D/1D scheme assumes a constant material axially in each plane, this subplane factor has no impact on the homogenized cross sections. However, the homogenized flux $\phi_{g,i}$ and fission source distribution $\chi_{g,i}$ will be changed, providing an axial shape for the source term in the eigenvalue calculation.

3.3.1.2 Coupling Coefficients

In addition to the homogenized cell terms, the coupling coefficients described by Equations 2.37 and 2.38 must be calculated for each subplane. To maintain consistency, the area-averaged current calculated by the radial sweeper must be preserved across the subsurfaces used by the subplane scheme. Thus, the current calculated by the radial sweeper at an interface is used at the corresponding interfaces for all subplanes in that plane. Additionally, to maintain consistency, this requires that the cell-homogenized flux used in the calculation of the diffusion coefficients be defined for the entire MOC plane as in Equation 2.35e rather than using the subplane scaling factor for each subplane.

The axial coupling coefficient can be treated in a more straightforward manner. Because the 1D axial solvers use the same pin-homogenized mesh as the CMFD solver, axial currents are naturally calculated at the top and bottom of each of the subplanes. Thus, these currents can be used together with the subplanes fluxes to calculate subplane-dependent axial coupling coefficients.

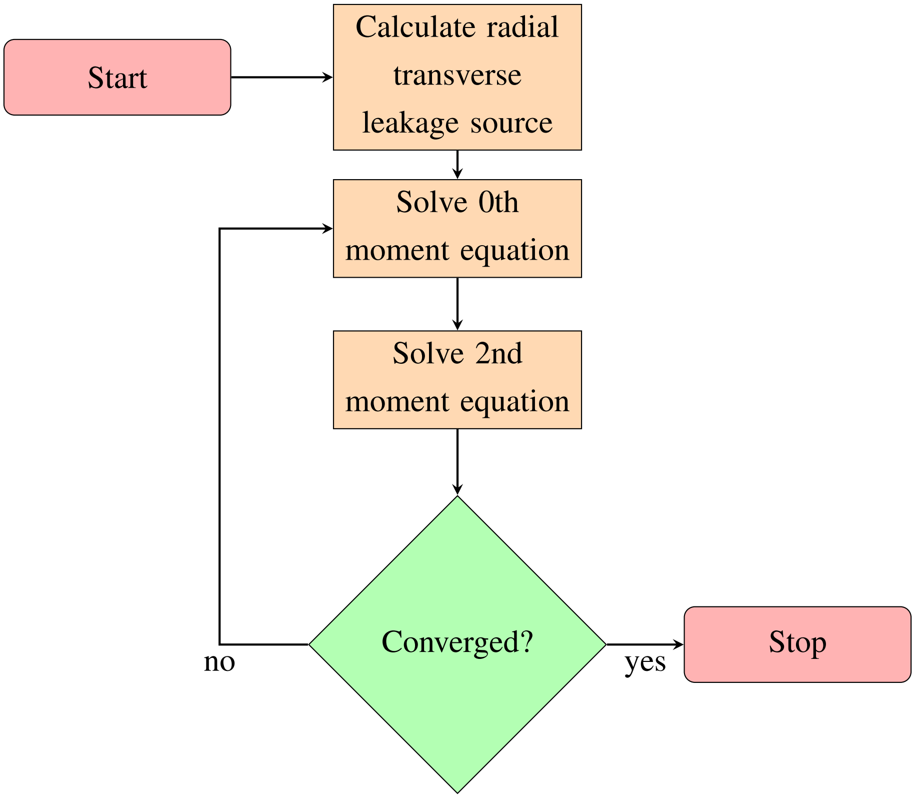
3.3.1.3 Projection

The projection of the CMFD flux back to the 2D planes must also account for the presence of the subplanes. To do this, the solution is volume-averaged over all subplanes for each pin cell, resulting in an equation similar to 2.40:

$$\phi^k_{trans,g,j} = \frac{\sum_{i'=1}^{N_{sp}} \phi^k_{CMFD,g,i'} V_i}{\sum_{i'=1}^{N_{sp}} \phi^k_{CMFD,g,i'} V_i} \phi^{k-1}_{trans,g,j} \,. \tag{3.8}$$

The calculation flow for 3D CMFD is shown in Figure 3.3.

3.3.2 1D NEM-P$_3$



Start
Calculate radial transverse leakage source
Solve 0th moment equation
Solve 2nd moment equation
Converged?
no
yes
Stop

Figure 3.4: Calculation flow for 1D axial calculations in MPACT

In MPACT, the 1D axial solvers operate on the same mesh as the 3D CMFD calculations, meaning that cell-homogenized quantities and radial currents have already been obtained from the CMFD calculation. All the 1D axial solver must do is construct a source term from the radial currents for each cell, then perform a calculation to obtain currents on the axial interfaces at the top and bottom of each node.

MPACT has a variety of 1D nodal methods that are capable of performing these calculations, including diffusion-based such as NEM and SENM and higher-order solvers such as P_N and S_N. For most calculations, MPACT uses P_3 wrapped in NEM kernels. The P_3 portion handles the angular dependence of the solution by calculating angular moments. The NEM kernels then handle the spatial dependence through the fourth-order polynomial expansion. Because the P_3 equations can be written as two sets of P_1 equations, they naturally lend themselves to being solved this way.

The P_3 equations consist of equations for angular flux moments 0 through 3. These equations can be combined into two equations for just the 0th and 2nd moments, as shown in Equations 2.45. Formulating P_3 this way makes it straightforward to use the NEM kernels for each of the two moment equations. The iteration scheme for this procedure is shown in Figure 3.4.

3.3.3 2D MOC

For the radial calculations, 2D MOC is used. This allows MPACT to easily calculate scalar fluxes and currents in each plane regardless of the geometric complexity. This section is devoted to discussing some of the details of the MOC implementation and sweeping algorithm in MPACT.

3.3.3.1 Ray Tracing

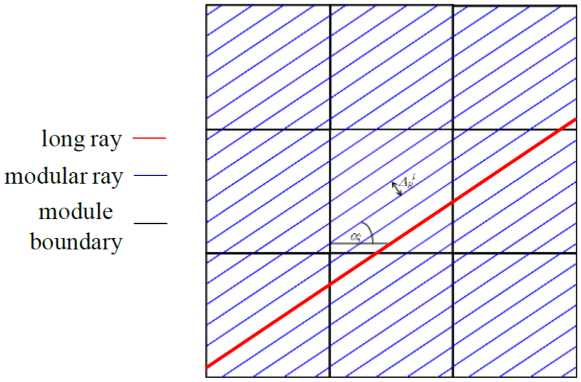


Figure 3.5: Modular ray tracing depiction [27]

One of the key features of MPACT's MOC implementation is that of modular ray tracing. Ray tracing is performed once at the beginning of a calculation and stored for the remainder of the calculation. Doing this greatly reduces the runtime of the MOC sweeps since the length of each ray segment and the region it is crossing are already known ahead of time. Furthermore, MPACT takes advantage of the repeatable nature of a reactor's geometry. Because reactor geometry is repetitive, small portions of the geometry which repeat frequently can be ray-traced separately instead of tracing the entire core. These smaller units of geometry are known as ray tracing modules, and in MPACT can be a full fuel assembly, a quarter fuel assembly, or a single fuel pin. After the unique modules are identified, each of them is ray-traced in such a way that the endpoints of a ray in each module will line up with the beginning of a ray in the neighboring module. This significantly reduces the storage requirements for the ray data since a small number of ray tracing modules can represent a full core problem. During each sweep, the long rays which traverse the entire model are reconstructed from the modular rays then swept. This adds a small amount of extra time to the sweep in exchange for significant memory savings.

As a result of the modular ray tracing, several corrections are required. First, to successfully perform the ray tracing the angles of the rays had to be adjusted slightly to line up. However, because a quadrature is used to integrate the angular flux, this angle modification requires a correction to the quadrature weights as well to maintain accuracy. Second, the spacing between the rays must also be adjusted slightly to ensure that all rays align. Along with these corrections, there are other MOC concepts such as volume corrections, cyclic rays, and others which are important to be aware of but will be deferred to the MPACT theory manual for details [27].

3.3.3.2 Sweeping Algorithm

To perform the MOC sweeps, MPACT uses a multi-group sweeping method. To do this, multi-group sources and cross sections are prepared for each of the fine mesh regions in each plane. There are then four total loops for the sweeping algorithm. From outermost to innermost, these loops are over azimuthal angle, ray (across the entire domain), polar angle, and energy group. Each ray is divided into many segments that the MOC steps its way along as described in Section 2.2.1. As these sweeps are performed, scalar flux is tallied in each fine mesh region, and, assuming CMFD is being used, currents are tallied at the interfaces between each pin cell. Normally only one inner MOC iteration is required per 2D/1D iteration. This algorithm is shown in Figure 3.6, with the sweep of a single MOC plane expanded in Figure 3.7.

MPACT also has the capability of performing the MOC sweeps with the energy loop as the outermost. This has two advantages. First, the previous group is used to construct the scattering source for the current group. This means that the iteration scheme is a Gauss-Seidel iteration instead of a Jacobi iteration, which can speed up the convergence of the problem. Secondly, the cross sections and sources only need to be stored for one group

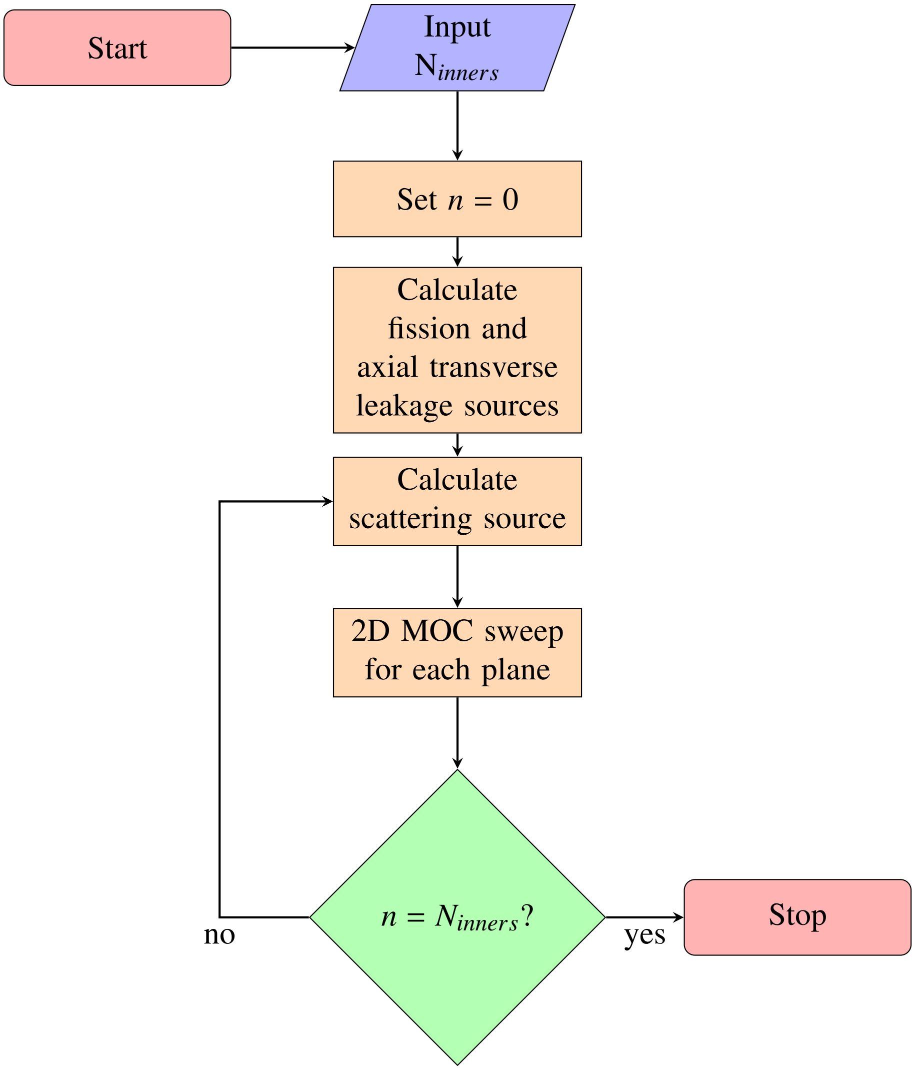

Start
Input
N_{inners}
Set $n = 0$
Calculate fission and axial transverse leakage sources
Calculate scattering source
2D MOC sweep for each plane
$n = N_{inners}$?
no
yes
Stop

Figure 3.6: Calculation flow for 2D MOC calculation in MPACT

at a time, minimizing the storage requirements for the calculations. However, when using transport-corrected cross sections, some instabilities have been observed in this iteration scheme when using only a single inner iteration. Furthermore, having the energy loop on the inside results in some improved cache efficiency when it comes to traversing the rays, reducing the runtime for a single MOC sweep [50]. For these two reasons, MPACT defaults to the Jacobi-style iteration scheme described first.

3.4 Parallel Decomposition

While the 2D/1D scheme greatly reduces runtime from a direct 3D transport calculation, it is still computationally expensive when compared to nodal methods traditionally used by

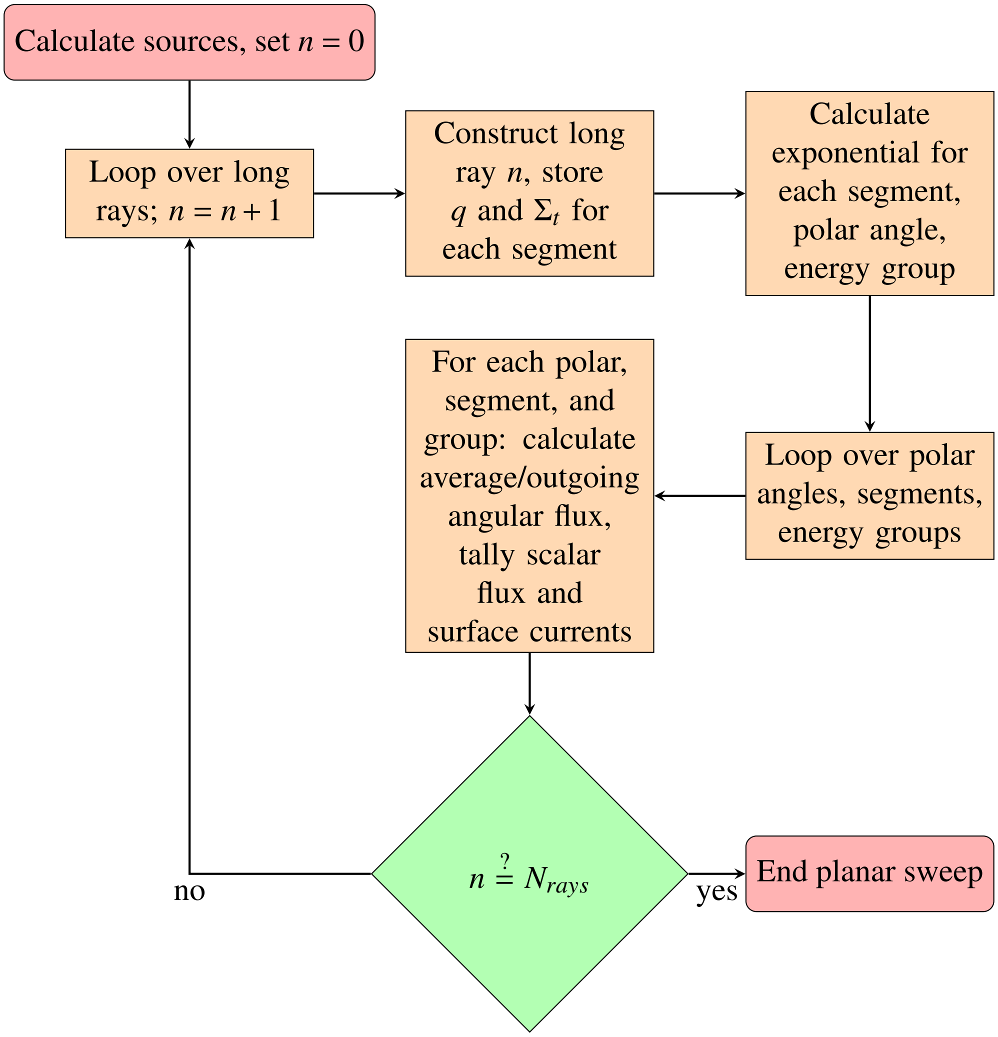

Calculate sources, set $n = 0$
Loop over long rays; $n = n + 1$
Construct long ray n, store q and Σ_t for each segment
Calculate exponential for each segment, polar angle, energy group
Loop over polar angles, segments, energy groups
For each polar, segment, and group: calculate average/outgoing angular flux, tally scalar flux and surface currents
$n \overset{?}{=} N_{rays}$
no
yes
End planar sweep

Figure 3.7: Calculation flow for 2D MOC sweep of a single plane

industry. To minimize the walltime required for 2D/1D calculations in MPACT, several different methods of decomposing the problem for parallel execution have been implemented. These methods allow MPACT to easily scale to hundreds or thousands of CPUs. Each of these methods will be briefly described in this section.

1. **Spatial Decomposition**

   When using this decomposition, each parallel process only has a portion of the model. Each portion is solved locally by one process, then boundary data is communicated to all processes which own neighboring portions of the model. The updated boundary data is then used in the following iteration. When using spatial decomposition, planar decomposition is performed first. This means that if the total number of parallel processes being used is less than or equal to the number of 2D MOC planes, then one or more entire planes is simulated by each process. If more processes are used than there are planes in the model, then radial decomposition is performed. This decomposes every plane radially into smaller pieces. Every plane must be radially decomposed in the same way, and the smallest unit allowed in radial decomposition is a single ray-tracing module. Because spatial decomposition does not duplicate much memory and does not decrease the computational efficiency significantly, it is usually the preferred choice of decomposition methods.

2. **Angle Decomposition**

   For angle decomposition, each process has the entire spatial domain. When the MOC sweeps are performed, each process only sweeps a subset of the angles in the selected quadrature. After the sweep, a reduction is performed to get the actual scalar flux and currents on all processes. For the CMFD calculation, the angle processes are repurposed as spatial processes. Each angle process owns the full domain, but only solves a portion of it as if it were spatially decomposed.

   It is possible to use both spatial and angle decomposition together. When this is done, spatial decomposition is performed first, then angle decomposition is done within each spatial domain. In general, the efficiency of angle decomposition calculations is less than that of spatial decompositions. Furthermore, it also requires that each angle process models all of the spatial domain, increasing the total memory required for the calculation compared with finer spatial decomposition. However, angle decomposition is still useful for reducing the runtime of cases where further spatial decomposition is not possible.

3. **Ray Decomposition**

   A third type of decomposition that can be done is to decompose the rays in the MOC calculation. Unlike the previous methods, the ray decomposition makes use of shared-memory threading instead of distributed memory message passing. While performing the MOC sweeps, several threads are used to solve all the rays in each angle. For the CMFD calculation, MPACT has internal RBSOR solvers which are capable of using threading. However, when third-party libraries are used for the CMFD calculations, the threading will be used only during the CMFD calculation. Threading can also be combined with both spatial and angle decomposition to further increase the parallelism of MPACT.

4. **Energy Decomposition**

   At this time, energy decomposition is not available in MPACT. However, when it is added, it will be similar to the angle decomposition. For the MOC calculation, each process will solve a subset of the energy groups on the spatial domain, and for the CMFD calculation, the energy processes will be re-purposed as space processes.

3.5 Sources of Error

The 2D/1D approximation has several sources of error. Some of these are addressed by mesh, ray spacing, or quadrature refinements, but others are due to approximations made in the method itself. The sources of error which are due to fundamental approximations in the 2D/1D method will be discussed first, followed by a brief (not comprehensive) list of some other common sources of error.

3.5.1 Axial Homogenization

When deriving the radial equations in Section 3.2.1, it was assumed that the cross sections were constant in the axial direction for each of the planes. While this is often the case if an appropriate axial mesh is selected, sometimes it is impractical to mesh the problem finely enough to ensure this. When an axial material heterogeneity is present in a plane, 2D MOC requires that these materials be homogenized. In some cases, a simple volume-averaging is sufficient, but if a material with a large cross section is being homogenized with a material that has a significantly different cross section, significant errors can result. To prevent this without refining the axial mesh, some modification to the 2D/1D scheme is required to improve the homogenization. This will be addressed in Chapter 4.

3.5.2 Axial Transverse Leakage Source

Another approximation relates to the assumptions made while deriving the axial equations in Section 3.2.2. The P_3 calculations are performed on a pin-homogenized mesh. Because of this, the axial currents used in the axial TL source are assumed to be flat across the entire pin cell. However, the currents will obviously be quite different in the fuel and moderator regions. Furthermore, the axial TL is treated isotropically. While this simplifies the axial calculations and MOC storage requirements, it does not perfectly reflect reality. Both these spatial and angular assumptions introduce some error into the axial TL source used by MOC.

3.5.3 Radial Currents

The radial TL source used by the axial P_3 solver has the same two approximations as the axial TL source did. Radial currents are used to generate the source, which assumes isotropy. Additionally, the spatial shape is flat across each pin cell. This is corrected to some extent since the axial solver produces a quadratic source shape using the neighboring nodes, but this is not a perfect solution.

Additionally, when using the subplane method, the $\hat{D}$ correction terms used by CMFD are assumed to be axially flat within each MOC plane. While this assumption improves the stability of the calculations, it forces CMFD to capture any axial shape the current has within an MOC plane. For most problems, this error is probably negligible, but for cases such as a partially inserted rod or other strong absorber, it would be beneficial to be able to have subplane-dependent $\hat{D}$ terms to more accurately calculate the radial currents. Doing this would improve the radial TL source in the P_3 solver, and the the overall 2D/1D results.

3.5.4 Other Sources of Error

Several other sources of error in the 2D/1D method will be briefly mentioned here, but not discussed in detail.

1. **Ray Spacing**

   The spacing between the rays in the MOC calculation is important to the accuracy of the calculation. At minimum, one ray needs to pass through each of the fine mesh regions, but multiple rays will improve the accuracy. A typical ray spacing is 0.05 cm, but sometimes finer spacing may be required.

2. **Radial Meshing**

   The radial and azimuthal meshing of each of the pin cells must be fine enough to give a good solution. In MPACT, fuel pins usually have 3 radial rings, with an extra ring in the moderator region to resolve the change in the flux near the edge of the fuel pin. Each ring is typically divided into 8 azimuthal regions.

3. **Axial Meshing**

   The axial mesh must be refined enough to capture the axial shape of the solution. Usually MOC planes of about 8 cm thick are used for a typical PWR calculation, which some thinner planes to resolve spacer grids, burnable poison inserts, or other components. Using thicker planes could decrease accuracy and worsen the convergence of the

CMFD calculations. This effect can be minimized for thick MOC planes by using the subplane method.

4. **Scattering Treatment**

   Scattering in a reactor, especially off the hydrogen atoms in the moderator, is anisotropic. To resolve this, a sufficiently high-order scattering treatment must be used in the MOC calculations. For PWRs, P_1 to P_3 is a typical range. MPACT is capable of going up to P_5 scattering treatment for libraries which have the required data.

   An alternative is to use transport-correction P_0 scattering. This can capture the anisotropy without increasing the runtime of the calculations. However, there are several methods of calculating the transport cross sections, and none of them are perfect. Thus, using the TCP_0 option in MPACT also has some non-trivial error associated with it.

5. **Cross-Section Library**

   To perform any calculations using the 2D/1D method, a multi-group cross section library must be available. While this is not technically a source of error in the 2D/1D method itself, the cross section library can be difficult to generate correctly. Any error in any isotope in the library will cause error in the 2D/1D calculations if the isotope is used in the model. Thus, the 2D/1D method is useless if the a bad cross section library is being used.

6. **Self-Shielding**

   Another potential source of error is related to spatial and energy self-shielding. To correctly deal with resonance absorption in the fuel while also accounting for the spatial self-shielding in the fuel, MPACT uses the subgroup method [51, 52]. Without using this method, the k_{eff} calculated by MPACT is off by several percent, along with an inaccurate flux distribution.

7. **Quadrature**

   One final source of error that can arise is in the selection of a quadrature. It is important to select both an appropriate number of azimuthal and polar angles as well as an appropriate type of quadrature. Typically around 16 azimuthal angles and 3 polar angles is sufficient. There are several different types of quadratures implemented in MPACT, but generally a Tabuchi-Yamamoto quadrature is used for the polar angles [53] with a Chebyshev quadrature for the azimuthal angles [54].

CHAPTER 4

Rod Cusping

This chapter will focus specifically on control rod cusping effects, which are the focus of this work. First, a more thorough definition of the problem and motivation for solving it will be presented. The next section will then present some of the solutions that have been used to minimize the cusping effects in the past, including a simplified decusping model implemented in MPACT itself. The third section will then discuss some newer methods based on the sublpane CMFD scheme that have recently been implemented in MPACT. Finally, a new “subray” MOC method will be proposed to deal with the cause of the cusping effects on a more fundamental level.

4.1 Background

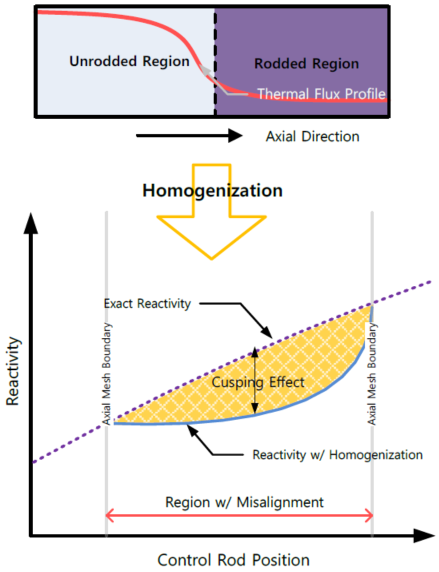


Figure 4.1: Illustration of Rod “Cusping” [23]

In Section 3.5, some potential sources of errors for the 2D/1D scheme were introduced. One of these was the error introduced by axial homogenization within a 2D MOC plane. In some cases, this can be done without introducing significant errors. For example, MPACT often homogenizes components outside the active fuel region, such as the end plugs and gaps at the end of the fuel rods. However, when strong neutron absorbers, such as control rods, are homogenized axially in active fuel region, this has the effect of introducing absorption in regions where there should be none. This effect is known as "cusping," [2] and is illustrated in Figure 4.1.

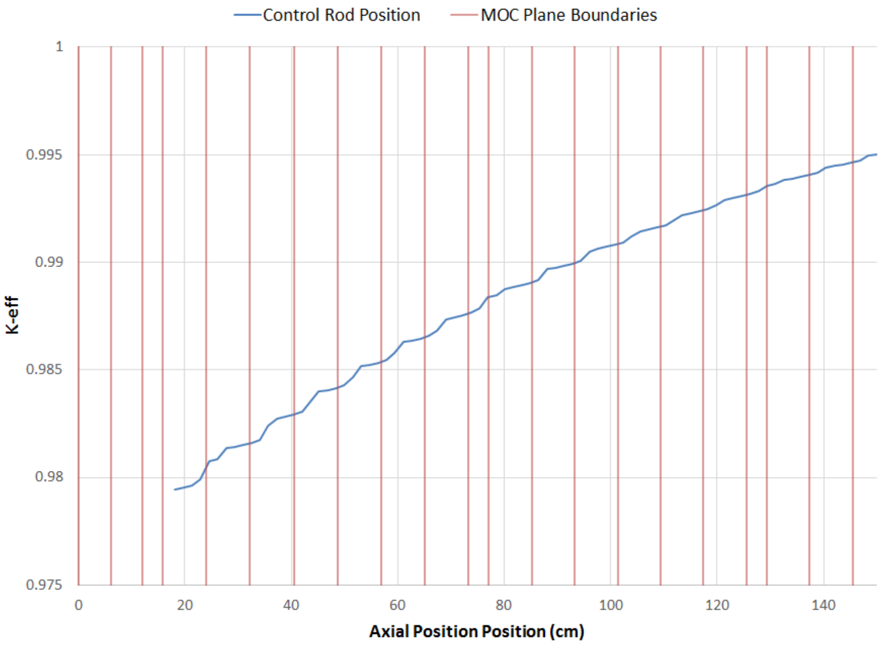


Figure 4.2: Control Rod Cusping Effects in MPACT for 3x3 Assembly

In some cases, this is easily handled by setting up an appropriate axial mesh which prevents the need for the homogenization, but this is not always a practical solution. Throughout the course of an entire cycle of operation (usually about 18 months), several different control banks in the reactor may move to a variety of positions to maintain criticality in the core. Control rods in a PWR typically have step sizes of approximately 1.5 cm, but a typical MOC plane in MPACT is about 8 cm thick in the active fuel region. In order to prevent cusping effects for an entire cycle, the user may have to create a very detailed axial mesh to ensure that all the control rod positions used throughout the cycle align with the edge of an MOC plane. Not only is this tedious for the user, but it also greatly increases the computational burden due to the increased number of MOC planes. Figure 4.2 shows the calculated k_{eff} as a function of control rod position. The cusping effects in this figure are further complicated by a heterogeneous rod with silver-indium cadmium alloy (AIC) and B_4C poison regions and a stainless steal tip. Thus, cusping effects occur not just at the control rod tip, but also at material interfaces throughout the rod.

4.2 Decusping Methods History

Before discussing the recent and proposed methods to address the rod cusping problem, it is useful to provide an overview of past methods. These will then be used as a comparison for recent work and justification for the proposed work. First, we will look at some of the ways rod cusping was address in older nodal codes. Then we will look at the effects in 2D/1D codes and recent work to address them.

4.2.1 Nodal Codes

Control rod decusping methods have been developed for nodal codes for over thirty years now. Many different methods have been developed over that time frame, making a comprehensive discussion impractical. However, several different methods will be discussed here to provide some context for how the rod cusping problem was handled in nodal codes.

4.2.1.1 Tabulation Methods

One of the most basic methods of handling the cusping problem was through pre-tabulation of cross section for the partially rodded node (PRN) [10]. For the nodal calculations, node-averaged cross sections were generated using some higher fidelity method on small portions of the domain. For the PRN, this calculation could simply be repeated many times for each of the possible rod positions. However, because of the many positions the control rod could have in a reactor, this required many different cross sections to be calculated. Furthermore, these calculations needed to be done on more than a single assembly to obtain acceptable accuracy due to inter-assembly effects [55]. Others tried using multi-assembly calculations [3] and color sets [4] to improve the accuracy of this method. However, the number of rod possible positions and the size of the problems required to generate accurate homogenized cross sections made this method impractical at the time, motivating further research into rod decusping methods.

One variation of this method which improved on the runtime used response matrices and node surface currents [5, 6]. For this method, response matrices were tabulated ahead of time based on the surface current boundary conditions. An iterative process could then be carried out between the global nodal calculation and a local fixed sources calculation.

During each iteration, the surface currents from the global calculation were used to update the homogenized cross sections in the PRN using the response matrices. At convergence, the cross sections in the PRN were calculated from the actual solution boundary conditions, greatly increasing the accuracy of the global solution compared with the simpler cross section tabulation method. However, this method still required up-front calculation of the response matrices, which could be expensive for some problems. Consistency was also not perfectly maintained between the local and global problems, which sometimes resulted in convergence difficulties.

4.2.1.2 Polynomial Flux Expansions

Another early method which sought to improve on the brute-force tabulation method used polynomial expansions of the intra-nodal flux shape in the axial direction [7, 8]. Methods such as NEM which are commonly used in nodal codes typically assume some shape to the source and flux inside each node. For this method, a quadratic shape was assumed and used to flux-volume weight the rodded and unrodded cross sections to generate homogenized cross sections for the PRN. This resulted in a reduction of the cusping errors by about 50% [56], which was not sufficient for many applications. Higher order polynomials were also attempted [9], but the axial shapes these generated were non-physical and sometimes led to oscillations and other numerical problems.

4.2.1.3 Collector-Predictor Method

Joo [10] focused on developing a method which would address rod decusping using only material compositions and node dimensions, rather than relying on boundary conditions as well. To do this, he developed an asymptotic method in which the PRN was modeled as two semi-infinite slabs, with one slab being composed of rodded material and the other slab being unrodded material. A 1D multi-group calculation was then carried out to determine the flux shape around the interface between the two materials. This flux shape was then used in the homogenization process for the PRN.

This method was found to be inaccurate because the infinite system is not actually a good representation of the flux shape in a reactor. To improve on this method, Joo developed what is known as the Collector-Predictor Method. There are two variations to this method. The first, simpler variation assumes that the first calculation has the tip of the control rod aligned with the boundary between two nodes. The collector step occurs at the end of the global calculation, collecting the radial transverse leakages and axial boundary currents for the rodded and unrodded nodes which neighbor each other. Then, rather than performing calculations using semi-infinite slabs as in the previous paragraph, a 1D calculation is instead performed using the information from the collector step. This predictor step generates a more accurate flux profile around the interface between the rodded and unrodded materials. Then, when the rod is moved throughout the calculation, it is assumed that this axial flux profile in the vicinity of the rod tip does not change significantly, so the profile can just be shifted with the rod and used in all subsequent cross section homogenizations. This method then improved to allow the first calculation to also have a PRN. Overall, this method significantly improved on previous ones by reducing decusping methods dynamically that converged consistently and did not rely on tabulated values. Furthermore, because the extra calculations were 1D on a small subdomain of the problem, the additional computational expense was small.

4.2.1.4 Bi-Linear Weighting Method

Many decusping methods which followed the Collector-Predictor Method focused on improvements to the 1D calculation or boundary condition information to improve accuracy [11–14]. However, one significant variation that was developed by Kim and Cho [15] is the Bi-Linear Weighting Method. This method uses the same concept as the Collector-Predictor method, but solves for both the forward and adjoint axial flux shapes. Both of these shapes are then used in the cross section homogenization, as shown in Equation 4.1. The use of the adjoint flux ϕ^* in the homogenization process resulted in significant improvements in the accuracy of the nodal calculations.

$$\overline{\Sigma} = \frac{\int \phi(z)\phi^*(z)\Sigma(z)\,dz}{\int \phi(z)\phi^*(z)\,dz}\,. \tag{4.1}$$

4.2.1.5 Nodal Expansion Method Modification

Another method which calculates axial flux profiles to improve homogenization is a modified form of NEM [16, 17]. To improve the accuracy of nodal calculations, NEM assumes a quartic polynomial for the intra-nodal flux shape to calculate interface currents. For the PRN, this quartic shape can can also be used to calculate improved cross sections. This is convenient because this axial profile must already be calculated when using NEM, so the only additional calculation is to integrate the shape over the rodded and unrodded region and use the result to homogenize the cross sections. Because Legendre polynomials are used for the flux expansion, these integrals can be calculated analytically, resulting in negligible increase in computational cost.

4.2.1.6 Equivalent-Node Method

The Equivalent-Node Method [18] attempts to calculate the PRN cross sections as if the PRN were modeled as two separate nodes: one fully rodded and one fully unrodded. This method sets up two different diffusion problems for the PRN: on with a single region and one with two regions. This is done in combination with NEM for each of the spatial moments being calculated. The cross sections in the case of the single region (homogenized cross sections) are formulated in terms weighting factors multiplied by the heterogeneous cross sections. It is then enforced that the integral of the reaction rates for the one- and two-region problems are equal. This preserves the reaction rates of the two-node problem in the homogenized problem and provides correction factors for the higher-order spatial moments. Solving these equations produces accurate, smoothly varying homogenized cross sections.

4.2.1.7 Inverse Spectral Index Method

With increased computing power, interest grew in performing full core calculations with finer spatial discretizations by perform nodal calculations using homogenized pins instead of homogenized assemblies or quarter assemblies. However, with the smaller spatial regions, the radial leakages become more important than before, causing many of the older rod decusping methods to become inaccurate. In 2004, Yamamoto developed the Inverse Spectral Index (ISI) method to address rod cusping for a pin-by-pin nodal calculation [19].

To develop this method, it must be noted that if the axial leakage is small compared to the radial leakage, the flux spectrum in a cell will be similar for an assembly calculation or a full core calculation. When performing assembly calculations to generate homogenized cross sections, one quantity which can be tabulated is the spectral index, defined as the ratio of the fast flux to the thermal flux in a pin. Because the fast flux is smoothly varying across the core compared to the thermal flux, it can be accurately approximated using the PRN's neighbor nodes. The spectral index from the assembly calculations, which explicitly modeled the rodded and unrodded regions, can then be used with the fast flux to calculate the thermal flux. A flux-volume weighting is then used to average the rodded and unrodded cross sections for the following iteration. Because the fast flux is affected by the PRN cross sections, iterations are performed to converge the flux and cross sections together

4.2.1.8 CIAMA Nodal Method

Recently, a new nodal method known as Channel-wise Intrinsic Axial Mesh Adaptation (CIAMA) has been developed [20, 21]. This method eliminates all rod cusping effects implicitly by eliminating the traditional constraint of other nodal methods: all nodes must be homogeneous. With this requirement removed, partially inserted control rods can be handled implicitly by the method, without need for any auxiliary calculations or corrections.

To do this, a subplane-like scheme is employed in the NEM formulation. As with a traditional NEM formulation, the nodes are coupled through transverse leakage terms. However, within each node, a refined heterogeneous axial mesh is used. The transverse leakage terms are still calculated on the coarse mesh with a quadratic polynomial fit. For each node, this polynomial can simply be integrated over the height of the subnodes to determine the subnode leakage source. Because the inter-node coupling is done on the coarse mesh, the axial submesh can be unique for each node. This allows the method to implicitly handle axial heterogeneities with minimal increases in runtime.

4.2.2 2D/1D Codes

Unfortunately, moving away from nodal methods to higher fidelity transport codes does not eliminate the rod cusping problem, as shown in Figure 4.2. There have not been as many 2D/1D codes as there have been nodal codes, but each of them still had to contend with this problem. This section will discuss some of the different 2D/1D codes that have been developed and how the rod cusping problem was dealt with in each of them.

4.2.2.1 Neighbor Spectral Index Method

The code CRX-2K [22] uses the 2D/1D fusion method to perform LWR simulations. To address the rod cusping problem in this code, a modified version of the ISI method is applied, called the Neighbor Spectral Index (NSI) method. The NSI method uses the same methodology as the ISI with one modification. Because the 2D/1D fusion method does not require standalone calculations to generate homogenized cross sections, the spectral index must be calculated on-the-fly during the 2D/1D iteration. To do this, the neighboring node above the PRN is used to obtain a rodded spectral index, and the neighboring node below the PRN is used to obtain an unrodded spectral index. These two indexes are then used with the rodded and unrodded cross sections and volume fractions to obtain homogenized cross sections for the PRN. This method causes significant improvements over other decusping methods and requires negligible additional computation since the fluxes are stored for every region already.

4.2.2.2 Subplane Decusping

Another 2D/1D code is nTRACER, which is under active development by Seoul National University [49]. To address rod cusping effects in nTRACER, Jung and Joo developed a more general method than the polynomial correction method used by MPACT [23]. This method pregenerates correction factors at the start of a simulation, rather than relying on hard-coded corrections. To do this, the assembly that will have a partially inserted control rod is identified, and a single-plane 3x3 assembly problem is set up using the partially rodded assembly and its neighbors. The radial and axial cusping effects are then determined separately. First, the radial cusping effects are determined by performing 2D MOC calculations on the 3x3 subdomain with the rod fully inserted and fully withdrawn. This provides radial flux profiles in the rodded assembly for both rodded and unrodded regions, as well as current coupling coefficients for CMFD for the rodded and unrodded CMFD nodes. Once this is done, the rod is simulated at positions of 25%, 50%, and 75% withdrawn from the plane. To reduce runtime, these calculations are done using only 3D subplane CMFD, using the heterogeneous rodded and unrodded cross section for the appropriate subplanes. This generates axial flux profiles for the full MOC plane for each of the possible rod positions. During the full core 2D/1D calculation, these axial flux profiles are then used to generate improved homogenized cross sections for the MOC calculation using equation.

$$\overline{\Sigma_i} = \frac{\phi^R_{rad,i}\phi^R_{ax,i}\Sigma^R_i h^R + \phi^U_{rad,i}\phi^U_{ax,i}\Sigma^U_i h^U}{\phi^R_{rad,i}\phi^R_{ax,i}h^R + \phi^U_{rad,i}\phi^U_{ax,i}h^U} \,. \tag{4.2}$$

4.2.2.3 Approximate Flux Weighting Method

Because nTRACER's subplane decusping method required up-front calculations for use, the total runtime of the 2D/1D calculations increased significantly. Because of this, the Approximate Flux Weighting Method was implemented [24]. This method was originally developed for nodal methods [11], but also proved effective for 2D/1D as well. In this method, it is assumed that the flux in the rodded and unrodded parts of the partially rodded node are close to the fluxes in the rodded node above and unrodded node below, respectively. This assumption is not far from reality, and it provides approximate values for the flux that can be used to homogenize the cross sections in the partially rodded node. This method is popular in a variety of nodal codes because it is relatively accurate and quite simple to implement.

4.3 Summary

The rod cusping problem has been shown to occur in both nodal codes and direct whole-core transport codes. Many methods have been developed over the years to address cusping. Many of these methods are deal with tabulating cross sections for nodal codes, and are therefore irrelevant for more modern planar synthesis methods. Others have either failed in fully eliminating the effects of the partially inserted rod or require enough additional calculation that they were abandoned in favor of faster methods, as seen with the nTRACER code in Section 4.2.2.3. Because of this, research into advanced, novel subgrid methods to deal with this problem is still required. The following chapter will present three methods which attempt to deal with rod cusping in varying degrees of complexity.

CHAPTER 5

Subgrid Decusping Methods

This chapter will describe three control rod decusping methods developed as part of this work. The first method is polynomial decusping, a simple correction based on pregenerated data. The second method is subplane collision probabilities. This method extends the subplane scheme by introducing a 1D collision probabilities calculation to capture subgrid information around partially inserted rods. The final method presented here is the subray method of characteristics. This modified form of MOC directly accounts for the partially inserted rod in the 2D MOC calculations instead of applying corrections to cross sections afterwards.

5.1 Polynomial Decusping

The polynomial decusping technique was developed to provide a fast, simple correction to the cusping problem in the MPACT code. This technique assumes that the reactivity and power around the partially inserted rod have a predictable shape as functions of the rod position within the MOC plane. Based on this assumption, correction factors were developed to reduce the volume fraction of control rod material and reduce the magnitude of the cusping effects.

5.1.1 Correction Data Generation

To generate this data, two different sets of calculations were required. Both sets were done using a 3×3 assembly case with a control rod in the center assembly. The first set of calculations began with the control rod aligned with MOC plane boundaries. The rod was then withdrawn upward through the MOC plane and simulated at different positions. A total of 9 simulations were conducted: one with the rod aligned with the bottom of the plane, one with the rod aligned with the top of the plane, and 7 with the rod partially inserted to various depths in the plane. The first and last calculations had no cusping effects because the rod was aligned with plane boundaries, but the remaining cases exhibited severe cusping effects.

The second set of calculations used the same problem and rod positions as the first set. However, for the 7 partially inserted calculations, an extra MOC plane boundary was added that aligned with the control rod tip. This eliminated rod cusping effects for the second set of calculations. These two sets can then be compared to determine the magnitude of the cusping effects for each rod position by plotting k_{eff} against rod position.

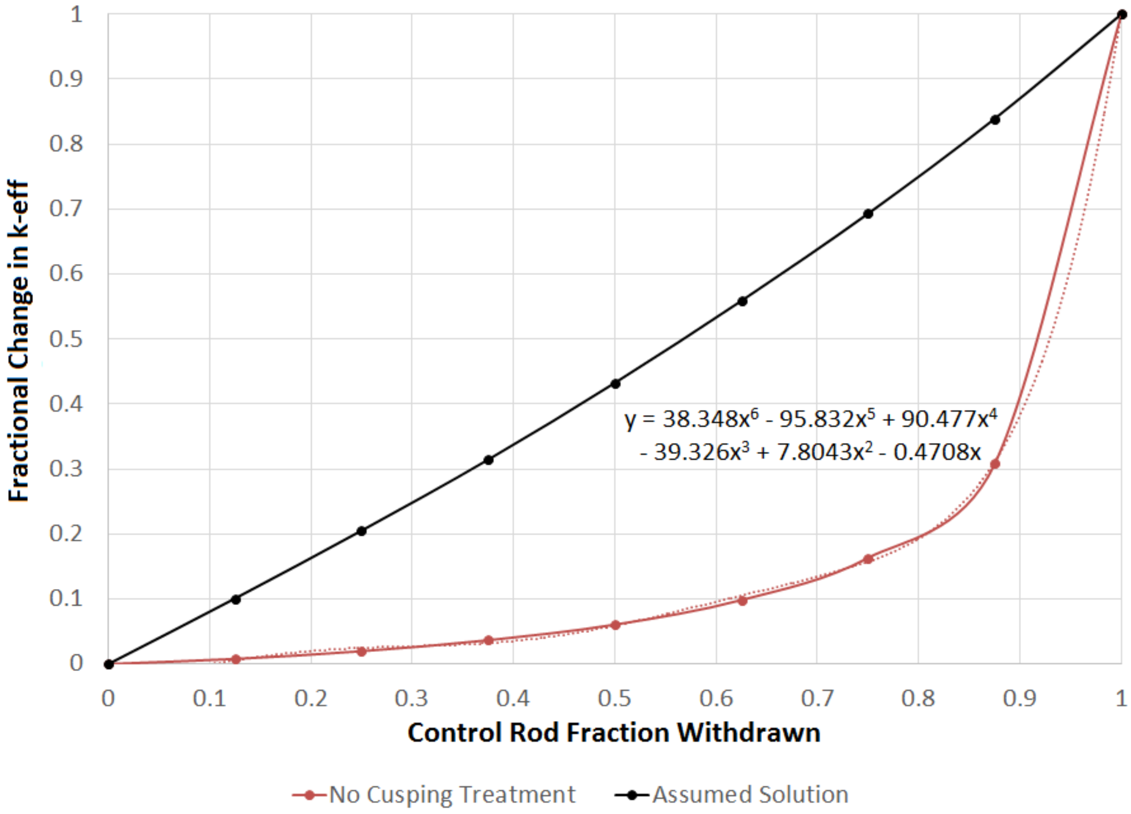


Figure 5.1: Polynomial decusping example data generation

Since these data need to be used to correct rod cusping for different MOC planes and reactors, the data is best plotted as a percent change in k_{eff} versus a percent rod withdrawal. An example of this is shown in Figure 5.1. The data from these calculations was then used to generate a sixth-order polynomial that approximates the shape of the rod cusping effects. Generating this polynomial in terms of percents allows the polynomial to approximate the response for any MOC plane in any reactor. However, the response varies significantly for different rod materials, so this process should be repeated for each unique type of control rod. For the work presented here, polynomials were generated for three different materials:

silver-indium-cadmium alloy (AIC), B$_4$C, and tungsten. The polynomials developed for each material are shown below:

$$P_{AIC}(V_u) = 38.348V_u^6 - 95.832V_u^5 + 90.477V_u^4 - 39.326V_u^3 + 7.8043V_u^2 - 0.4708V_u \ , \tag{5.1}$$

$$P_{B4C}(V_u) = 46.843V_u^6 - 117.14V_u^5 + 109.78V_u^4 - 47.104V_u^3 + 9.1734V_u^2 - 0.5527V_u \ , \tag{5.2}$$

$$P_{TUN}(V_u) = 4.7734V_u^6 - 14.751V_u^5 + 19.512V_u^4 - 12.762V_u^3 + 4.0769V_u^2 - 0.1505V_u \ , \tag{5.3}$$

where V_u is the unrodded volume fraction and $P_x(V_u)$ is the corresponding percent change in k_{eff} for a given rod type, with x being AIC, $B4C$, or TUN.

5.1.2 Correction Application

Following the procedure in the previous section gives polynomials to approximate the shape of the cusping effects for different rod materials. To apply these polynomials to 2D/1D calculations, the volume fractions used in the homogenization prior to 2D/1D calculations can be modified. This homogenization is done using a simple volume weighting in the partially rodded regions:

$$\Sigma_x = V_u \Sigma_x^U + V_r \Sigma_x^r \ , \tag{5.4}$$

where Σ_x is the cross section being homogenized, V is the volume fraction, u and r subscripts and superscripts denote rodded and unrodded quatities, respectively, and $V_u + V_r = 1$. It is evident from Equation 5.1 that cusping always lowers the value of k_{eff}, so if V_u is increased and V_r is decreased, the rod cusping errors should be reduced.

At the beginning of the calculation, the volume fraction of the control rod is determined. For example, if an AIC rod is 50% inserted into a plane, the data from Figure 5.1 can be used to determine the expected percent change in k_{eff}. In MPACT, the approximation is made that the reference solution in Figure 5.1 is a straight line, resulting in a 50% change in k_{eff} from a 50% inserted rod. Next, the point along the polynomial which corresponds to 50% change in k_{eff} is identified. Finally, the volume fraction which produces that value of the polynomial is used in the homogenization. For this example, values of $V_u = 0.5$ and $V_r = 0.5$ would be used for a 50-50 mixture of moderator and AIC control rod without any decusping method. To improve this solution, the model actually used would use values of approximately $V_u = 0.93$ and $V_r = 0.07$ to minimize the error in k_{eff}.

To find the volume fraction required for improved homogenization, a root-finding method must be applied. For the example in the previous paragraph, MPACT does this by setting $P_{AIC}(V_u) = 0.5$ in Equation 5.1 and moving it to the right-hand side of the equation. Because these polynomials are smooth, monotonically increasing on the interval [0,1], and have analytic derivatives that can be easily programmed in MPACT, Newton's method [57] can be easily applied to obtain the root, though other methods could be used as well. The search is performed until the value is within 0.001 of the actual root; usually this requires only 4-5 iterations, meaning that these calculations are trivially fast compared to 2D/1D.

5.2 Subplane Collision Probabilities

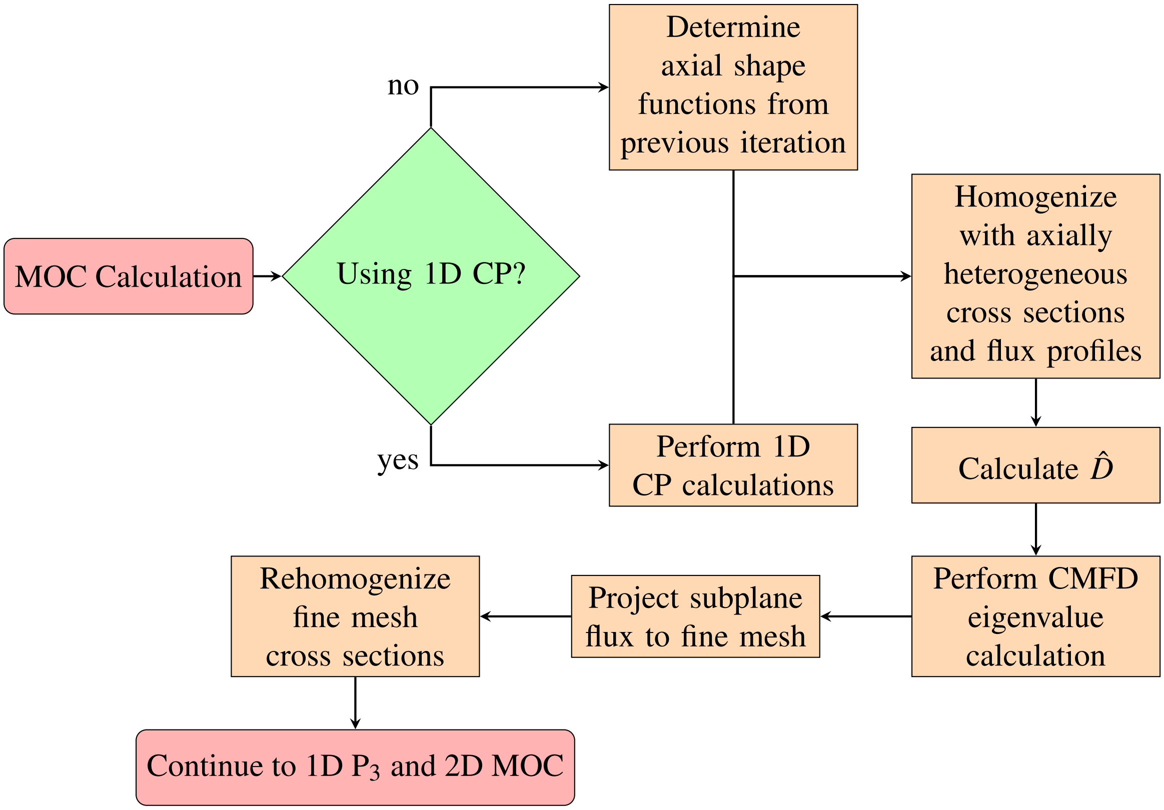

MOC Calculation
Using 1D CP?
no
yes
Determine axial shape functions from previous iteration
Perform 1D CP calculations
Homogenize with axially heterogeneous cross sections and flux profiles
Calculate $\hat{D}$
Perform CMFD eigenvalue calculation
Project subplane flux to fine mesh
Rehomogenize fine mesh cross sections
Continue to 1D P_3 and 2D MOC

Figure 5.2: Calculation flow for 2D/1D with subplane collision probabilities

The subplane scheme as it was originally conceived was used primarily to address stability issues caused by thin MOC planes in early 2D/1D codes. While other improvements to the 2D/1D method have largely eliminated these problems, the idea of capturing subplane information using the subplane scheme can be useful for addressing partially inserted rods without substantially increasing the computational cost. To do this, three changes were made to the basic subplane scheme: an axial correction performed during CMFD homogenization, an optional radial correction to generate improved flux profiles, and an MOC cross section correction. A flow chart of this process is shown in Figure 5.2

5.2.1 Axial Correction

Traditionally, the subplane scheme uses axially constant cross sections for all subplanes in each MOC plane. When a control rod is partially inserted in the plane, a flux-volume homogenized cross section is calculated and used for each subplane. This is done by applying Equations 2.35 usnig the axially homogenized cross sections from the MOC calculations. This allows the subplane scheme to be used, but does little to account for the partially inserted rod.

To resolve this issue, the subplanes with the control rod use the rod cross section, and the subplane without the rod use the moderator cross section. These cross sections are still homogenized using flux-volume weighting as before, but without homogenizing axially. Thus, Equations 2.35 are used as before, but with the unhomogenized control rod or moderator cross sections instead of the homogenized MOC cross sections. Doing this allows both the CMFD and P_3 calculations to capture some of axial effects of the partially inserted rod, reducing the magnitude of the cusping errors around the rod.

5.2.2 Radial Correction

Using axially heterogeneous cross sections within an MOC plane corrects some of the cusping effects, but it does not accurately capture the radial effects of the partially inserted rod. In reality, the radial flux shape in the rodded subplanes is completely different from the shape in the unrodded subplanes. The MOC calculations are done on the thicker MOC planes using axially homogenized cross sections in the partially rodded regions. This produces a radial flux shape that is not representative of either the rodded or unrodded region. Thus, using this radial flux shape to calculate the pin-homogenized cross sections for CMFD P_3 introduces some error in the cross sections.

To correct the radial effects, 1D collision probabilities calculations can be used as illustrated in Figure 5.3. As described in Section 2.2.4, the CP method is used to generate flux profiles for pin cell calculations. The MOC calculations are done using axially homogenized cross sections which produce incorrect radial flux profiles. Thus, after the MOC calculations and before the CMFD homogenization step, standalone CP calculations can be set up to generate new radial flux profiles. For a partially inserted rod, two calculations will be set up: one for the rodded portion and one for the unrodded portion. Each calcu-

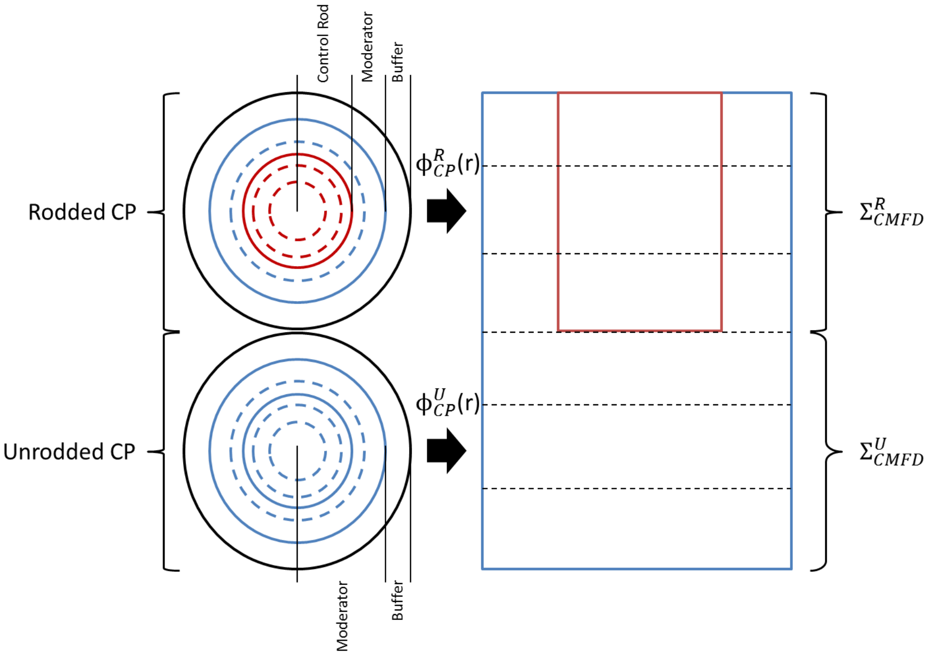


Figure 5.3: Illustration of subplane collision probabilities used on a partially inserted rod

lation will use heterogeneous cross sections to generate a radial flux profile for the rodded or unrodded axial region. These flux profiles are then used in place of the MOC flux when performing the homogenization described in the previous section.

To set up the problem, the pin cell is cylindricized to preserve area. The control rod and guide tube rings remain the same, but the outermost moderator region must be transformed from a rectangle to a ring. This is done using the following equation:

$$R_N = \sqrt{R_{N-1}^2 + \frac{L^2}{\pi}}\,, \tag{5.5}$$

where R_N is the radius of the cylindricized rectangular moderator region, R_{N-1} is the radius of the outermost ring prior to cylindricizing, and L is the length of one side of the pin cell (assuming a square pin cell). Each ring then uses the heterogeneous cross sections corresponding to the axial level being set up. For the partially rodded regions, either rodded or unrodded cross sections will be used; for the remaining regions, the cross section will be the same for each axial level. Finally, in order to drive the CP problem to a physical solution, a buffer region must be set up. This is done by taking the 8 neighboring pin cells and homogenizing their sources and cross sections into an additional ring outside the final moderator ring. This done by applying Equation 2.35 to all 8 neighboring pin cells to generate homogenized cross sections for a single region. A variation of Equation 5.5 is then applied to the buffer region:

$$R_{N+1} = \sqrt{\frac{8L^2}{\pi} + R_N^2}\,, \tag{5.6}$$

where R_{N+1} is the radius of the cylindricized buffer region and the 8 neighboring pin cells used for the buffer regions have the same dimensions L as the partially rodded pin cell.

The matrix for each of these calculations is a dense $N_r \times N_r$ matrix as described in Appendix A, where N_r is around 10 or less. It is dense because it describes the probability of neutrons born in any cell havnig a collision in every cell including itself, resulting in a completely dense matrix. This matrix must be set up for each axial level of interest (in most cases 2), each energy group, and each partially inserted rodlet (24 per rodded assembly for the results presented in Chapter 5). Each matrix must be inverted and multiplied by a source vector. Because the matrices are so small, these calculates are negligible compared to the cost of the full 2D/1D calculation. Thus, the data used during homogenization can be greatly improved without incurring a significante runtime penalty.

5.2.3 MOC Correction

The final step in this decusping technique is to use the subplane information from the CMFD/P$_3$ calculations to improve the MOC calculations. To do this, the volume-homogenized cross sections are updated each iteration using a flux-volume weighting that involves the axial flux shape from the CMFD/P$_3$ system and the radial flux shape from the CP calculations. These are combined the same way as the nTRACER method using Equation 4.2, repeated here:

$$\overline{\Sigma_i} = \frac{\phi^R_{rad,i}\phi^R_{ax,i}\Sigma^R_i h^R + \phi^U_{rad,i}\phi^U_{ax,i}\Sigma^U_i h^U}{\phi^R_{rad,i}\phi^R_{ax,i}h^R + \phi^U_{rad,i}\phi^U_{ax,i}h^U}\,,$$

where Σ_i is a cross section for radial ring i, $\phi_{rad,i}$ is the radial flux shape for ring i obtained from the 1D CP calculations, $\phi_{ax,i}$ is the axial flux shape for the pin cell, and superscript R and U indicates a value take from either the rodded or unrodded region. This equation can be applied to any number of subplanes for a given MOC plane, but is shown here for the simplest and most common instance of one rodded subplane and one unrodded subplane.

It is also possible to use only the axial correction and not the radial correction. In this case, $\phi^R_{rad,i} = \phi^U_{rad,i}$ for each radial ring because the MOC flux must be used in place of the pair of 1D CP flux shapes. This introduces some axial correction into the homogenized MOC cross sections, but no radial correction. Chapter 5 will show results for "Subplane" (axial only) and “Subplane + CP” (axial and radial) to show the importance of each of these effects for different rod types.

5.3 Subray Method of Characteristics

The previous two methods are each effective in reducing the effects of rod cusping, as will be shown in Chapter 5. Despite these improvements, each method has significant drawbacks:

- **Polynomial Decusping**
  - Corrections are limited to certain control rod materials (AIC, B_4C, Tungsten).
  - Limited accuracy, especially for problems significantly different from those used to generate data.
- **Subplane Collision Probabilities**
  - CP is limited to TCP_0 scattering.
  - CP implementation for non-cylindrical geometries can be complicated (e.g., BWR control blades).
  - Instability can be introduced in the CP calculations by large transport corrections.
  - MOC calculations are still performed using homogenized cross sections.

The most important inaccuracy in the subplane CP method is the final point. To fully address the partially inserted rod, it is desirable to account for the rod in the 2D MOC calculations themselves using heterogeneous cross sections. Doing so will greatly improve the overall accuracy of the 2D/1D calculation by improving the accuracy of the transport itself.

5.3.1 1D MOC Code and Problem Description

To develop this new method, a 1D MOC code was developed to investigate the behavior of angular and scalar flux near a partially inserted control rod. This code was written in Matlab [58] and is set up to take in a description of pins and materials to be used for the calculations. For the geometry, a pin pitch is specified which is used for all pins. Each pin consists of a list of radii. Assuming a square pin cell, these pins are then transformed from cylindrical geometry to slab geometry while preserving the volume fraction of each material:

$$w = \frac{\pi\left(R_N^2 - R_{N-1}^2\right)}{2L} . \tag{5.7}$$

The pin pitch is the same in slab geometry as in cylindrical geometry. Each ring is then transformed into two slabs using Equation 5.7, where w is the width of each slab. In Equation 5.7, R_N and R_{N-1} are the outer and inner radii of each ring, with $R_0 = 0$ for the center region in the pin cell, and L is the pin pitch. Thus, the thickness of the slabs is slightly less than the radius of each region because the volume fraction is preserved for each pin.

Each region in the problem is divided into subregions to ensure the solution is mesh-converged. Each region is then assigned a material from a cross section library file. This file uses the "user library" format supported by MPACT, which allows the user to input macroscopic cross sections for absorption, nu-fission, kappa-fission, chi, and scattering moments. For all these calculations, the C5G7 benchmark cross sections [59, 60] were used. These cross sections are included in Appendix B.

For the MOC sweeps, a Gaussian quadrature [54] is used with 2, 4, 8, 16, or 32 polar angles, with half of the angles being used in each direction. The MOC sweeps are done similarly to how they are done in MPACT, with the loop over energy groups being the innermost loop. The code can be run as either a fixed source solver or an eigenvalue solver. For the eigenvalue mode, power iteration is used after each MOC calculation to determine and updated k_{eff}. The fixed source mode can be used to run either a specified number of iterations or to run until the scattering source is converged below some tolerance. This allows some flexibility on exactly what kinds of results can be obtained.

The problem used for these calculations was a 1D variation of VERA Problem 4, illustrated in Figure 5.4. The center row of pins across all three assemblies was pulled out and used for the 1D model, resulting in a row of 51 pins (17 pins across, 3 assemblies) with a pin pitch of 1.26 cm (the inter-assembly gap was neglected for this model). The center assembly had 4 guide tubes in it which contained a mixture of moderator and control rod to represent a partially inserted control rod. These partially rodded locations were the only part of the problem that had any material changes. This allowed the effects of the cross section homogenization to be isolated for each calculations.

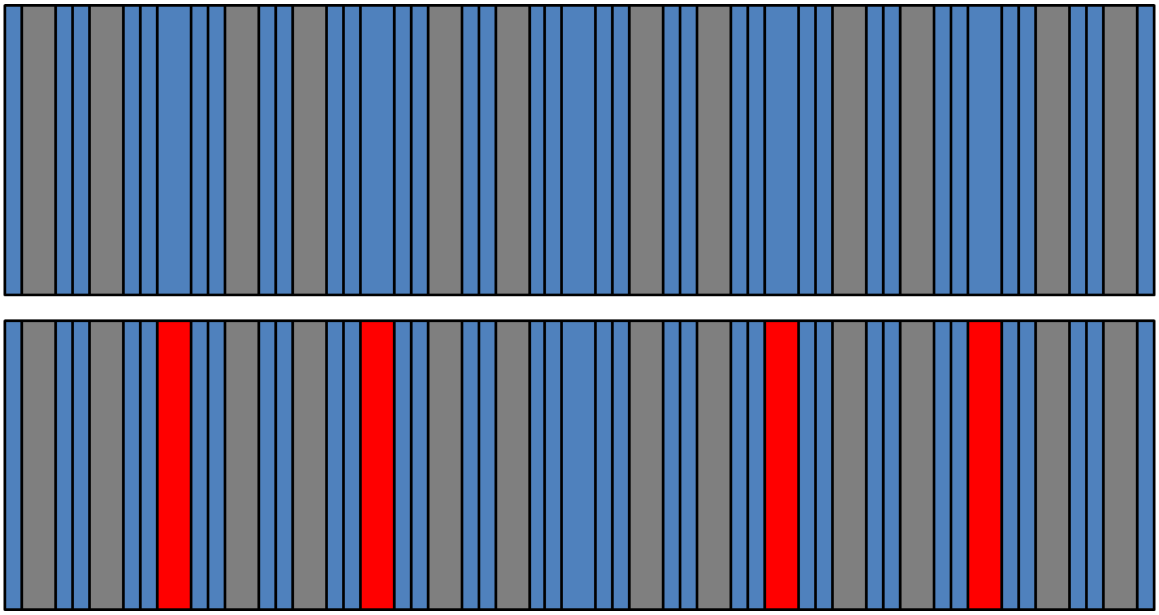


Figure 5.4: Illustration of 1D MOC 3×17 geometry, where the top part represents the left and right assemblies and the bottom part represents the center assembly, with materials fuel (grey), control rod/moderator mixture (red), and moderator (blue).

5.3.2 1D MOC Investigation

5.3.2.1 Specified Total Source

The first set of calculations performed were done using a specified total source. To do this, the guide tubes were filled with 50% control rod and 50% moderator volume fraction mixture, and a full eigenvalue calculation was performed. The total source distribution (fission and scattering) from this calculation were then passed to the the fixed source solver. A single iteration was run using this source on three different variations of the problem: the 50-50 mixture, fully rodded, and fully unrodded. Because the multi-group source is set up before performing any MOC sweeps, this resulted in all three of those calculations having an identical source for the MOC sweep. The only difference between them was the cross sections used in the guide tubes.

Figure 5.5 shows the scalar flux resulting from these three calculations. The most important thing to note in this data is that the effects of the rod are very local in the MOC calculation. Moving through the rodded pin cell, the rodded, unrodded, and mixed cases have converged back to the same shape by the time the edge of the partially rodded pin cell is reached. Because of the exponential nature of the MOC solution in Equation 2.33b, if the sources are the same the two different solutions converge to each other quickly as you move away from the rod.

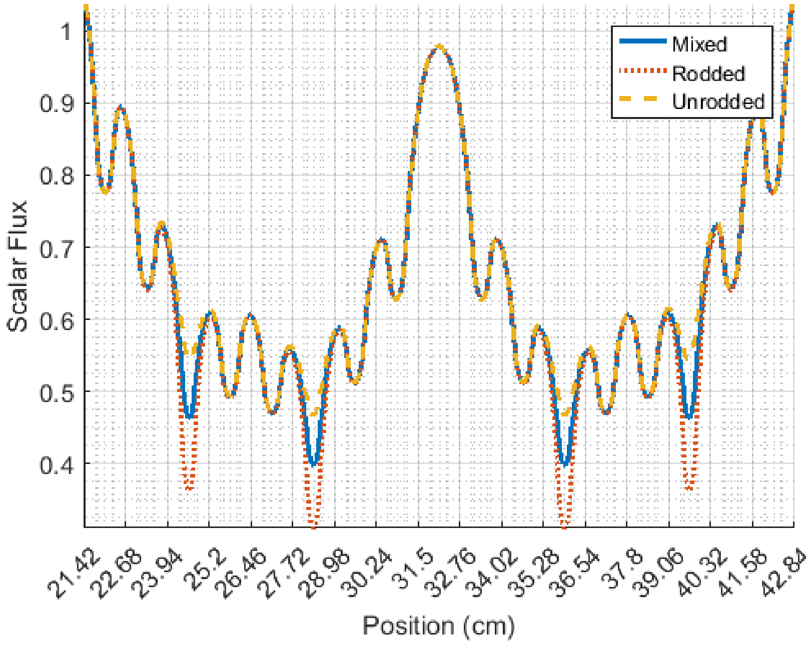
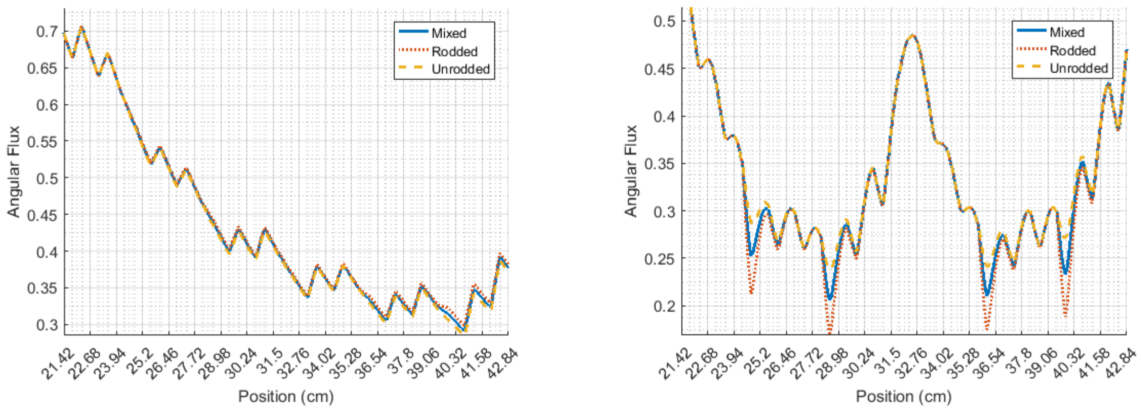


Figure 5.5: Group 7 scalar flux comparisons for a fixed fission and scattering source calculation

Figure 5.6 shows the right-going angular flux in groups 1 (fast) and group 7 (thermal) for the center assembly using the flattest polar angle $\mu = 0.997263861849482$, where μ is the cosine of the polar angle. This angle is shown because it has the most significant differences between the different cases; the steeper angles require fewer regions to converge to each other since the effective length of the MOC track is longer. The group 7 angular flux behaves similarly to the group 7 scalar flux in that the effects are localized around each rod. The three different angular flux shapes are still somewhat different at the edge of the neighboring pin cell, but these differences are nearly indiscernible upon reaching the clad and fuel in the next pin cell. The reason for this is that the mean free path of thermal neutrons is small. The total group 7 cross section in the moderator is about 2.65 cm^{-1}, which corresponds to a mean free path of about 0.38 cm, which is less than one third of the pin pitch for a typical PWR. This combined with the exponential behavior of the MOC solution washes out the differences between the solutions quickly upon moving away from the rodded region.



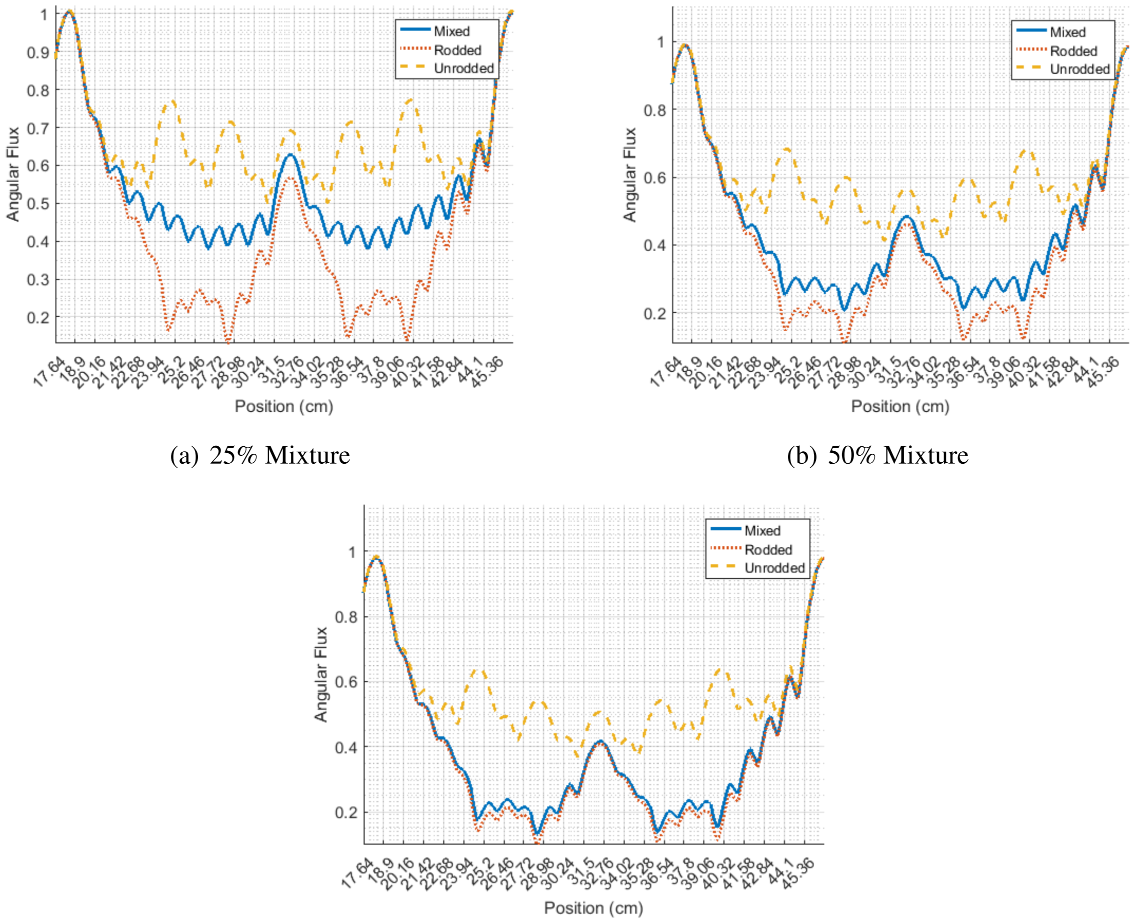
(a) Group 1

(b) Group 7

Figure 5.6: Angular flux comparisons for a fixed fission and scattering source calculation

The same cannot be said for the fast flux. The mean free path of the fast flux is about 6.3 cm, which is the width of five pin cells. Thus, it can be seen that each of the control rods after the first builds on the effects of the previous control rod. Thus, the largest differences between the solutions is seen near the exit of the assembly after passing through all 4 rodded regions. However, the differences are still small since control rods are generally thermal absorbers and have only a small impact on the flux in fast energy groups.

5.3.2.2 Fixed Fission Source



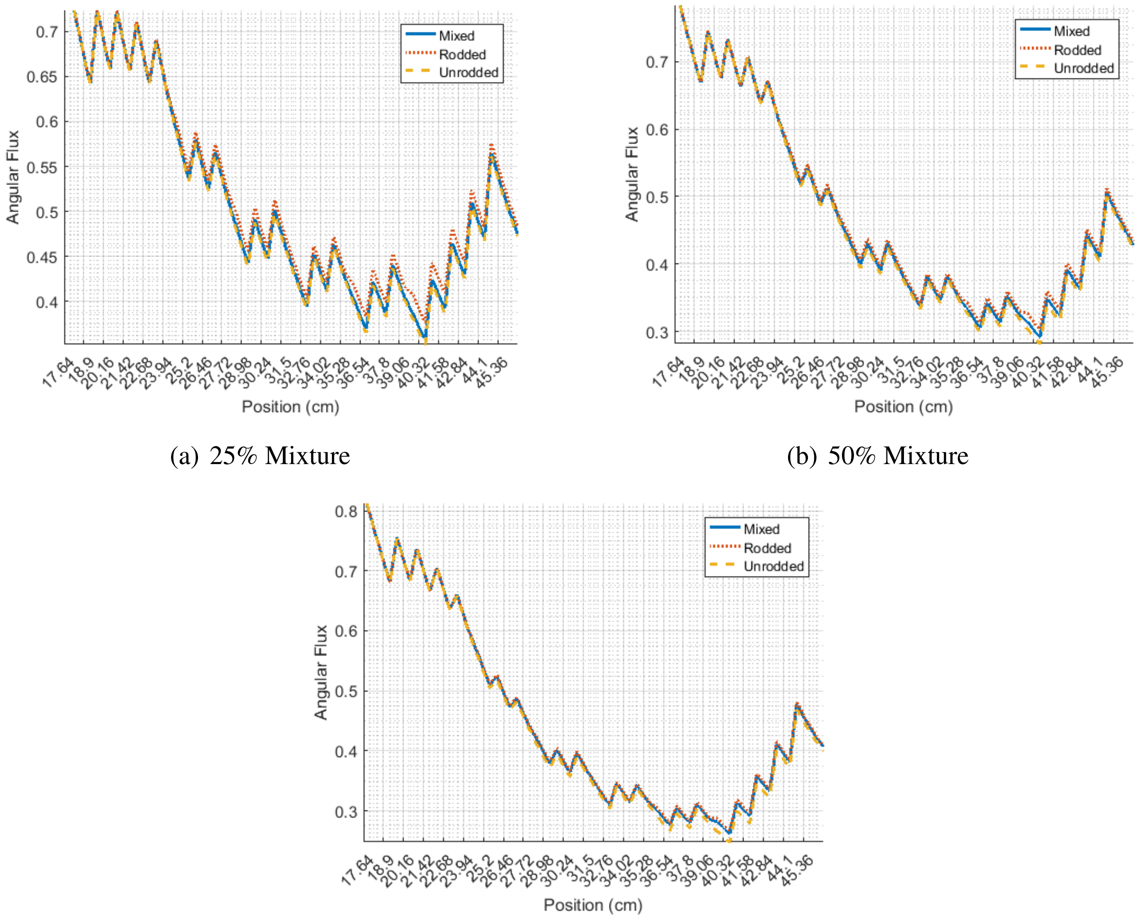
(a) 25% Mixture

(b) 50% Mixture

(c) 75% Mixture

Figure 5.7: Group 7 angular flux comparisons for 25% and 75% mixtures

The second set of calculations that was performed used a fixed fission source, but allowed the scattering source to fully converge for each calculation. As in the previous section, an eigenvalue calculation was completed using partially rodded cross sections. This was done for 25%, 50%, and 75% rodded cases. For each case, a fixed fission source calculation was done with the fully rodded and fully unrodded cross sections. This time, multiple iterations were allowed for each material to converge the scattering source. This allows us to see the effects of the rod on the scattering source distribution without worrying about changes in the eigenvalue and fission source distribution.



(a) 25% Mixture

(b) 50% Mixture

(c) 75% Mixture

Figure 5.8: Group 1 angular flux comparisons for 25% and 75% mixtures

Figure 5.7 shows the right-going group 7 angular flux comparisons for each of the three mixtures, again for the flattest value of μ. We immediately see that for each of them, the angular flux for the mixture is skewed toward the rodded result instead of being a volume-fraction–weighted average of the rodded and unrodded solutions. For example, comparing Figures 5.7(b) and 5.6(b) shows that the angular flux is much closer to the rodded solution than when the scattering source was fixed, despite the small mean free path of thermal neutrons. Figure 5.8 shows the same comparison for the fast flux and provides some insight into the differences in the thermal flux solutions. The difference between the rodded, unrodded, and mixed fast fluxes are small for this calculation, similarly to the fixed total source calculations in the previous section. However, these small differences have the effect of spreading the scattering source across the problem due to the long mean free path. Thus, the small differences in the fast flux cause large differences in the thermal flux scattering source, which in turns effects the fission source throughout the problem.

5.3.2.3 Summary

These calculations provide several insights into what a new decusping method must be able to accomplish. First, it must be able to account for the heterogeneous rodded and unrodded cross sections. This is critical to obtaining an accurate shape for the thermal flux. Second, sources must be calculated around the partially inserted rod using different scalar fluxes. If the same scalar flux is used for both the rodded and unrodded calculations, the results are non-sensical. Finally, the angular flux exiting the partially rodded region must accurately represent the partially inserted rod, especially for the fast flux. If this outgoing flux is skewed incorrectly toward the rodded or unrodded fast flux solution, even small errors can have a significant impact on the scattering and fission source distributions throughout the rest of the problem.

5.3.3 Subray MOC Description

Using the results of the 1D MOC investigation described above, a new method has been developed called the subray methods of characteristics, illustrated in Figure 5.9. For the majority of a domain, this method will behave the same as traditional MOC. However, when approaching a region with a partially inserted rod, the ray along which the angular flux is being solved can be split into two (or more) separate rays. One ray will use the control rod cross section while the other will use the cross section for the material below the control rod. The ray can be split into as many subrays as necessary to handle complicated rods or other heterogeneities, but the most common use case will split into just two subrays. In each region where subray MOC is used, the scalar flux can be tallied using a modified form of Equation 2.33b:

$$\overline{\varphi}_{g,i,n,j} = \sum_{z=1}^{Z} V_z \left(\frac{q_{g,n,i,z}}{\Sigma_{t,g,i,z}} + \frac{1}{A_j \Sigma_{t,g,i,z}} \left(\varphi^{in}_{g,i,n,j,z} - \frac{q_{g,n,i,z}}{\Sigma_{t,g,i,z}} \right) \left(1 - e^{-\Sigma_{t,g,i,z} A_j} \right) \right) , \tag{5.8}$$

where Z is the number of subrays used in region i and V_z is the volume fraction associated with subray level z. Other subscripts in this equation are g for energy group, i for region index, n for angle index, and j for MOC ray index. If the previous region also used subray, $\varphi^{in}_{g,i,n,j,z}$ is unique for each subray; otherwise, it is the same. This equation averages the angular flux for each subray segment to obtain an axially-averaged, segment-averaged angular flux, which is then used to calculate the scalar flux. Furthermore, the subray angular fluxes can also be used to calculate the scalar flux for these subregions to generate subregion source terms for the subsequent iteration.

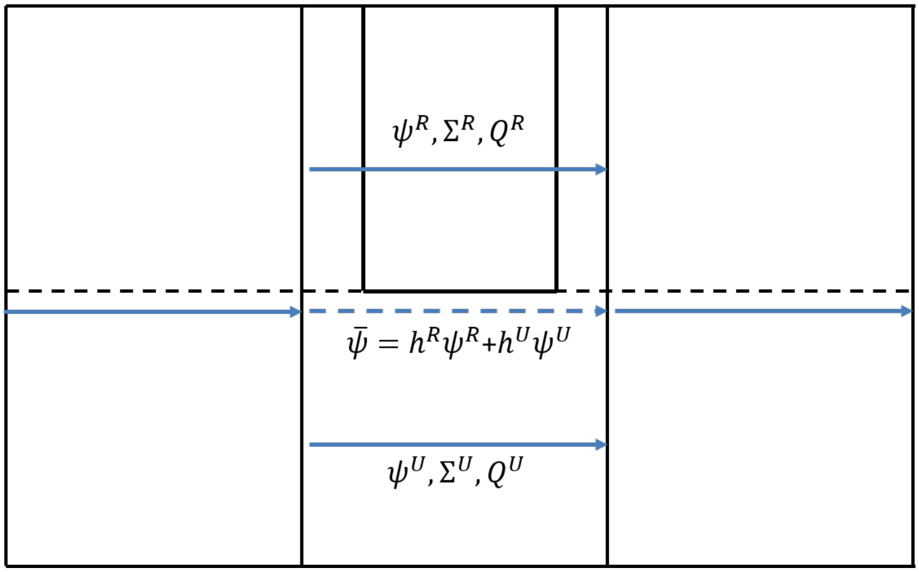


Figure 5.9: Illustration of subray method of characteristics for partially inserted rod

After passing completely through the partially rodded regions, the subray angular fluxes can be recombined into a single ray using the volume fractions associated with each subray:

$$\varphi_{g,i,n,j}^{out} = \sum_{z=1}^{Z} V_z \left(\varphi_{g,i,n,j,z}^{in} e^{-\Sigma_{t,g,i,z} A_j} + \frac{q_{g,i,n,z}}{\Sigma_{t,g,i,z}} \left(1 - e^{-\Sigma_{t,g,i,z} A_j} \right) \right) , \tag{5.9}$$

where $\varphi_{g,i,n,j}^{out}$ is the axially averaged outgoing angular flux at the end of track j for direction n, region i, and energy group g. The errors introduced from this are minimal due to the exponential nature of the MOC ray calculations. After the rod, both subrays have the same cross section again and the axial shape of the sources begins to flatten out. Because of this, the rays then exponentially converge toward each other. Thus, as long as an appropriate amount of distance is left between the end of the partially rodded region and the recombination point, minimal errors are introduced. This is discussed in more detail in Section 5.3.4.2.

This method should accurately capture the cross section and source effects around the partially inserted rod by using subrays. Furthermore, recombining the outgoing fluxes after the rod should provide a more accurate outgoing flux than that calculated from a homogenized cross section. This method addresses all the insights gathered from the 1D MOC investigation. Furthermore, it addresses all the limitations of the polynomial and subplane collision probabilities decusping techniques discussed earlier in this chapter.

5.3.4 MPACT Implementation

Implementation of subray MOC in the 1D MOC code is straightforward since the code is small and uses only power iteration. Implementing the method in the 2D/1D code MPACT is significantly more complicated. First, even for single-plane 2D cases, MOC calculations can be expensive. Because of this, the MOC sweep kernels need to be modified to use subray in such a way that the kernels will still be well optimized for the subray calculations. When moving from 2D to 2D/1D other complexities are introduced from the 3D CMFD and 1D P_3 calculations that have to be correctly incorporated in the subray MOC calculations. The following sections discuss some of the details of how this is done.

5.3.4.1 MOC Sweeper Modifications

Implementing a subray MOC sweeper in MPACT presents some difficulties since MPACT was not designed with these types of subgrid methods in mind. The MOC sweepers have been optimized to ensure that the rays are swept in the most efficient way possible. The long rays are set up once up front, then they are swept in a separate loop with no branching statements to ensure the hardware is performing the calculations as quickly as possible. Subray MOC naturally introduces branching statements that would destroy this efficiency.

To get around this, an extra loop was added just inside the loop over long rays. This is illustrated in Figure 5.10, which is a modified form of Figure 3.7. This loop goes from one to the number of subrays for that long ray. For most long rays, this number will just be one, meaning that long ray gets swept exactly one time. The results of the sweep for those long rays is exactly the same as if subray MOC were not being used. For the long rays that intersect with a partially rodded region, the ray is constructed and swept multiple times (usually just twice unless the partially inserted rod has multiple axial regions). During the sweep, volume fractions are used to tally the surface currents for CMFD and the region-wise scalar fluxes outside the partially rodded region as shown in Equation 5.10:

$$\phi_{g,i} = \phi_{g,i} + \overline{\varphi}_{g,i,n,j,z} w_n V_z \ , \tag{5.10a}$$

$$J_{g,s} = J_{g,s} + \varphi^{out}_{g,i,n,j,z} \mu_n w_n V_z \ , \tag{5.10b}$$

where $\phi_{g,i}$ and $J_{g,s}$ are the region i scalar flux and surface s net current, V_z is the volume fraction associated with subray z, $\overline{\varphi}_{g,i,n,j,z}$ and $\varphi^{out}_{g,i,n,j,z}$ are the average angular flux along track j and the outgoing angular flux at the end of track j, and μ_n and w_n are the quadrature azimuthal angle cosines and weights, respectively. Inside the partially rodded region, the scalar fluxes are tallied into two separate regions (one rodded, one unrodded) as described

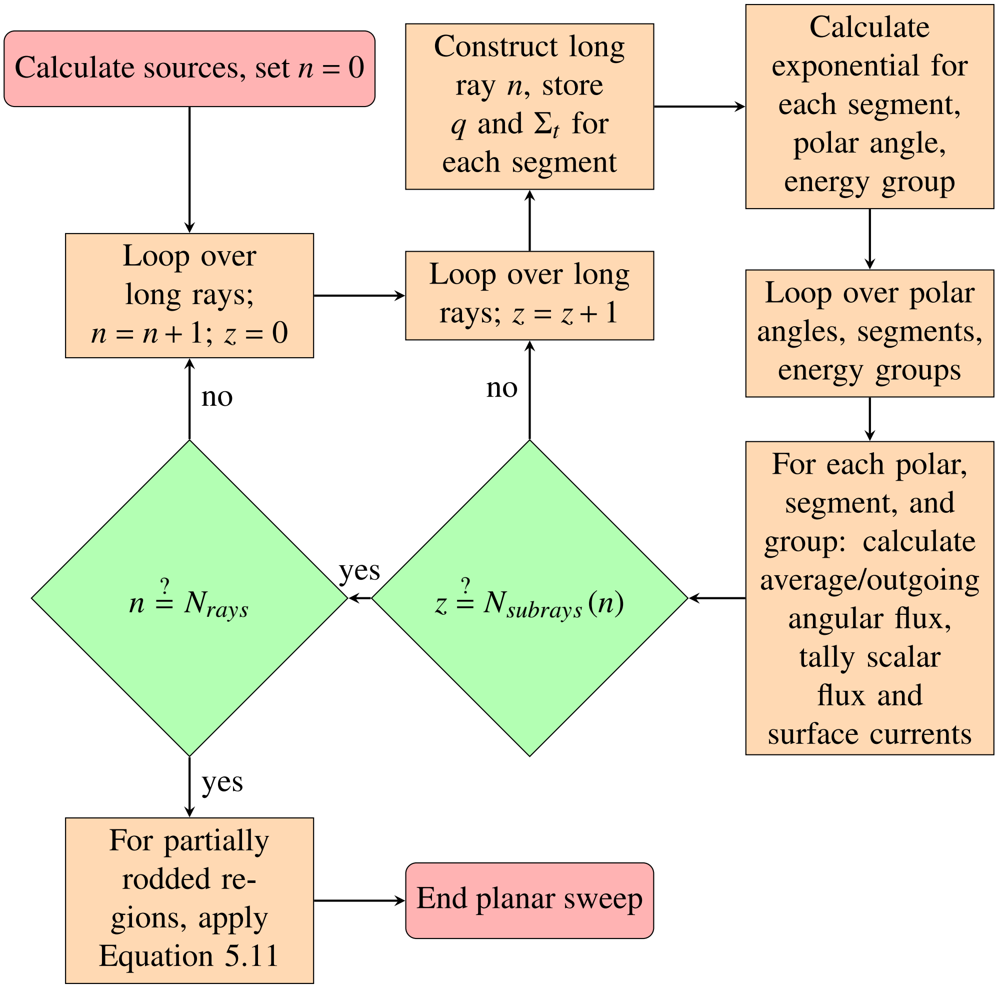

Calculate sources, set $n = 0$
Construct long ray n, store q and Σ_t for each segment
Calculate exponential for each segment, polar angle, energy group
Loop over long rays; $n = n + 1$; $z = 0$
Loop over long rays; $z = z + 1$
Loop over polar angles, segments, energy groups
no
no
For each polar, segment, and group: calculate average/outgoing angular flux, tally scalar flux and surface currents
$n \overset{?}{=} N_{rays}$
yes
$z \overset{?}{=} N_{subrays}(n)$
yes
For partially rodded regions, apply Equation 5.11
End planar sweep

Figure 5.10: Calculation flow for 2D MOC sweep of a single plane with subray

in Section 3.3.3.2. This allows these subregion scalar fluxes to construct the scattering source for the next iteration. After the sweep of the full plane is complete, the subregion scalar fluxes are combined using volume fractions as a post-processing step:

$$\phi_{g,i} = \sum_{z=1}^{Z} V_z \phi_{g,i,z} \, , \tag{5.11}$$

where $\phi_{g,i}$ is the scalar flux for group g and region i for the full-height MOC plane, V_z is the subregion volume fraction for axial level z, and $\phi_{g,i,z}$ is the subregion scalar flux for group g, region i, and axial level z.

This approach has the drawback of duplicating several of the long rays. For problems discussed in Chapter 5, we can reasonably expect around 20-30% of the long rays to intersect a partially rodded region. However, this approach allows the MOC sweeper kernels

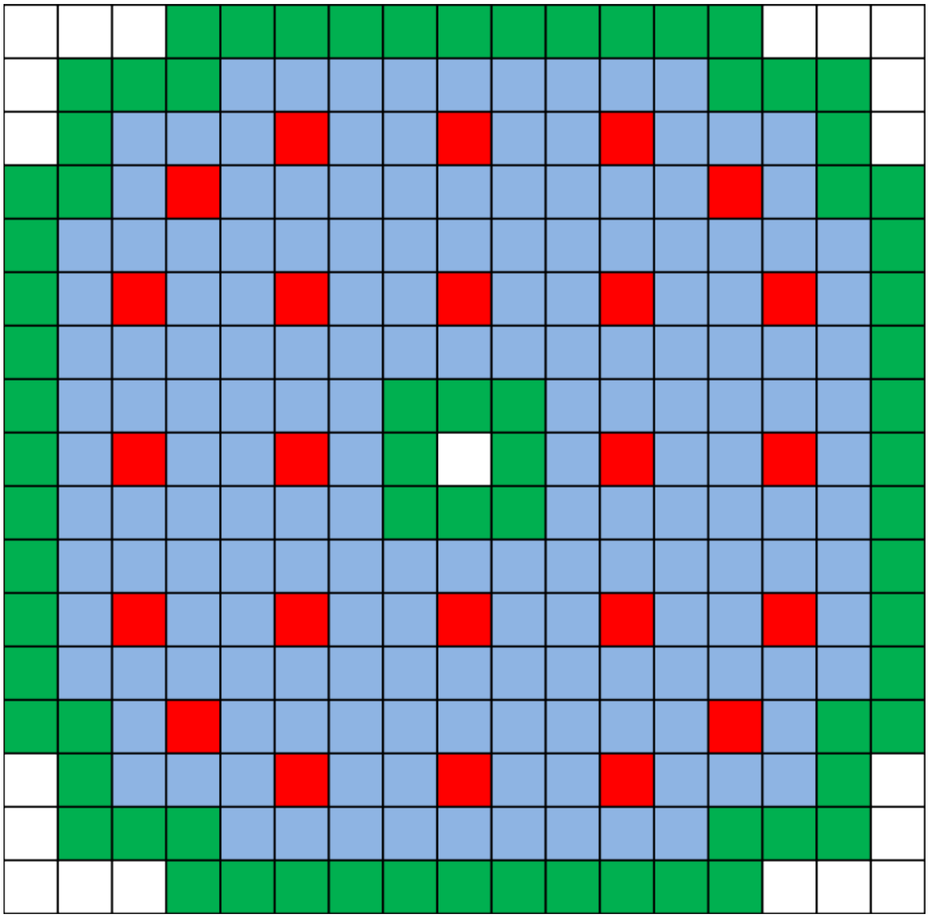


Figure 5.11: Pin cells in 17×17 assembly using subray MOC for recombination flags of 0 (red), 1 (blue), 2 (green), and 3 (white).

to maintain a high degree of efficiency during the calculations. Furthermore, based on duplicating 30% of the long rays, we could expect a 30-40% speedup compared with simply having 2 separate MOC planes to resolve the partially inserted rod. Thus, taking this approach allows subray MOC to be implemented and tested without rewriting MPACT while still attaining significant speedup.

5.3.4.2 Subregion and Recombination Selection

Before performing the subray MOC sweeps, it must be determined which long rays will be duplicated. This is done by examining the mesh for partially rodded regions. Due to the source effects described previously, whenever a partially rodded region is identified, all other regions in that pin cell are also flagged as partially rodded. This allows the axial shape of the source in the surrounding moderator region to also be resolved.

After identifying these regions, neighboring pins can also be flagged. As more rings of neighboring pins are added, the solution should grow closer to that of 2 separate MOC planes. In MPACT, this was implemented via a user-defined recombination flag. A value of 0 will only mark partially rodded regions; a value greater than 0 will add that number of rings around the partially rodded pin cell. Once all these regions are identified, any long ray that passes through any of these regions will be swept multiple times using subray MOC. Some of these rays will not actually intersect the control rod at all, but will serve to resolve the source effects surrounding the rods. The rays will "split" upon entering any of the flagged regions and will recombine only upon entering a non-flagged region. In Figure 5.11, a recombination value of 0 will use subray only for the red pins, which actually have partially inserted control rods; a value of 1 will use subray for red and blue pins; a value of 2 will use subray for red, blue, and green pins; and a value of 3 will use subray for the entire assembly. Regardless of the value, regions in neighboring assemblies will not be flagged unless those assemblies also have partially inserted control rods.

Because the current implementation in MPACT duplicates the whole long ray, the rays are not actually recombined until the problem boundary. However, the logic to only store axially varying sources for certain regions causes this implementation to produce identical answers compared with a more advanced implementation which actually splits and recombines each long ray while sweeping it. The results obtained using this implementation are therefore exactly what is expected from subray MOC, though the efficiency is worse.

5.3.4.3 Subplane CMFD and P_3

Another consideration that must be made concerns the interactions between the subplane CMFD/P_3 calculations and subray MOC. In theory, one could use subray MOC in 2D/1D without using the subplane scheme. However, the fluxes used to calculate the sources for the MOC calculations would be the same for each subregion in an MOC region, limiting the accuracy of the subray MOC calculations. Instead, it is better to modify the projection procedure described in Section 3.3.1.3 to provide an axial shape in the MOC regions. For each MOC subregion, an axial scaling factor is generated using Equation 3.7. The full-height MOC fluxes are multiplied by this scaling factor in each subregion. These values are then used to calculate the scattering and fission sources for the subray MOC calculations. Following this procedure produces more accurate sources and ensures that the volume-averaged flux in each CMFD cell is preserved by both the full-height MOC regions and their subregions.

Furthermore, the axial P_3 calculations are also done on the subplane mesh. The regular MOC regions will use currents at the top and bottom of the MOC plane to calculate the axial TL source. If the subplanes are aligned with the control rod, then currents are also available at the top and bottom of each of the subregions used by subray MOC using Equation 3.3. This allows a more accurate axial shape for the axial TL source, which changes especially quickly in the vicinity of a control rod tip. Thus, using the subplane scheme will improve the accuracy of all three sources used by MOC in the 2D/1D method: scattering, fission, and axial TL.

It should also be noted that as when using subplane CMFD, the radial $\hat{D}$ coupling coefficients should be calculated on the full-height MOC mesh using Equations 2.38. Since much of the MOC calculations are done on this mesh, it is more stable to calculate all radial $\hat{D}$ terms on this mesh. In theory, the radial $\hat{D}$ terms could be calculated separately for each subplane since the subregion fluxes were calculated during the subray MOC sweep, but this subgrid information is only available in the vicinity of the partially inserted rod, making it more stable to use teh full-height fluxes. The axial $\hat{D}$ terms are calculated on the subplanes using the results of the axial P_3 calculations as described in Chapter 3 with no modifications to account for subray MOC.

5.4 Summary

This chapter presented 3 new rod decusping techniques for the 2D/1D method. These techniques vary in complexity and expected accuracy, but each of them captures some of the effects of the partially inserted control rod. The following chapter will present the results of calculations using each of these methods and provide futher discussion on the strengths and limitations of each method.

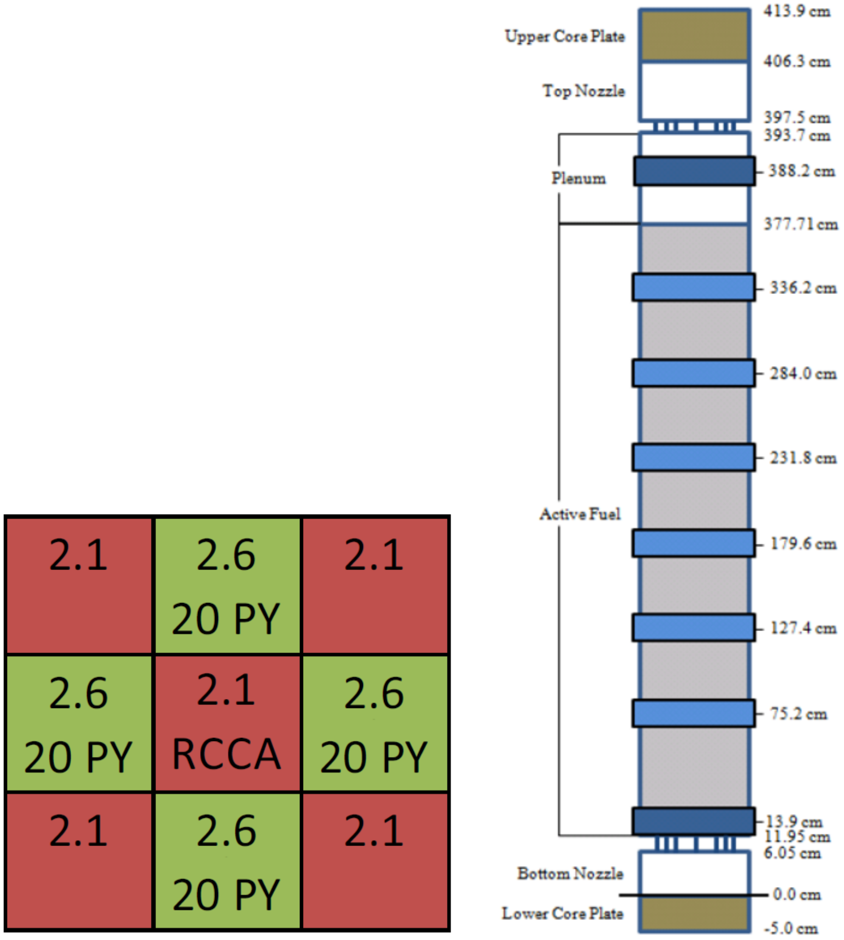
CHAPTER 6

Results

This chapter will present the results of calculations using each of the decusping methods described in the previous chapter. First, results for the polynomial and subplane collision probabilities methods will be presented. The subplane collision probabilities results are presented as two sets of data: one with just subplane (axial correction only) and one with both subplane and CP (axial and radial), allowing the difference in magnitude between the axial and radial effects of the rod to be determined. These methods were tested using the VERA Progression Problems [61], a series of benchmark problems based on the Watts Bar Unit 1 pressurized water reactor (PWR) that provide realistic test cases for the 2D/1D methods. The control rods in these models were moved to various locations that stress the mesh typically used by MPACT to solve these problems, allowing the benefit of the decusping techniques to be clearly seen. For some of these problems, detailed 3D power distributions were generated using the Monte Carlo code KENO-VI [62], allowing for comparisons of the decusping methods with both refined MPACT meshes and Monte Carlo reference solutions.

In the second half of the chapter, results for the subray method of characteristics will be presented. The first set of results come from the 1D prototype code discussed in Section 5.3.1. This code provided a quick implementation to show that it was worth pursuing the method in a full-scale transport code. After this, results from MPACT's implementation of subray MOC will be presented for both 2D and 3D problems. These results will be compared to refined mesh solutions in MPACT as well as the polynomial and subplane collision probabilities solutions to determine the relative accuracy of each of these methods. All subray MOC results will use the C5G7 benchmark problems [59, 60]. Because these problems have specified macroscopic cross sections, some of the complexities of cross section processing and shielding calculations for subray MOC can be deferred to later research while still examining the accuracy and performance of subray MOC compared with other solutions to the rod cusping problem.

6.1 VERA Problem 4



2.1	2.6 20 PY	2.1
2.6 20 PY	2.1 RCCA	2.6 20 PY
2.1	2.6 20 PY	2.1

Figure 6.1: VERA Problem 4 radial (left) and axial (right) layouts

VERA Progression Problem 4 is composed of a 3x3 set of Westinghouse 17×17 fuel assemblies with an active fuel height of 365.76 cm and a control bank in the center assembly. The radial layout of the problem is shown in Figure 6.1(a), and the axial layout of each assembly is shown in Figure 6.1(b). The control rods were placed at an axial elevation of 257.9 cm above the core plate, about one third inserted into the core. The rod in the original problem specification is made of AIC with a B_4C follower and a stainless steel tip. First, results will be presented which use several difference uniform rods to simplify the analysis, then results will be presented with the heterogeneous rod. Lastly, a differential rod worth developed with the heterogeneous rod is shown with comparisons to KENO-VI reference solutions at regular intervals along the curve.

6.1.1 Homogeneous Control Rods

For the reference solution, 58 MOC planes were used with 49 of them in the active fuel region. It was also ensured that the end of the control rods were exactly aligned with one of the MOC plane boundaries. The cases with decusping methods used the same mesh, but with the 2 MOC planes around the tip of the control rod merged into a single plane to introduce cusping effects. The accuracy and convergence data for these cases are shown in Table 6.1.

Table 6.1: Comparison of Rod Decusping Methods in MPACT for VERA Progression Problem 4 for Homogeneous Control Rods

Rod Material	Case	k_{eff} Difference (pcm)	Pin Power Differences RMS	Max	2D/1D Iterations	Runtime (Core-Hours)
AIC	Reference	–	–	–	15	19.6
	No treatment	−29.5	1.53%	11.84%	12	14.7
	Polynomial	−0.3	0.44%	4.08%	12	14.3
	Subplane	−11.5	0.73%	8.21%	12	14.2
	Subplane + CP	−5.6	0.37%	4.25%	12	15.3
B_4C	Reference	–	–	–	15	16.9
	No treatment	112.0	6.98%	69.37%	12	12.7
	Polynomial	112.6	6.89%	66.73%	12	12.1
	Subplane	−17.9	1.14%	11.36%	13	15.2
	Subplane + CP	−11.0	0.69%	6.37%	12	13.4
Tungsten	Reference	–	–	–	15	23.5
	No treatment	−8.4	0.37%	3.37%	12	15.5
	Polynomial	−4.4	0.24%	2.72%	12	14.5
	Subplane	1.6	0.07%	0.60%	12	13.9
	Subplane + CP	−0.9	0.06%	0.94%	12	15.6

The “No Treatment” cases show the magnitude of the cusping effects for each of the rod types. The largest cusping errors occur for the B_4C rod, which is a strong thermal absorber. For this rod, the polynomial treatment has very little effect, leaving large errors in the solution. The subplane method reduces the error in k_{eff} to an almost negligible -18 pcm, but still leaves significant power distribution errors of 1.14% RMS and 11.36% Max. Introduing the 1D CP calculations brings the maximum error below 7%. These results indicate that the polynomial method is not capable of accurately resolving the B_4C rod. Because it is a strong thermal absorber, smearing the rod material throughout the plane at all introduces large errors that cannot be correct just by reducing the volume fraction. The subplane-based methods are both able to resolve the rod more accurately since they actually modify the cross sections used in the CMFD and P_3 calculations.

For the AIC rod, the polynomial correction performs much better than for the B_4C

rod, reducing the maximum error from 12% to 4%. The subplane treatment only reduces the error to 8.21%, while adding the CP calculations reduces the errors to about the same point as the polynomial treatment. The AIC rod has resonance absorption and is not as strongly absorbing in the thermal energy ranges as the B_4C rod, causing some differences in behavior of these methods. The polynomial method does a better job with the AIC rod since the complex effects are accounted for in the polynomial generation. However, the subplane treatment does not perform well since it does nothing to account for the radial effects of the AIC rod. For both rod types, the subplane treatment with radial CP calculations performs well since it is able to capture both radial and axial effects of the rod.

For the tungsten rod, the errors are not nearly as high as for the AIC or B_4C rods. However, the subplane-based treatments still perform significantly better than the polynomial treatment, bringing the k_{eff} error to negligible levels and reducing the maximum power errors under 1%.

For all 3 rod types, Table 6.1 also shows the convergence and runtime data for each calculation. The decusping treatments consistently converge in fewer iterations than the reference. This is due to the fact that the reference has an additional thin MOC plane that the other cases do not have. This thin plane requires underrelaxation for 2D/1D to converge, causing a small increase in the number of 2D/1D iterations required. Because of the difference in iterations, all decusping methods also require less runtime by about 25% on average. The subplane-based methods are generally a bit slower than the polynomial method, despite requiring the same number of 2D/1D iterations. The reason for this is that the subplane-based methods modify the CMFD system, which results in more CMFD inner iterations to converge during each outer 2D/1D iteration. This increases the total runtime of the problem even though 2D/1D itself converges the same. However, in most cases the trade-off between runtime and accuracy would favor the use of the subplane-based

6.1.2 Heterogeneous Control Rod

Next, the same set of results can be shown again, but with the heterogeneous rod. The same 58-plane reference mesh was used for these results. However, the decusping cases have only 55 planes instead of 57. Three additional planes are eneded in the reference case to account for all three material interfaces in the heterogeneous rod: B_4C/AIC, AIC/SS, and SS/moderator. The decusping methods are applied at all three of these locations simultaneously, with the exception of the SS/moderator interface for the polynomial decusping since there is no SS polynomial data. The results are shown in Table 6.2.

With no treatment, the maximum error is over 20%. The polynomial method performs

Table 6.2: **Comparison of Rod Decusping Methods in MPACT for VERA Progression Problem 4 for Heterogeneous Control Rod**

Case	k_{eff}	Pin Power Differences		2D/1D Iterations	Runtime
	Difference (pcm)	RMS	Max		(Core-Hours)
Reference	–	–	–	15	16.6
No treatment	−45.9	2.43%	20.45 %	12	13.8
Polynomial	−2.5	0.46%	5.07 %	12	13.0
Subplane	−17.3	1.10%	11.77 %	13	14.9
Subplane + CP	−5.5	0.42%	3.87 %	12	14.7

wells, reducing the error to about 5%, with an RMS power error under 0.5% and k_{eff} error of -2.5 pcm. This behavior is consistent with what is expected based on the homogeneous results. The primary location of rod cusping effects is at the AIC/SS interface, and the errors obtained from the polynomial decusping method are comparable to those observed in the homogeneous AIC rod calculations. Following this trend, the subplane treatment without radial CP does not perform as well as the polynomial treatment, reducing the maximum error to just under 12%. Finally, the subplane treatment with radial CP generates the most accurate results, with a maximum power error under 4%. This method makes no assumptions about which materials are next to each other, can handle all three control rod materials, and accounts for both radial and axial effects of the rod, making it the most consistent and accurate of the methods across a variety of cases.

The trends in the convergence and runtime data are similar to those observed with the homogeneous rods. The decusping treatment calculations take fewer 2D/1D iterations because fewer thin planes are used in the model, requiring less underrelaxation than the reference case. Furthermore, some increase in runtime is seen in the subplane-based methods over the polynomial method due to changes in the convergence of the CMFD system during each 2D/1D iteration. The CP calculations themselves are negligible since they consist only of inverting an 8×8 matrix for each energy group, which can be done very quickly. Again, the trade-off between runtime and accuracy for this problem favors using the subplane treatment with radial CP if the fine mesh solution's accuracy is not required.

6.1.3 Differential Rod Worth Curve

To show the effectiveness of each decusping technique as the rod moves upward through the reactor, a differential rod worth curve was developed for each method. This curve is shown in Figure 6.2. With no decusping method, the errors in the rodworth approach 50%

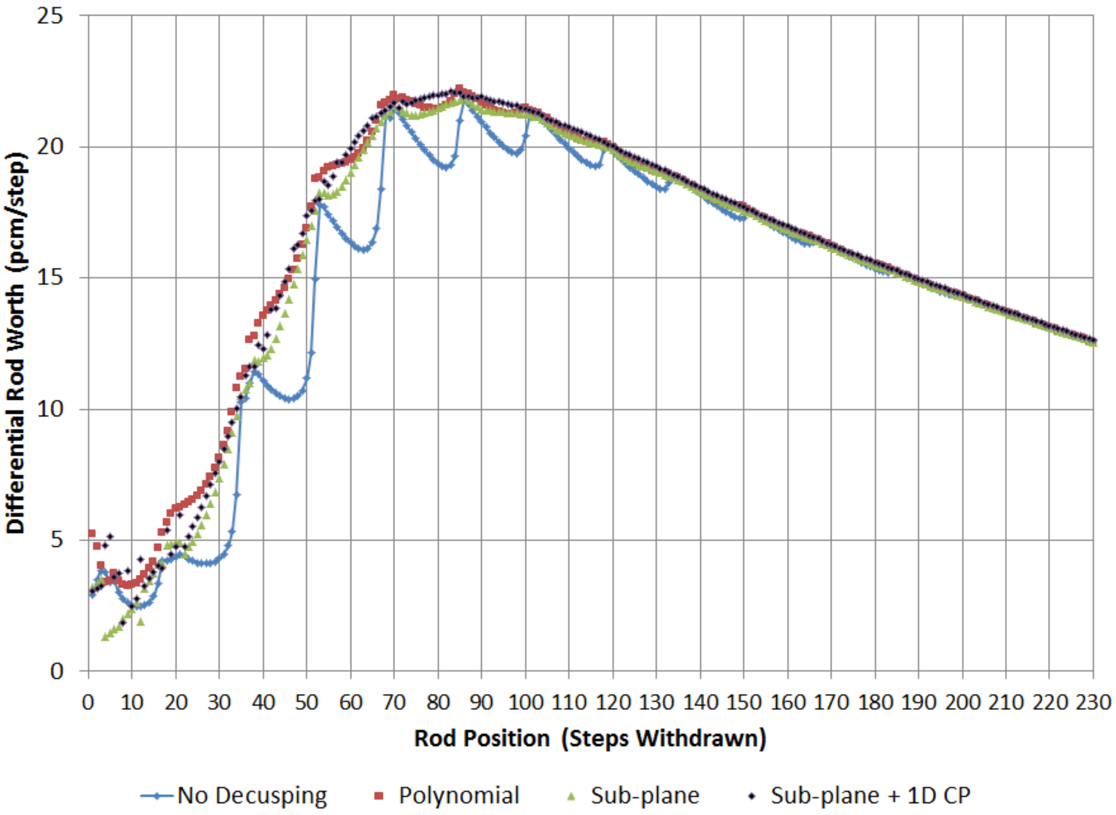


Figure 6.2: Differential rod worth curves for MPACT decusping techniques.

at some positions. The polynomial treatment greatly reduces these errors, but does not fully smooth the curve. Especially near the peak of the curve, some oscillation about the true solution can be seen when using the polynomial treatment. The subplane treatment behaves similarly to the polynomial treatment, but with slightly larger errors in most places. Finally, adding the radial CP calculations generates a smooth curve at almost every rod position.

The exception to the good behavior of the subplane + CP method occurs when the rod is almost fully inserted. At such positions, all methods fail to produce a smooth rod worth curve due to the severe power shape. The polynomial method is always limited in its ability to resolve the partially inserted rod if the local power shape differs significantly from the one with which the polynomial data was generated. The subplane-based treatments struggle with the fully inserted rod because each subplane in an MOC plane uses the same MOC solution data for the CMFD and P_3 solutions. In the cases where MOC planes are near both the control rod tip and the boundary of the problem, the radial solution is changing more quickly than normal and cannot be resolved as well by the subplane scheme.

6.1.4 KENO-VI Comparisons

Table 6.3: Average Differences between MPACT and KENO-VI for VERA Problem 4

Cases	Decusping Method	k_{eff} Difference	Pin Power Difference RMS	Max
Average	None	−24.9	5.380%	25.902%
	Polynomial	34.8	1.502%	8.957%
	Subplane	34.6	0.984%	4.597%
	Subplane + CP	41.4	0.763%	3.386%
Worst – 20%	None	−176.0	14.709%	63.929%
	Polynomial	13.9	3.344%	25.373%
	Subplane	9.6	1.921%	9.900%
	Subplane + CP	45.9	1.324%	4.921%
Fully Withdrawn	–	40.5	0.34 %	1.493%

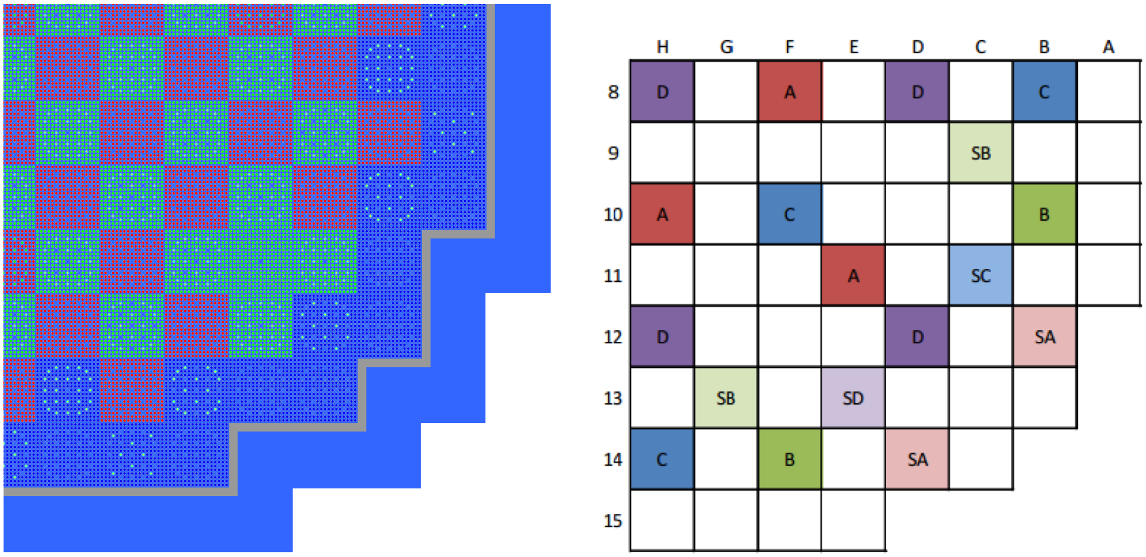
For the rod worth curves in the previous section, KENO-VI calculations were done in 10% intervals along the curve, giving a total of 11 KENO-VI data sets that can be used as a reference for MPACT. Each KENO-VI calculation was run for 500 inactive generations and 1000 active generations, with 5×10^6 particles per generation, for a total of 5×10^9 particles contributing to the final solution and statistics. The uncertainty in the KENO-VI results was about 1 pcm for the eigenvalue and less than 0.3% for the power distribution. While the uncertainty in the power distribution is a bit higher than would normally be used, it is more than sufficiently converged to discuss the relative accuracy of rod cusping treatments. For the first ten sets of KENO-VI data along the curve, each decusping method was compared. Table 6.3 shows the average results for all 10 of these points, along with results for the worst of the 10, which occurs at 20% (46 steps withdrawn) where the differential rod worth curve is steepest. The table also shows a comparison for the 100% withdrawn data set. This final data set occurs for the tip of the control rod above the top of the active fuel, so there are no cusping effects. This serves to give an idea of how accurate MPACT is compared to Monte Carlo when there are no control rod cusping effects.

The fully withdrawn case shows that when no cusping effects are present, MPACT can be expected to have less than 0.5% RMS power difference and approximately 1.5% maximum power difference compared with a Monte Carlo solution. When cusping is introduced, these errors jump to around 25% on average, with a maximum error over 60%. For both the average and maximum cases, the polynomial treatment compares worst with KENO-VI and the subplane treatment with 1D CP performs best, reducing the maximum power errors to about 5% in the worst case scenario. While these errors are still larger than desired for 2D/1D, they are much closer to acceptable levels than some of the 20-30% errors observed in many calculations. However, these KENO-VI comparisons do indicate that use of the polynomial decusping treatment should be restricted to specific cases in which it is known to perform well. Using it for everything could easily and unknowingly result in unacceptably large errors in the 2D/1D solution.

6.2 VERA Problem 5

To demonstrate the behavior of the decusping methods on a full core problem, VERA Problem 5 was also run. Problem 5 is the a beginning-of-cycle simulation of the Watts Bar Unit 1 PWR. The model of this reactor uses the same axial layout shown in Figure 6.1(b) with the radial layout shown in Figure 6.3. For these calculations, Bank D was set to a position of 257.9 cm above the core plate while all other banks were fully withdrawn to 383.3125 cm, about 6 cm above the top of the active fuel.

Like Problem 4, the reference case was run with 58 planes while the decusping cases were run with 57 planes. Radial decomposition was used with 16 cores per MOC plane. This resulted in a slightly different number of cores for the reference case compared with the others, as seen in the Problem 4 calculations. The accuracy and convergence results for the partially rodded plane are shown in Table 6.4.



	H	G	F	E	D	C	B	A
8	D		A		D		C	
9						SB		
10	A		C				B	
11				A		SC		
12	D				D		SA	
13		SB		SD				
14	C		B		SA			
15								

Figure 6.3: VERA Problem 5 radial geometry (left) and rod bank positions (right)

As seen in most variations of Problem 4, we see that the CP results are the best, with

Table 6.4: VERA Problem 5 decusping results for the partially rodded plane

Case	k_{eff}	Pin Power Differences		Iterations		Runtime
	Difference (pcm)	RMS	Max	2D/1D	CMFD	(Core-Hours)
Reference	–	–	–	13	481	361.7
No Treatment	-22	6.90%	30.55%	13	523	410.7
Polynomial	-5	1.15%	4.85%	13	463	373.7
Subplane	-5	2.09%	10.20%	13	499	399.0
Subplane + CP	-1	0.50%	2.74%	13	529	425.6

less than 3% maximum error in the roded plane. The maximum errors occur in the partially rodded assembly, as expected. The sublpane decusping without the CP treatment is worse than in Problem 4, showing the importance of correctly treating the radial effects. The runtime increase is also between 15% and 20% for the sublpane and CP decusping methods. This is due primarily to two effects. The first is that the convergence of the CMFD system can require more iterations using the subplane-based treatments, as discussed in the Problem 4 results. The second effect is that there is an imbalance in the parallel partitioning of the CMFD system. The parallel partitioning in MPACT is tied to MOC planes, so MOC planes with a larger number of subplanes will take longer to be solved than others, leaving some compute cores waiting on others to finish their calculations. This effect would have also occurred some Problem 4, but is more significant in Problem 5 due to its larger size. A more clever strategy for parallel decomposition would help to alleviate this imbalance and reduce the runtime of the subplane-based methods without any impact on their accuracy.

6.3 C5G7 Results

This section will focus on the results for the C5G7 benchmark problems for the subray method of characteristics. First, 1D results generated by a simple prototype code will be presented, followed by 2D and 3D results from the 2D/1D code MPACT. While the focus of this section is on subray MOC, the 2D and 3D results will also include the polynomial and subplane collision probabilities decusping techniques to show the accuracy of subray MOC compared with the other methods discussed already.

6.3.1 1D Subray

The 1D subray MOC results were generated using the code described in Section 5.3.1. To show the effectiveness of subray MOC a small 10 pin problem, illustrated in Figure 6.4, was

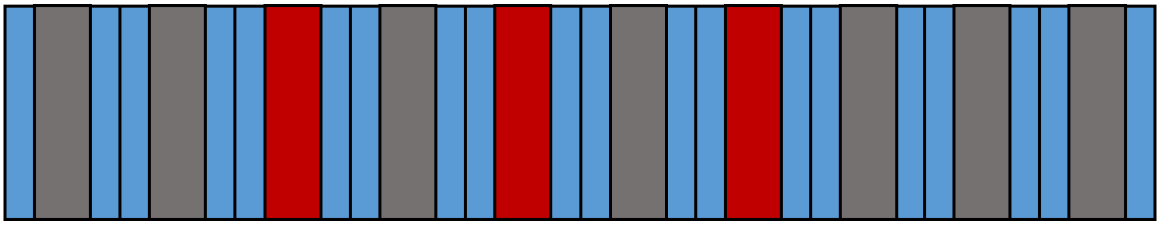


Figure 6.4: Illustration of 1D MOC 10 pin test geometry with materials fuel (grey), control rod/moderator mixture (red), and moderator (blue).

used. This smaller problem was used because the rods are closer together, making it more difficult to accurately resolve the rods, and to minimize runtime. A series of eigenvalue calculations was performed with the control rod volume fraction varying from 100% to 0% to simulate a rod withdrawal. This was done using traditional MOC and subray MOC to show the correction achieved by using subray MOC. The reference solution for each of these calculations was generated from the fully rodded and fully unrodded solutions. At each point, those two solutions were mixed using the rodded/unrodded volume fractions. For example, when the rod was 40% inserted, the reference solution was $0.4\phi_R(x) + 0.6\phi_U(x)$, where ϕ_R is the fully rodded scalar flux and phi_U is the fully unrodded scalar flux. Doing this removes the effects of the cross section homogenization from the solution and provides a reference for subray MOC. The results of these calculations are shown in Figure 6.5.

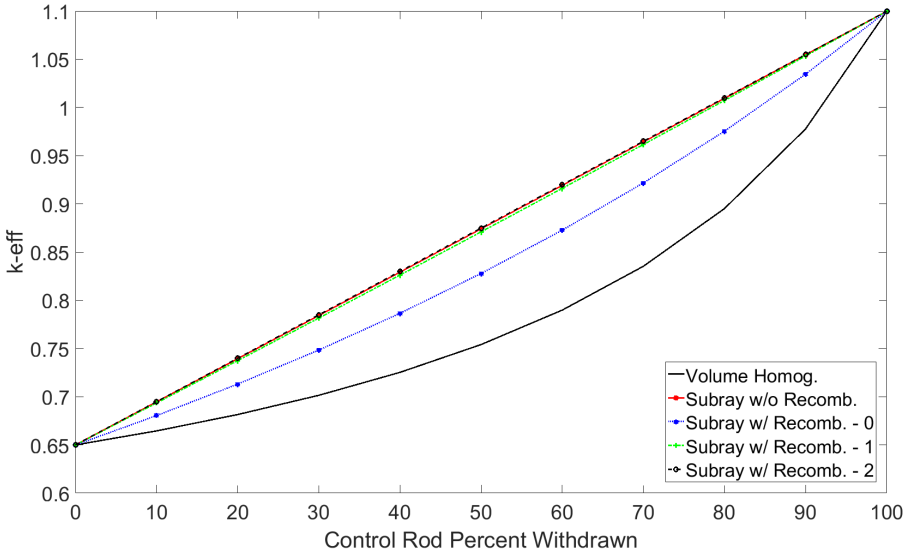
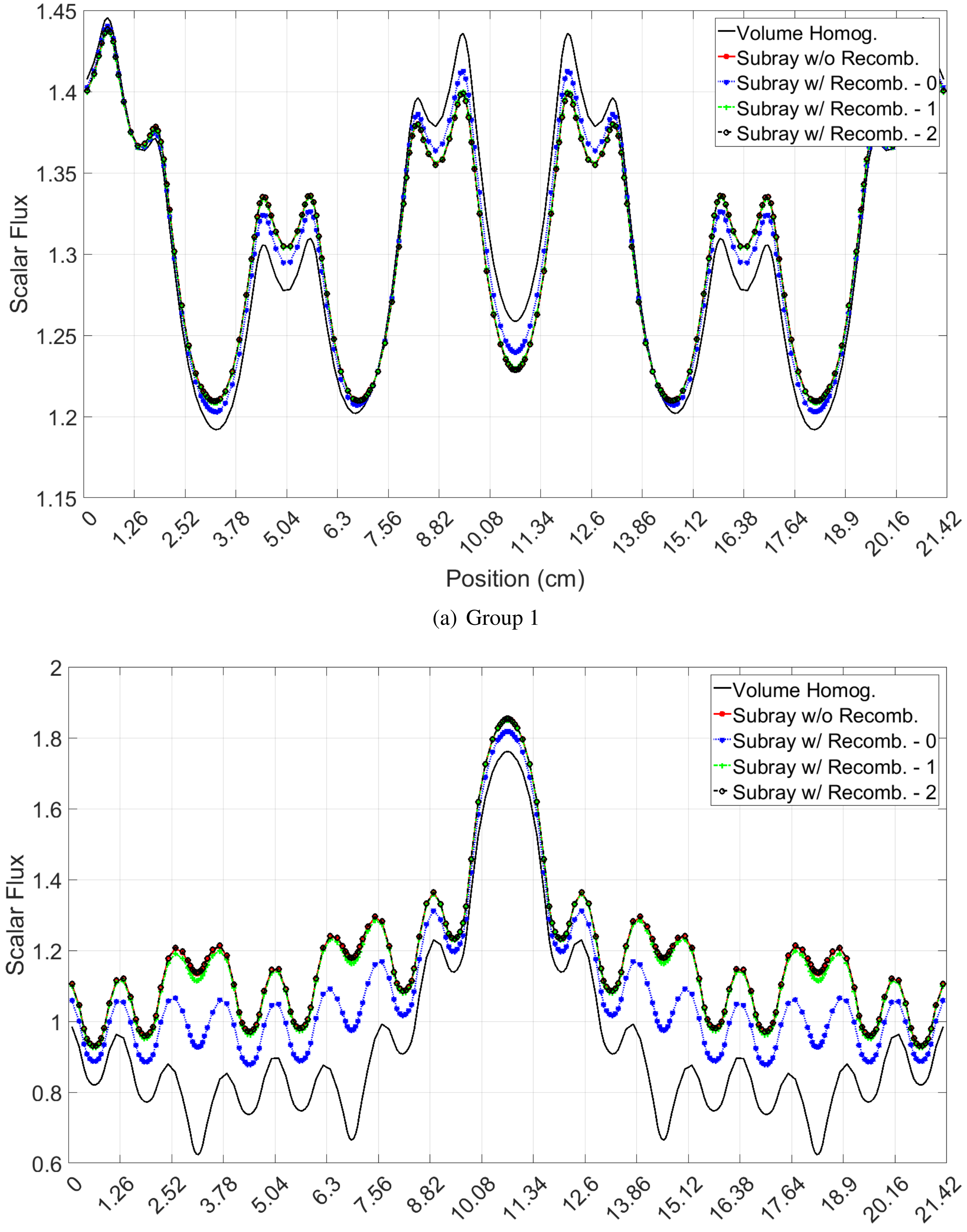


Figure 6.5: Eigenvalue comparisons for 1D rod withdrawal

In Figure 6.5, the red line is subray without recombination, meaning that two completely separate solves were done each iteration and the resulting scalar fluxes were mixed



(a) Group 1

(b) Group 7

Figure 6.6: Scalar flux distributions for 1D subray MOC calculations

using volume fractions. The rodded and unrodded solves have separate angular flux boundary conditions which are saved for each subsequent iteration. This provides the reference solution. The volume homogenization solution is a traditional MOC calculation without treating the partially inserted rod. As is evident, the errors from this method are large, nearing 15% around 50-60% withdrawal. Using subray MOC in only the partially rodded pin cells, indicated in the figure by "Subray w/ Recomb. - 0", corrects a significant amount of the error and brings the maximum difference closer to 5%. Using a recombination flag of 1 means that subray is now used in all the pin cells between the control rods, correctly resolving the interference effects of the rods and eliminating almost all of the remaining error. Finally, a recombination flag of 2 extends the subrays all the way to the problem boundary, effectively performing the same calculation as the reference solution.

In addition to the eigenvalue calculations, the scalar flux differences are shown for groups 1 and 7 at 50% rod withdrawal in Figure 6.6. For both groups, it is evident that subray MOC effectively reduces the error caused by the partially inserted rod. Only performing subray MOC in the rodded pin cells eliminates the majority of the error, but for both the eigenvalue and scalar flux distributions, it is necessary to delay recombination of the angular fluxes for at least one neighboring pin cell to capture the effects of the rod on the source in neighboring pin cells. This effect is exaggerated by the 1D geometry since the solve is being performed exlusively in the direction that minimizes the distance between pins. Furthermore, most PWR geometries have at least 2 pins between control rodlets, which further diminishes the effects of rod interference. While these 1D results do not definitively indicate that recombination should be extended beyond the edge of the partially rodded pin cell, it is important to keep this in mind in the 2D and 3D calculations.

6.3.2 2D C5G7

To test the mechanics of subray MOC in MPACT, 2D variations of the C5G7 benchmark problems were used. These problems consist of 17×17 UO_2 and MOX assemblies with control rods and a radial reflector region. Figure 6.7 shows an illustration of the core layout, and Figure 6.8 shows the assembly layout. This section will present results for subray MOC using 3 different 2D problems derived from these descriptions.

All calculations in this section used 0.03 cm ray spacing with 16 azmiuthal angles and 3 polar angles per octant. A 1D P_3 solver was used for the subplane axial calculations unless otherwise noted. The calculations were converged until the change in both the eigenvalue and fission source were less than 10^{-6}. All pin meshes in the fuel assemblies used 5 radial rings in the fuel and control rods, 2 rings in the moderator, and 8 azimuthal divisions in

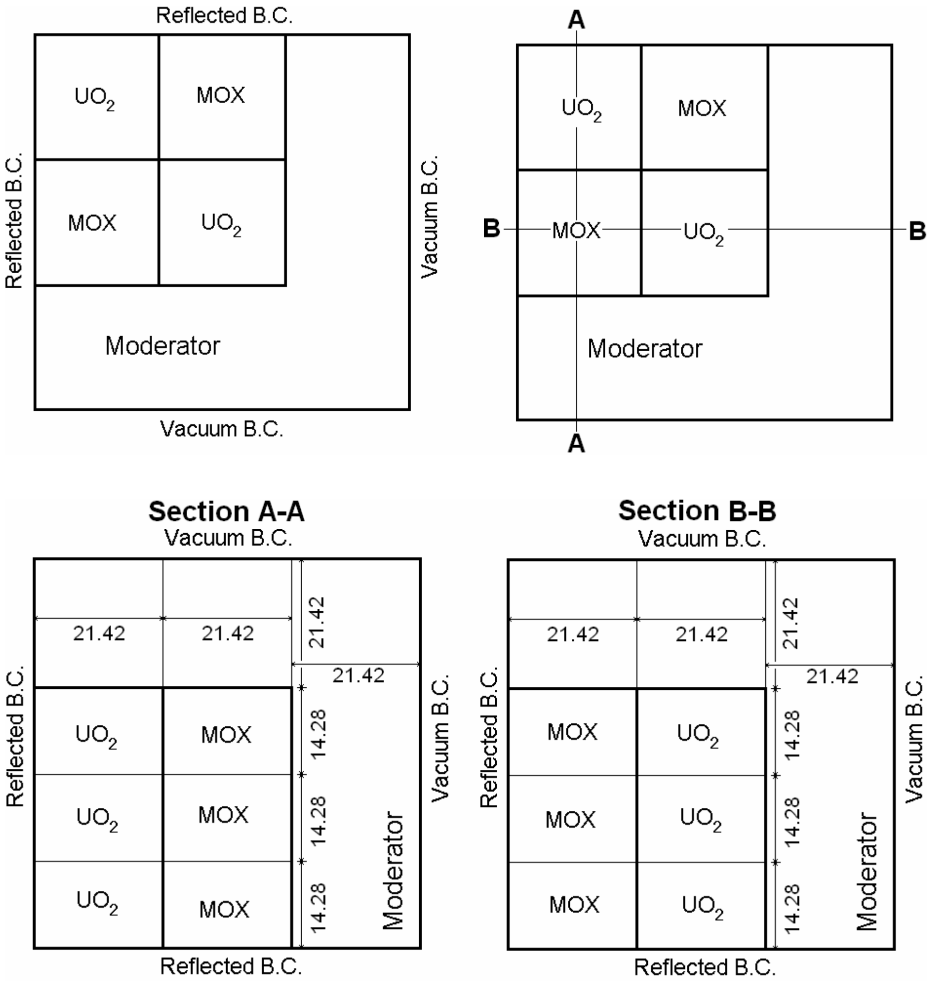


Figure 6.7: Description of C5G7 Core [60]

every radial region for a total of 64 flat source regions in each pin cell. This mesh was also used for the rodded pin cells in the axial reflector region. The remaining axial reflector pin cells and all radial reflector pin cells used a Cartesian grid with 5 x- and y- divisions, for a total of 25 equal-volume flat source regions.

These problems will not truly be 2D calculations since there will still be some coupling between the rodded and unrodded portions of the problem. However, they will each consist of a single MOC plane (with the exception of the reference calculations). Furthermore, the case with no decusping method truly is a 2D calculation with rod cusping results caused by volume homogenizing the rod with the moderator. For simplicity, these cases will simply be called 2D C5G7 to distinguish them from the multi-plane 3D C5G7 problems presented in a later section.

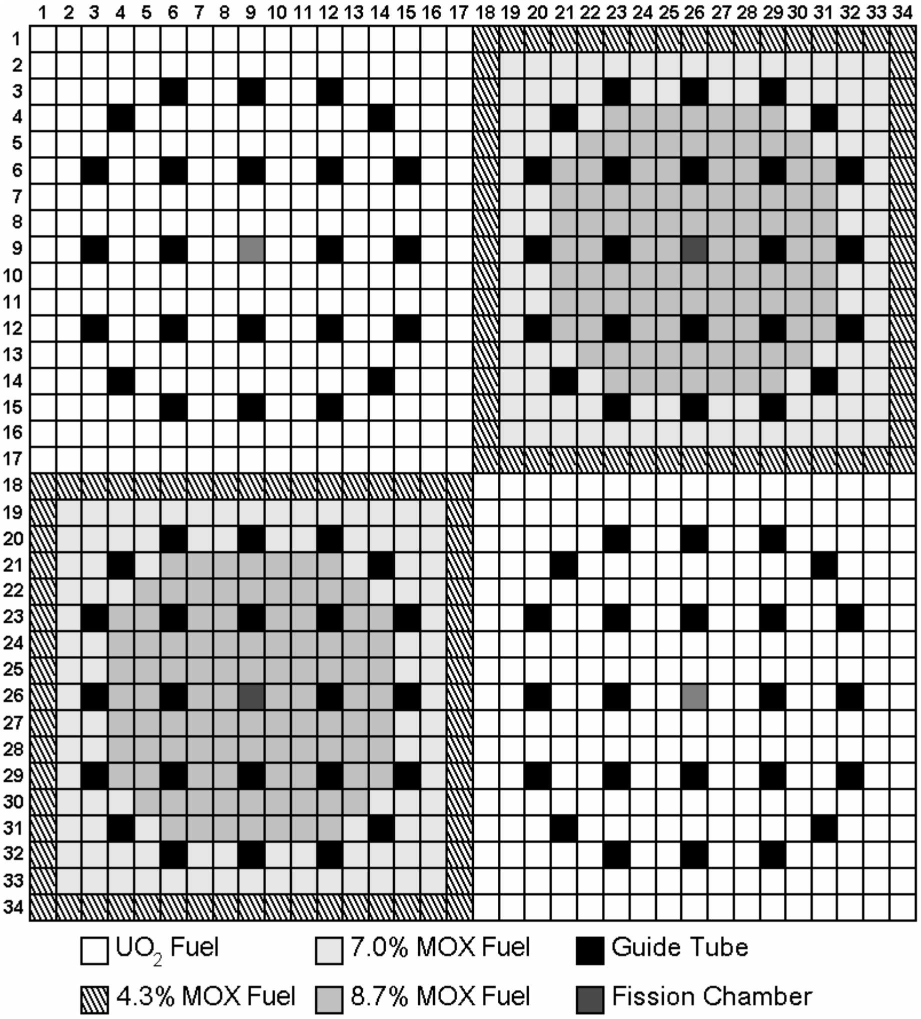


Figure 6.8: Description of C5G7 Assembly [60]

6.3.2.1 UO$_2$ Assembly

The first 2D problem is a single UO$_2$ assembly with control rods added to it. A similar procedure was followed as with the 1D calculations with the problem being simulated at 10% increments during the rod withdrawal. The reference for these calculations was generated by dividing the MOC plane into two separate planes with their boundary aligned with the control rod. Comparisons could then be made between the decusping methods and the reference solution for k_{eff} and the radial power distribution.

Table 6.5 shows the results for the 2D assembly case for recombination flag values of 0, 1, 2, and 3. The results for both k_{eff} and power distribution are best for the Subray-0 option, with average k_{eff} differences less than 200 pcm and maximum power differences averaging 0.22%. The remaining subray options each perform almost exactly the same,

Table 6.5: 2D C5G7 UO_2 assembly eigenvalue and power distribution differences for subray recombination parameter sensitivity. A * indicates that finite difference was used axially instead of P_3

Rod	Reference		Subray-0			Subray-1			Subray-2			Subray-3	
Position	k_{eff}	k_{eff}	Pin Powers		k_{eff}	Pin Powers		k_{eff}	Pin Powers		k_{eff}	Pin Powers	
			RMS	Max		RMS	Max		RMS	Max		RMS	Max
1*	1.02535	-23	0.05%	0.09%	-59	0.06%	0.11%	-58	0.06%	0.11%	-58	0.06%	0.11%
2	1.06065	-117	0.11%	0.21%	-121	0.11%	0.22%	-120	0.11%	0.22%	-120	0.11%	0.22%
3	1.09578	-165	0.14%	0.27%	-175	0.15%	0.28%	-175	0.15%	0.28%	-174	0.15%	0.28%
4	1.13011	-200	0.15%	0.29%	-219	0.16%	0.30%	-218	0.16%	0.30%	-217	0.16%	0.30%
5	1.16334	-223	0.15%	0.29%	-249	0.17%	0.31%	-248	0.17%	0.30%	-247	0.17%	0.30%
6	1.19557	-235	0.14%	0.27%	-265	0.16%	0.29%	-264	0.16%	0.29%	-264	0.16%	0.29%
7	1.22711	-237	0.13%	0.24%	-269	0.15%	0.26%	-269	0.14%	0.26%	-269	0.14%	0.26%
8	1.25858	-231	0.11%	0.20%	-263	0.13%	0.22%	-262	0.12%	0.22%	-262	0.12%	0.22%
9*	1.29131	-216	0.09%	0.16%	-236	0.10%	0.18%	-235	0.10%	0.18%	-235	0.10%	0.18%
Average	–	183	0.12%	0.22%	206	0.13%	0.24%	205	0.13%	0.23%	205	0.13%	0.23%

Table 6.6: 2D C5G7 UO_2 assembly eigenvalue and power distribution differences for each decusping method. A * indicates that finite difference was used axially instead of P_3

Rod	Reference		Subray-0			None			Subplane			Subplane + CP	
Position	k_{eff}	k_{eff}	Pin Powers		k_{eff}	Pin Powers		k_{eff}	Pin Powers		k_{eff}	Pin Powers	
			RMS	Max		RMS	Max		RMS	Max		RMS	Max
1*	1.02535	-23	0.05%	0.09%	-1210	0.70%	1.46%	-361	0.20%	0.42%	-696	0.34%	0.78%
2	1.06065	-117	0.11%	0.21%	-3123	1.71%	3.55%	-724	0.38%	0.78%	-626	0.30%	0.65%
3	1.09578	-165	0.14%	0.27%	-4777	2.49%	5.11%	-937	0.46%	0.94%	-574	0.26%	0.56%
4	1.13011	-200	0.15%	0.29%	-6053	2.98%	6.09%	-1033	0.47%	0.96%	-522	0.23%	0.48%
5	1.16334	-223	0.15%	0.29%	-6848	3.19%	6.47%	-1046	0.44%	0.90%	-465	0.20%	0.40%
6	1.19557	-235	0.14%	0.27%	-7067	3.11%	6.26%	-996	0.40%	0.81%	-400	0.16%	0.33%
7	1.22711	-237	0.13%	0.24%	-6588	2.73%	5.45%	-897	0.34%	0.68%	-327	0.13%	0.25%
8	1.25858	-231	0.11%	0.20%	-5245	2.04%	4.03%	-754	0.27%	0.54%	-242	0.09%	0.18%
9*	1.29131	-216	0.09%	0.16%	-2803	1.01%	1.98%	-561	0.19%	0.38%	-146	0.05%	0.10%
Average	–	183	0.12%	0.22%	4857	2.22%	4.49%	812	0.35%	0.71%	444	0.20%	0.41%

and only slightly worse than the Subray-0 variation. The reason that the other options do not perform as well as Subray-0 is that during the CMFD projection, the same radial shape is used for each axial level. This impacts all subray MOC calculations, but Subray-0 uses subrays only in the rodded pins, not fuel pins. Because this approximation is applied to some fuel pins for the other subray variations, it introduces some small changes in the solution. However, the most important conclusion from these results is that the differences in subray MOC variations are so small that it is likely not worth it to ever use subray MOC in the neighboring pins.

Table 6.6 compares the results of subray MOC with the sublpane-based decusping methods described earlier. The polynomial method is not compared because the data required to use them fo C5G7 has not been generated. The subplane decusping method performs better with CP than without, as expected. Overall, its performance is good, reducing the maximum power error to around 0.5% on average and reducing the k_{eff} error from an average of almost 5% to under 500 pcm. Subray MOC performs the best, with k_{eff} differences around 200 pm and maximum power distribution errors less than 0.3% for every case. This is to be expected since subray MOC is doing something similar to the Subplane + CP method, but improving on it by using 2D MOC instead of 1D CP and allowing the improved solution to influence neighboring pins during the MOC sweep through the improve outgoing angular fluxes. Thus, it is to be expected that subray MOC would perform best.

This relationship between Subplane + CP and Subray-0 is expected. Both of them are improving the solution in individual rodded pin cells by solving the rodded and unrodded levels separately. However, Subplane + CP does this after the MOC calculation, only improving the CMFD and 1D P_3 cross sections. The Subray-0 calculation uses 2D MOC, which should provide a solution superior to 1D CP. It also calculates improved radial currents, and improves the solution in pins neighboring the control rodlets by improving the angular flux exiting the rodded pin cells. Thus, it is expected that Subray-0 would consistently provide a better solution than the Subplane + CP decusing method.

It should be noted at this point that certain calculations whose results are shown in these and following tables failed to converge using the axial P_3 solver. This occurred for all calculations that had multiple planes or used the subplane scheme. This is a known issue with the axial solvers caused by the interpolation of the radial TL source. Whenever a thin plane is next to a thick plane and a very large change in the radial TL occurs from the thin plane to the thick plane, the quadratic interpolation of the radial TL can cause shapes that make the P_3 solver unstable. This can be shown to be the source of the instability by limiting the magnitude of the linear and quadratic components of the shape.

One of the other MPACT developers is currently implementing a new solver that employs a different technique to solve the P_3 equations, and is expected to resolve this instability. Since that solver is not yet ready for use and limiting the linear and/or quadratic moments of the radial TL interpolation can have significant effects on accuracy, the cases which did not converge for either the reference or decusping method calculations simply used diffusion-based NEM solver instead of the P_3 solver. The diffusion-based solver uses a two-node iteration scheme as opposed to the one-node approach used by the P_3 solver, making it more stable in some cases. The diffusion solver is not as accurate as P_3, but using it in cases with and without cusping allows for a fair assessment of methods being tested here without abandoning the axial solve altogether. The results that used diffusion instead of P_3 will be denoted with an asterisk (*) in all tables in this section.

Table 6.7: Long Ray Data for 2D C5G7 UO_2 Rodded Assembly

Method	Long Rays per Plane	Increase
Reference	29480	–
Subray-0	48440	64%
Subray-1	52832	79%
Subray-2	56304	91%
Subray-3	58960	100%

Table 6.8: Average Performance and Convergence Data for C5G7 Rodded UO_2 Assembly

Method	Iterations	Runtime (s)	Speedup
Reference	16.4	119	–
None	14.6	56	2.14
Subplane	14.9	57	2.10
Subplane + CP	15.2	63	1.91
Subray-0	14.9	94	1.27
Subray-1	15.3	104	1.15
Subray-2	15.4	109	1.09
Subray-3	15.3	107	1.11

While the subray MOC method is more accurate than the previous decusping methods, it is also more expensive. Table 6.7 shows the long ray data for the reference case and each variation of the subray MOC calculations. The first column is the total number of long rays including those duplicated for subray MOC. The second column shows the percent increase in the number of long rays. This value can be used to approximate the savings obtained by using subray MOC to resolve partially inserted rods instead of 2 separate MOC planes. For the single assembly case, it is expected that Subray-3 and the reference case would take about the same amount of time for each MOC sweep since every long ray was duplicated for Subray-3.

Table 6.8 shows the convergence and runtime data for each decusping method averaged over all 9 rod positions. Each of these calculations was conducted on a development cluster with 4 AMD Opteron™ 6376 processors clocked at 2.3 GHz. The speedup of each case is defined as the reference runtime divided by the comparison runtime. Using Table 6.7, we can approximate an expected speedup for each case by taking the number of long rays in the reference (2 MOC planes) and dividing by the number in the comparison case (1 MOC plane). Doing this, we get expected speedups of 1.22, 1.12, 1.05, and 1.0 for Subray-0, Subray-1, Subray-2, and Subray-3, respectively. The observed speedups are similar to, but consistently a little better than predicted. On average, subray MOC takes between 1 and 1.5 iterations fewer than the reference case, which can account for some of the difference. The other methods all see a speedup around 2.0, which is expected since the amount of time spent in MOC is exactly half that of the reference. The differences in the CMFD solve and number of iterations introduce some variation in the speedup, which is to be expected.

6.3.2.2 2D Core – Rodded Center Assembly

Table 6.9: 2D C5G7 center assembly rod withdrawal eigenvalue and pin power differences for subray recombination parameter sensitivity. A * indicates that finite difference was used axially instead of P_3

Rod	Reference		Subray-0			Subray-1			Subray-2			Subray-3	
Position	k_{eff}	k_{eff}	Pin Powers		k_{eff}	Pin Powers		k_{eff}	Pin Powers		k_{eff}	Pin Powers	
			RMS	Max		RMS	Max		RMS	Max		RMS	Max
1*	1.06839	-15	0.10%	0.29%	-15	0.10%	0.29%	-15	0.10%	0.29%	-15	0.10%	0.29%
2	1.07746	-33	0.22%	0.67%	-34	0.22%	0.68%	-34	0.22%	0.67%	-34	0.22%	0.67%
3	1.08777	-53	0.32%	1.03%	-56	0.34%	1.07%	-55	0.34%	1.06%	-55	0.34%	1.06%
4	1.09919	-72	0.41%	1.34%	-78	0.45%	1.45%	-78	0.45%	1.44%	-78	0.45%	1.44%
5	1.11160	-89	0.46%	1.53%	-99	0.51%	1.69%	-98	0.50%	1.66%	-98	0.50%	1.66%
6	1.12495	-102	0.49%	1.66%	-115	0.55%	1.83%	-115	0.54%	1.82%	-115	0.54%	1.81%
7	1.13925	-112	0.49%	1.70%	-127	0.55%	1.88%	-126	0.55%	1.87%	-126	0.55%	1.86%
8	1.15469	-117	0.47%	1.65%	-133	0.53%	1.83%	-132	0.53%	1.81%	-132	0.52%	1.81%
9*	1.17190	-117	0.43%	1.50%	-127	0.46%	1.61%	-126	0.46%	1.60%	-126	0.46%	1.60%
Average	–	79	0.38%	1.26%	87	0.41%	1.37%	87	0.41%	1.36%	87	0.41%	1.36%

Table 6.10: 2D C5G7 center assembly rod withdrawal eigenvalue and pin power differences for each decusping method. A * indicates that finite difference was used axially instead of P_3

Rod	Reference		Subray-0			None			Subplane			Subplane + CP	
Position	k_{eff}	k_{eff}	Pin Powers		k_{eff}	Pin Powers		k_{eff}	Pin Powers		k_{eff}	Pin Powers	
			RMS	Max		RMS	Max		RMS	Max		RMS	Max
1*	1.06839	-15	0.10%	0.29%	-286	1.73%	4.47%	-87	0.52%	1.35%	-169	0.99%	2.46%
2	1.07746	-33	0.22%	0.67%	-811	4.75%	12.70%	-198	1.13%	3.06%	-174	0.97%	2.57%
3	1.08777	-53	0.32%	1.03%	-1369	7.71%	21.30%	-290	1.56%	4.42%	-181	0.97%	2.70%
4	1.09919	-72	0.41%	1.34%	-1918	10.33%	29.42%	-360	1.83%	5.35%	-185	0.94%	2.75%
5	1.11160	-89	0.46%	1.53%	-2400	12.25%	36.00%	-405	1.93%	5.84%	-184	0.88%	2.68%
6	1.12495	-102	0.49%	1.66%	-2738	13.12%	39.83%	-424	1.89%	5.89%	-174	0.78%	2.47%
7	1.13925	-112	0.49%	1.70%	-2820	12.55%	39.38%	-416	1.72%	5.52%	-155	0.65%	2.11%
8	1.15469	-117	0.47%	1.65%	-2478	10.08%	32.80%	-377	1.44%	4.76%	-124	0.48%	1.62%
9*	1.17190	-117	0.43%	1.50%	-1461	5.31%	18.00%	-300	1.06%	3.60%	-80	0.29%	1.00%
Average	–	79	0.38%	1.26%	1809	8.65%	25.99%	317	1.45%	4.42%	158	0.77%	2.26%

The second problem uses the full radial core layout from Figures 6.7 and 6.8. A control rod was added to the northwest UO_2 assembly, and the same procedure was followed as for the 2D assembly. Table 6.9 shows the results for subray MOC as a function of recombination value. Overall similar trends occur for this problem, with Subray-0 resulting in the best results by a small margin and the other variations producing nearly identical results. The k_{eff} errors are smaller because larger problems are less sensitive to local reactivity effects. However, the power distribution is more challening to resolve with the reflector regions and multiple fuel types in the model. This increases the errors in the power distribution, but as shown in Table 6.10, subray MOC still reduces the errors much more than any other method. The Subplane + CP method has maximum power errors that are almost a factor of 2 worse that Subray-0. Without CP, these errors are greater than 5% for about half the cases.

Table 6.11: Long Ray Data for C5G7 Core

Method	Long Rays per Plane	Increase
Reference	88440	–
Subray-0	107400	21%
Subray-1	111792	26%
Subray-2	115264	30%
Subray-3	117920	33%

Table 6.12: Average Performance and Convergence Data for 2D C5G7 Core with Rodded NW Assembly

Method	Iterations	Runtime (s)	Speedup
Reference	25.7	1107	–
None	25.8	739	1.50
Subplane	26.7	725	1.53
Subplane + CP	26.9	695	1.61
Subray-0	26.4	861	1.29
Subray-1	25.9	881	1.26
Subray-2	26.1	920	1.21
Subray-3	26.1	938	1.18

As with the single assembly, the long ray data can again be used to make some predictions about performance. Because this problem has multiple assemblies and only one is partially rodded, the speedup observed for subray MOC is expected to be larger than that of the single assembly case. Using the same procedure as for the assembly, we can predict approximate speedups of 1.65, 1.59, 1.54, and 1.50 for Subray-0, Subray-1, Subray-2, and

Subray-3, respectively. In this case the observed speedups are significantly worse than predicted, for several reasons. First, subray MOC takes more iterations on average than the reference. Second, the subplane CMFD and P_3 systems are the same size for all decusping methods as for the reference. This was not important for the single assembly, but the 2D core is large enough that these other parts of the 2D/1D iteration scheme can no longer be ignored in predicting the runtime. However, in spite of these issues, a speedup of 20-30% is still achieved with each of the subray MOC methods.

6.3.2.3 2D Core – Rodded Corner Assembly

Table 6.13: 2D C5G7 corner assembly rod withdrawal eigenvalue and pin power differences for subray recombination parameter sensitivity. A * indicates that finite difference was used axially instead of P_3

Rod	Reference		Subray-0			Subray-1			Subray-2			Subray-3	
Position	k_{eff}	k_{eff}	Pin Powers		k_{eff}	Pin Powers		k_{eff}	Pin Powers		k_{eff}	Pin Powers	
			RMS	Max		RMS	Max		RMS	Max		RMS	Max
1*	1.18889	-2	0.03%	0.11%	-2	0.03%	0.11%	-1	0.03%	0.11%	-1	0.03%	0.11%
2	1.18966	-4	0.06%	0.23%	-4	0.06%	0.23%	-3	0.06%	0.23%	-3	0.06%	0.23%
3	1.19048	-6	0.09%	0.33%	-5	0.09%	0.34%	-5	0.09%	0.33%	-5	0.09%	0.33%
4	1.19135	-8	0.11%	0.40%	-7	0.12%	0.42%	-7	0.12%	0.41%	-7	0.12%	0.41%
5	1.19227	-9	0.13%	0.45%	-9	0.14%	0.47%	-8	0.14%	0.47%	-8	0.14%	0.47%
6	1.19324	-10	0.14%	0.47%	-10	0.16%	0.50%	-9	0.15%	0.50%	-9	0.15%	0.50%
7	1.19427	-10	0.15%	0.47%	-11	0.17%	0.51%	-10	0.16%	0.51%	-10	0.16%	0.51%
8	1.19538	-10	0.15%	0.46%	-11	0.17%	0.50%	-11	0.17%	0.50%	-11	0.17%	0.50%
9*	1.19666	-10	0.15%	0.42%	-10	0.16%	0.44%	-10	0.16%	0.44%	-10	0.16%	0.44%
Average	–	8	0.11%	0.37%	7	0.12%	0.39%	7	0.12%	0.39%	7	0.12%	0.39%

Table 6.14: 2D C5G7 corner assembly rod withdrawal eigenvalue and pin power differences for each decusping method. A * indicates that finite difference was used axially instead of P_3

Rod	Reference		Subray-0			None			Subplane			Subplane + CP	
Position	k_{eff}	k_{eff}	Pin Powers		k_{eff}	Pin Powers		k_{eff}	Pin Powers		k_{eff}	Pin Powers	
			RMS	Max		RMS	Max		RMS	Max		RMS	Max
1*	1.18889	-2	0.03%	0.11%	-26	0.43%	1.35%	-9	0.13%	0.42%	-16	0.24%	0.73%
2	1.18966	-4	0.06%	0.23%	-71	1.16%	3.57%	-18	0.28%	0.88%	-16	0.24%	0.74%
3	1.19048	-6	0.09%	0.33%	-114	1.86%	5.67%	-25	0.39%	1.19%	-16	0.24%	0.74%
4	1.19135	-8	0.11%	0.40%	-153	2.50%	7.48%	-30	0.46%	1.38%	-17	0.24%	0.73%
5	1.19227	-9	0.13%	0.45%	-185	3.01%	8.84%	-33	0.50%	1.47%	-16	0.23%	0.70%
6	1.19324	-10	0.14%	0.47%	-205	3.32%	9.58%	-34	0.51%	1.47%	-15	0.22%	0.64%
7	1.19427	-10	0.15%	0.47%	-207	3.33%	9.43%	-33	0.49%	1.39%	-13	0.19%	0.55%
8	1.19538	-10	0.15%	0.46%	-182	2.89%	7.98%	-30	0.45%	1.22%	-11	0.15%	0.43%
9*	1.19666	-10	0.15%	0.42%	-110	1.71%	4.61%	-24	0.36%	0.96%	-7	0.10%	0.27%
Average	–	8	0.11%	0.37%	139	2.25%	6.50%	26	0.40%	1.15%	14	0.21%	0.61%

The final 2D problem is the same as the previous one, except that the southeast corner UO_2 assembly is rodded instead of the northwest assembly. Tables 6.13 and 6.14 show similar results to the previous two problems. For all methods, the errors are lower for this problem than for the first multi-assembly problem due to the difference in control rod position. The rods in this problem are in a lower power region, so the absolute differences in power in the region with the largest errors will automatically be smaller. This is seen in subray MOC and each of the other methods. With this taken into account, each of the methods performs similarly to the previous problem. Subray-0 is the best but nearly the same as Subray-1, Subray-2, and Subray-3. The Subplane + CP method performs worse, but still significantly reduces the cusping errors from almost 6.5% average maximum differences to less than 1%.

Table 6.15: Average Performance and Convergence Data for 2D C5G7 Core with Rodded SE Assembly

Method	Iterations	Runtime (s)	Speedup
Reference	24.4	1006	–
None	24.0	657	1.54
Subplane	25.3	636	1.59
Subplane + CP	25.3	629	1.60
Subray-0	25.3	798	1.26
Subray-1	25.0	833	1.21
Subray-2	25.0	855	1.18
Subray-3	25.0	878	1.15

The primary reason to show this problem is that because the rods are in the center of the model, the long rays duplicated for the subray MOC calculation are longer for some angles, especially those between the northeast and southwest corners of the model. Thus, while the total number of rays duplicated is the same as for the previous problem (Table 6.11), the amount of work done for the duplicated rays is different. This will show what, if any, impact the position of the control rods in the core could have on the efficiency of the subray MOC calculations.

Table 6.15 shows the convergence and runtime data for this problem. Compared with the previous 2D core problem, the speedups are similar, but lower by 3-5% for each subray MOC calculation. The convergence behavior of this problem is similar to the previous one, so this small decrease in speedup is likely due to duplicating longer core rays. Considerable fluctuation can occur from one calculation to the next when using a development machine shared by other users, but the consistency in these calculations indicates that there will not be any major performance penalties based on the position of the rods in the core. Performance will be based primarily on the number of pins requiring subray treatment.

6.3.3 3D C5G7

In addition to the single-plane problems in the previous section, 3D problems were also set up to test the subray MOC calculations. A 3D variation of each of the single-plane problems will be presented in this section. The 3D problems have a 42.84 cm active fuel height, divided into 3 MOC planes for the decusping methods and 4 MOC planes for the reference calculation. Additionally, a single 21.42 cm axial reflector plane is positioned above the active fuel neight. Following a similar procedure to the previous problems, the control rod is withdrawn through the entire active fuel region in 30 steps of 1.428 cm. Eigenvalue and power distribution comparisons can then be made between the refined mesh reference solution and each of the multigrid methods.

6.3.3.1 3D C5G7 UO_2 Assembly

Table 6.16: 3D C5G7 UO_2 assembly rod withdrawal eigenvalue and pin power difference summary

Case	Method	k_{eff}	Pin Powers	
		Diff.	RMS	Max
Average	None	2193	6.05%	10.95%
	Subplane	222	0.88%	1.64%
	Subplane+CP	114	0.45%	0.84%
	Subray-0	52	0.25%	0.54%
	Subray-1	56	0.25%	0.55%
	Subray-2	56	0.25%	0.54%
	Subray-3	56	0.25%	0.54%
Position 8	None	-91	2.88%	4.19%
	Subplane	-319	1.51%	2.66%
	Subplane + CP	-106	0.52%	0.89%
	Subray-0	-104	0.53%	0.94%
	Subray-1	-104	0.52%	0.94%
	Subray-2	-105	0.53%	0.98%
	Subray-3	-105	0.53%	0.98%

The first problem is a 3D assembly. This problem uses the UO_2 assembly from the C5G7 core and adds control rods to it. Table 6.16 shows the average errors for each method for this problem for 27 different rod positions; positions 0, 10, 20, and 30 are exlucded from the averages because they align with the MOC planes and have no cusping effects. With the introduction of multiple MOC planes, each subray MOC variation performs nearly identically to the others. With multiple MOC planes, more axial effects are introduced, slighlty reducing the relative importance of some of the approximations made during the radial calculations. As with the 2D calculations, subray MOC performs better than the other methods. The k_{eff} errors are better by a factor of 2, while the power distribution errors are also much improved. The maximum power distribution errors for subray MOC tend to be comparable with the RMS power errors for Subplane + CP, which is a significant improvement in the solution.

Table 6.17: Average Performance and Convergence Data for 3D C5G7 Rodded UO$_2$ Assembly

Method	Iterations	Runtime (s)	Speedup
Reference	27.5	490	–
None	27.8	361	1.36
Subplane	27.5	374	1.32
Subplane + CP	27.2	353	1.39
Subray-0	27.6	444	1.11
Subray-1	28.7	464	1.07
Subray-2	28.8	475	1.04
Subray-3	28.8	478	1.03

Table 6.16 also shows the results for each method at control position 8, which is where subray MOC gives the worst power distribution results. In this case, Subplane + CP narrowly beats out subray MOC in the power comparisons. However, the differences between the methods is negligible, so even in the worst case, subray MOC is about as good as Subplane + CP and still much better than Subplane without CP. The results with no decusping method also indicate that this is a region where the cusping effects are not as large compared to the average of all the rod positions. Thus, the simpler approximation is helped by the less severe cusping effects. The worst case power error for Subray-0 is 0.94%, which compares favorably with the average maximum difference of 0.84% for Subplane + CP.

The 3D assembly has the same long ray parameters as those shown in Table 6.7 for the 2D assembly. For the plane with the rod, the number of long rays matches with the subray entries of the table, while for the remaining planes the number of long rays matches the reference entry. Because of this, the subray MOC calculation runtimes should be closer to the subplane calculation times than to the reference calculation, since 3 of the 4 planes are the same as the reference case. Table 6.17 shows the convergence and runtime results for each of the methods. The Subplane and Subplane + CP methods show speedups between 1.3 and 1.4. Using 4 planes instead of 5 should produce a speedup of approximately 1.25. This is lower than observed due to other differences in initialization, machine load during these calculations, and other minor factors. The subray MOC calculations each show worse speedup than the other methods due to the duplication of long rays in the partially rodded plane. There is also a small increase in the average number of iterations that contributes somewhat to the increased runtime. However, each subray MOC variation still results in a speedup greater than 1.0, incidicating that there is still some savings associated with it.

6.3.3.2 3D Core – Rodded Center Assembly

Table 6.18: 3D C5G7 core with rodded center assembly rod withdrawal eigenvalue and pin power difference summary

Case	Method	k_{eff} Diff.	Pin Powers RMS	Pin Powers Max
Average	None	21	6.62%	29.30%
	Subplane	21	0.69%	3.47%
	Subplane+CP	21	0.34%	1.69%
	Subray-0	21	0.20%	1.06%
	Subray-1	25	0.20%	1.14%
	Subray-2	25	0.20%	1.11%
	Subray-3	21	0.20%	1.11%
Position 16	None	-1730	12.62%	55.69%
	Subplane	-183	1.08%	5.61%
	Subplane + CP	-76	0.45%	2.38%
	Subray-0	-46	0.30%	1.76%
	Subray-1	-54	0.35%	2.01%
	Subray-2	-53	0.34%	1.97%
	Subray-3	-53	0.34%	1.96%

The second 3D problem is the 3D C5G7 core with the northwest UO_2 assembly rodded. The results for these calculations are shown in Table 6.18. The power distribution errors are quite large with no decusping method, with the maximum power being around 30% on average for all rod positions. The Subplane and Subplane + CP methods reduce that value to 3.47% and 1.69%, respectively. Finally, subray MOC reduces the maximum errors to just over 1%. The worst case scenario for this problem occurs at rod position 15. Unlike the 3D assembly, this position is one of the worst for all the methods, not just subray

MOC. Because of this, subray MOC performs best in every aspect, with the greatest power error for any rod position being 2.01% and the greatest k_{eff} error being -54 pcm, both for Subray-1.

With multiple assemblies in this 3D problem, a small trend begins to appear for the recombination flag. Subray-2 makes small improvements on Subray-1, and Subray-3 improves on Subray-2. While all 3 of these variations are worse than Subray-0, it does indicate that there is potential benefit to delaying recombination of the subrays. However, for that benefit to be fully realized, some improvements need to be made to the approximations in the coupling between the CMFD/P$_3$ system and the radial MOC sweeps.

Table 6.19: Average Performance and Convergence Data for 3D C5G7 Core with Rodded Center Assembly

Method	Iterations	Runtime (s)	Speedup
Reference	27.8	3256	–
None	28.2	2581	1.25
Subplane	29.1	2454	1.32
Subplane + CP	28.9	2454	1.32
Subray-0	28.8	2851	1.13
Subray-1	28.9	2920	1.11
Subray-2	29.1	2963	1.09
Subray-3	29.2	3009	1.08

For the 3D core, speedup is also seen for each of the decusping methods in Table 6.19. The Subplane methods show a speedup of around 30%, while each variation of subray MOC shows a speedup of approximately 10%. The subray MOC performance improves slightly on the single assembly cases due to the fact that a smaller fraction of long rays are being duplicated in the partially rodded plane. Each method shows a consistent increase in the number of 2D/1D iterations required to converge. The more complicated coupling between 2D MOC and the other solvers results in a slightly slower rate of convergence. However, eliminating an entire MOC plane outweighs the runtime penalty incurred from the small increase in 2D/1D iterations.

6.3.3.3 3D Core – Rodded Corner Assembly

The final problem is the 3D core with the rods in the southeast fuel assembly. The results for this problem are shown in Table 6.20. The k_{eff} difference are trivial for each method, but minor improvements are observed with increasing method complexity. As with all other problems before, the average behavior of subray MOC is significantly better than the other

Table 6.20: 3D C5G7 core with rodded corner assembly rod withdrawal eigenvalue and pin power difference summary

Case	Method	k_{eff} Diff.	Pin Powers RMS	Max
Average	None	72	1.53%	7.01%
	Subplane	8	0.18%	1.08%
	Subplane+CP	4	0.09%	0.56%
	Subray-0	2	0.05%	0.35%
	Subray-1	2	0.06%	0.37%
	Subray-2	2	0.06%	0.37%
	Subray-3	2	0.06%	0.37%
Position 16	None	-108	2.49%	12.25%
	Subplane	-15	0.29%	1.62%
	Subplane + CP	-6	0.12%	0.73%
	Subray-0	-4	0.08%	0.56%
	Subray-1	-4	0.10%	0.62%
	Subray-2	-4	0.10%	0.62%
	Subray-3	-4	0.10%	0.62%

methods, in this case brining the average maximum power difference under 0.4% and the average RMS difference to about 0.5%. The worst performance for subray MOC occurs at rod position 16. As with the previous problem, subray MOC outperforms the other methods at this position, though by a smaller margin. This position is one of the worst without any cusping method, and also has some of the largest errors for each method. However, the errors in this problem are smaller in general, so all methods perform well in reducing the cusping effects.

Finally, the performance and convergence data for this problem are shown in Table 6.21. The overall behavior is similar to the previous 3D core. The speedups for subray MOC are somewhat worse than before. However, the Subplane methods also show worse speedup, meaning that any differences between this problem and the previous one are attributable to changes in the state of the machine during the calculations.

Table 6.21: Average Performance and Convergence Data for 3D C5G7 Core with Rodded Corner Assembly

Method	Iterations	Runtime (s)	Speedup
Reference	27.2	3346	–
None	27.0	2751	1.22
Subplane	28.2	2716	1.18
Subplane + CP	28.2	2751	1.17
Subray-0	27.9	3115	1.08
Subray-1	28.1	3099	1.08
Subray-2	28.1	3132	1.07
Subray-3	28.2	3155	1.06

CHAPTER 7

Conclusions

7.1 Summary

To motivate the need to improve planar synthesis solutions to the transport method using subgrid methods, an overview of relevant discretizations, approximations, and numerical methods used to solve the Boltzmann transport equation was presented. Using this foundation in transport methods, an in depth description of the 2D/1D method was provided. This discussion provided a context to show the cause of the control rod cusping problem, which is the most severe axial heterogeneity observed in planar synthesis methods. While rod decusping methods have an extensive history, none of them have fully addressed the problem, necessitating the research presented here.

With the groundwork laid for 2D/1D and rod cusping, three new methods developed during this research were presented. The first, polynomial decusping, is reminiscent of those used in nodal codes. Pregenerated data is functionalized and used to adjust the weighting of rodded and unrodded cross sections during the 2D/1D calculation. This is similar to cross section tabulation methods used by nodal codes and is specific to certain control rod types. It was shown that compared to no decusping method, this polynomial decusping caused significant improvement for most cases. However, the results were still worse than desired for a direct transport calculation such as 2D/1D, while a few results were unacceptably bad. Furthermore, additional data collection would be required to extend this method, making it cumbersome as a long-term, general solution to the rod cusping problem. This was shown in the fact that the polynomial decusping could not resolve the stainless steel tip in the VERA Progression Problems and could not be used at all for the C5G7 benchmark problems.

The second method was the subplane collision probabilities method. This method uses an auxiliary solver to generate improved flux profiles which are then used in the subplane CMFD and P_3 calculations to solve the problem on a refined axial mesh. This method significantly improved on the results of the polynomial decusping. Furthermore, because the axial and radial effects are handled directly during the 2D/1D iteration, this method was less susceptible to changes in rod type. The results for every problem showed that it performed consistently, significantly reducing the errors in power distribution caused by the control rods.

The final method is the subray method of characteristics. This method sought to address the fundamental quantity for which MOC is solving: the angular flux. By accounting for the rod during the MOC sweep and obtaining a more correct angular flux, the scalar flux and other derived quantities are more accurate as well. This method was then tested using the C5G7 Benchmark in 1D, 2D, and 3D. In 1D, subray MOC fully resolved the effects of the partially rodded cells. For the 2D and 3D C5G7 cases, subray MOC consistently performed better than the next best method, often by close to a factor of 2. The speedups obtained using subray MOC were less than those of other methods, but were about as good as expected.

Several conclusions can be reached about the methods presented in this research:

- The polynomial decusping method is limited in its ability to resolve partially inserted control rods. The rod materials in the model must have polynomials generated for them and implemented in the code for the method to work. Furthermore, it was shown that even with the appropriate materials, the the polynomial method was limited in its accuracy. It typically did well in predicting k_{eff} since the curves were generated based on reactivity, but the power distribution predictions suffered. However, its simplicity makes it an option to address rod cusping effects in the near term.

- The subplane collision probabilities proved to be able to handle a variety of rod types and provide good results for both k_{eff} and power distributions. The method was less susceptible to rod position in the core than the polynomial method as well, shown by the more consistent behavior of its differential rod worth curve.

- Subplane-based methods introduce some changes in the convergence and runtime behavior of the 2D/1D calculations. Subplane collision probabilities consistently saw an increase in the number of CMFD iterations required for each 2D/1D iteration. The issue of load balance in parallel can also be important when using the subplane scheme with large radial partitions.

- Rod cusping effects can be mostly eliminated by using a method like subray MOC. Because the angular flux calculations are improved, the results of the most challenging problems tested here were still good.

- The current set of approximations required to use subray MOC in the 2D/1D method make the recombination flag useless for values greater than 0. Furthermore, even if those approximations are improved, the 1D MOC results indicate that the accuracy improvements will be small and not worth the increased computational expense.

Based on these conclusions, recommendations can be made regarding the pursuit of subgrid methods. The polynomial method is not recommended except to deal with urgent rod cusping errors. It is not useful in dealing with other axial heterogeneities, and its accuracy is no reliable enough to be a long-term solution. The subplane collision probabilities method is recommended as a good solution to subgrid heterogeneity. This method can be applied to a wide variety of axial heterogeneity and performs relatively consistently from one problem to the next. It is also relatively straightforward to implement this method since it is a separate calculation conducted between the MOC and CMFD calculations. There are also some areas in which it could be modified to further improve accuracy. Finally, subray MOC is recommended as the best of these three methods. It consistently provided the best answers for every calculation and has the best theoretical foundation. However, because it is embedded directly into the MOC calculations and affects the iteration between the MOC and CMFd/P_3 calculations, it is significantly more complicated and expensive than the other two methods. If this method is used, better optimization should be pursued as well.

7.2 Future Work

7.2.1 Subgrid Method Applications

There are several types of applications that could be made with these subgrid methods that have not been investigated in this dissertation. First, these methods were developed and analyzed in the framework of 2D/1D. However, the 2D/3D method also uses 2D MOC planes to inform its 3D sweep. Because of this, 2D/3D can also realize performance improvements by reducing the number of MOC planes used in the calculation. Applying any of these subgrid methods to the MOC planes in 2D/3D should have results in benefits similar to those observed with the 2D/1D method.

Another topic of interest is applying these methods to other axial heterogeneities. Geometric components such as fuel end caps or spacer grids could be treated differently using subray MOC or subplane collision probabilities. Reactors with non-uniform fuel heights would also be a good opportunity to use some of these methods without introduce thin MOC planes. Enabling thermal-hydraulic feedback would also present the opportunity to use these methods to capture subgrid temperature and density distributions. Any heterogeneity that exists in many locations in the core would be a good application for the subplane collision probabilities method, while severe heterogeneities that tend to be only in specific regions of the core are good applications for subray MOC.

Finally, exercising these methods on a greater diversity of problems would further quantify their effectiveness in capture subgrid heterogeneity. Modeling rod ejection transient problems is a common application for rod decusping methods that could be addressed. Depletion calculations are important for reactor analysis and provide opportunities to test these methods as control rods move throughout a reactor's cycle.

7.2.2 Polynomial Decusping Improvements

The polynomial decusping method could be extended primarily by adding more control rod types. This would allow it to become more flexible and applied to a wider variety of problems. One example would be adding a curve for stainless steel, allowing the method to resolve the stainless steel tip used in many PWRs. Other control rods have followers that cannot currently be account for, some of which are even fuel. Generating more data for the polynomial decusping method would make it more flexible and provide an easy, efficient means of handling cusping effects for a variety of problems.

7.2.3 Subplane Collision Probabilities

Several improvements could be made to the subplane collision probabilities method. First, other solvers could be used to generate the pin cell flux profiles. The current 1D CP solver could be replaced by a 2D CP kernel that solves in the radial and axial directions to account for the local 3D effects of the rod; a 2D MOC pin cell solver could be used in each level to account for corner effects in the pin cell and incorporate higher order scattering; a fast 3D CP or MOC solver could even be used to obtain a 3D flux shape around the tip of the control rod. Any of these solvers would likely improve the solutions.

Second, the current CP implementation in MPACT solves only in cylindrical coordinates. This works well for most geometric components in PWRs, but there are situations where other geometries would be necessary. Expanding the CP kernel to be able to handle some other common geometries would also make it useful in more situations.

7.2.4 Subray MOC

Finally, there are a number of improvements that can be made to subray MOC. The first few improvements are features which have been implemented in 2D/1D codes already for regular 2D MOC. The current implementation in MPACT does not use any parallelism. Subray MOC could be decomposed by angle, space, or energy using distributed memory, or by ray using threading. Doing this would present some interesting challenges concerning parallel load balance for both subray MOC and subplane CMFD/P_3. Higher order scattering could also be introduced instead of assuming P_0/TCP_0 scattering. Cross section shielding calculations will also be required if subray MOC is to be used for more realistic problems. When subray MOC is used for the main solve, it should also be used for the shielding calculations to obtain shielding data for the subregions used by the subrays. This would allow subray MOC to be used on realistic problems such as the VERA Progression Problems instead of being limited to macroscopic cross sections such as those used in the C5G7 problems.

Another area in which subray MOC could be improved is the sweep algorithm and data structures. Because MPACT was intended for a more traditional 2D/1D approach, implementing subray MOC required duplication of the long rays. This introduces a significant amount of unnecessary calculation. Redesigning the ray tracing data to account for subrays would then allow the implementation of more efficient algorithms for the subray MOC sweep that do not duplicate calculations unnecessarily.

There are also improvements that could be made to the axial and radial transverse leakage sources. For the radial TL source, the $\hat{D}$ coupling coefficients are calculated on the full-height MOC plane when subplane CMFD is used to ensure stability. With subray MOC, the currents could conceivably be tallied on the subplane surfaces instead, providing a more accurate shape for these coefficients and improving the radial TL source for the axial solver. Doing this would also necessitate some improvements in the flux homogenization and projection. The implementation presented here neglects the subregion flux after the MOC calculation is finished, relying only on the subplane CMFD solutions to generate axial shapes for subsequent MOC calculations. Instead, changes should be made to incorporate the subregion fluxes, which would improve both the MOC source calculations and the radial $\hat{D}$ calculations.

The axial TL source could also be improved through the use of better approximations. The current approximation is a flat TL source, meaning that the source is the same for all flat source regions belonging to a pin cell. It is clear that the actual TL source would be very different in the fuel regions than in the moderator regions. Devising a strategy to account for this would improve all 2D/1D calculations (including those using the subgrid methods developed in this work), especially in the vicinity of a strongly absorbing control rod.

APPENDIX A

Supplemental Derivations

A.1 Simplified Spherical Harmonics

This section will provide a detailed derivation of the SP$_3$ equations. The SP$_1$ equations are easily obtained by neglecting higher order moments of the flux, and SP$_N$ approximations of order greater than 3 are easily obtained by following the same procedure. These higher order approximations are not of interest because they were not used in this work and only marginally improve on the accuracy of SP$_3$.

To begin, we start with the mono-energetic planar geometry transport equation with anisotropic scattering:

$$\mu \frac{\partial \psi}{\partial x} + \Sigma_t(x)\psi(x,\mu) = \int_{-1}^{1} \Sigma_s(x,\mu,\mu')\psi(x,\mu')\,d\mu' + \frac{Q(x)}{2} \ . \tag{A.1}$$

Now we assume that the angular flux and scattering cross sections are both expanded in terms of Legendre polynomials, with these expansions truncated at the Nth term as follows:

$$\psi(x,\mu) \approx \sum_{n=0}^{N} \frac{2n+1}{2}\psi_n(x)P_n(\mu) \ , \tag{A.2a}$$

$$\Sigma_s(x,\mu,\mu') \approx \sum_{n=0}^{N} \frac{2n+1}{2}P_n(\mu)P_n(\mu')\Sigma_{s,n}(x) \ . \tag{A.2b}$$

Substituting these expansions into the transport equation, multiplying by $P_n(\mu)$, and integrating, we obtain the following:

$$\frac{d}{dx}\left[\frac{n+1}{2n+1}\psi_{n+1}(x) + \frac{n}{2n+1}\psi_{n-1}(x)\right] + \Sigma_t(x)\psi_n(x) = \Sigma_{s,n}(x)\psi_n(x) + Q(x)\delta_{0,n} \ . \tag{A.3}$$

To obtain the SP$_3$ equations, we solve Equation A.3 for moments 0-3:

$$\text{n=0:} \quad \frac{d\psi_1}{dx} + \Sigma_t(x)\psi_0(x) = \Sigma_{s,0}(x)\psi_0(x) + Q(x)$$

$$\frac{d\psi_1}{dx} = Q - (\Sigma_t(x) - \Sigma_{s0}(x))\psi_0(x) \ , \tag{A.4a}$$

$$\text{n=1:} \quad \frac{d}{dx}\left[\frac{2}{3}\psi_2(x) + \frac{1}{3}\psi_0(x)\right] + \Sigma_t(x)\psi_1(x) = \Sigma_{s,1}(x)\psi_1(x)$$

$$\psi_1(x) = -D_0\frac{d}{dx}\left[2\psi_2(x) + \psi_0(x)\right] \ , \tag{A.4b}$$

$$\text{n=2:} \quad \frac{d}{dx}\left[\frac{3}{5}\psi_3(x) + \frac{2}{5}\psi_1(x)\right] + \Sigma_t(x)\psi_2(x) = \Sigma_{s,2}(x)\psi_2(x)$$

$$\frac{d\psi_3}{dx} = -\frac{5}{3}(\Sigma_t(x) - \Sigma_{s2}(x))\psi_2(x) - \frac{2}{3}\left[Q(x) - (\Sigma_t(x) - \Sigma_{s0}(x))\psi_0(x)\right] \ , \tag{A.4c}$$

$$\text{n=3:} \quad \frac{3}{7}\frac{d\psi_2}{dx} + \Sigma_t(x)\psi_3(x) = \Sigma_{s,3}(x)\psi_3(x)$$

$$\psi_3(x) = -\frac{5}{3}D_2(x)\frac{d\psi_2}{dx} \ , \tag{A.4d}$$

$$D_0(x) = \frac{1}{3(\Sigma_t(x) - \Sigma_{s1}(x))} \ , \tag{A.4e}$$

$$D_2(x) = \frac{9}{35(\Sigma_t(x) - \Sigma_{s3}(x))} \ , \tag{A.4f}$$

where Equation A.4c was obtained by substituting Equation A.4a into it.

Now we make the following definitions:

$$\Phi_0(x) = \psi_0(x) - 2\psi_2(x) \ , \tag{A.5a}$$

$$\Phi_2(x) = \psi_2(x) \ . \tag{A.5b}$$

Solving these equations for the 0th and 2nd moments gives the following:

$$\psi_0(x) = \Phi_0(x) - 2\Phi_2(x) \ , \tag{A.6a}$$

$$\psi_2(x) = \Phi_2(x) \ . \tag{A.6b}$$

Substituting these expressions into each of the moment equations gives the final form of the SP$_3$ equations. These forms of the 0th and 2nd moment equations were obtained by substituting the 1st and 3rd moment equations into the 0th and 2nd moment equations.

$$\text{n=0:}\quad -D_0(x)\frac{d^2\Phi_2}{dx^2}+(\Sigma_t(x)-\Sigma_{s0}(x))\Phi_0(x)=Q(x)+2(\Sigma_t(x)-\Sigma_{s0}(x))\Phi_2(x)\ , \tag{A.7a}$$

$$\text{n=1:}\quad \psi_1(x)=-D_0(x)\frac{d\Phi_0}{dx}\ , \tag{A.7b}$$

$$\text{n=2:}\quad -D_2(x)\frac{d^2\Phi_2}{dx^2}+\left(\frac{9}{5}\Sigma_t(x)-\Sigma_{s2}(x)-\frac{4}{5}\Sigma_{s0}(x)\right)\Phi_2(x)= -\frac{2}{5}\left[Q(x)-(\Sigma_t(x)-\Sigma_{s0}(x))\Phi_0(x)\right]\ , \tag{A.7c}$$

$$\text{n=3:}\quad \psi_3(x)=-\frac{5}{3}D_2(x)\frac{d\Phi_2}{dx}\ . \tag{A.7d}$$

Because the odd moments have been completely eliminated from the even moment equations, Equations A.7a and A.7c can be solved by iterating between the two equations without solving explicitly for the odd moments.

A.2 Method of Collision Probabilities

A more thorough derivation of the Collision Probabilities (CP) method is conducted in this section. The derivation begins by considering the probability of a neutron reaching a vertical line in space form its point of emission or scatter. From here, the derivation follows closely with [37] to eventually arrive at a linear system for the flux.

A.2.1 Derivation

To derive the transport matrix for a cylindrical pin cell, we begin by considering the fraction of neutrons from a point source which will reach a line whose closest distance to the point source is τ mean free paths. We will consider the point source to be an isotropic unit point source. The polar angle is θ and the azimuthal angle is α. The fraction of neutrons emitted into a specific direction $d\Omega$ about $\mathbf{\Omega}$ from the point source to the line in question is given by

$$\frac{d\Omega}{4\pi}=\frac{sin(\theta)d\theta d\alpha}{4\pi}\ . \tag{A.8}$$

Integrating this expression over the polar angle gives the probability that neutrons will be emitted from the source in $d\alpha$ about α and reach the line a distance of τ away:

$$\int_0^\pi e^{-\frac{\tau}{\sin\theta}}\frac{\sin\theta d\theta d\alpha}{4\pi}\ . \tag{A.9}$$

We are only concerned with the fraction of neutrons in $d\alpha$ which are also in $d\Omega$. The fraction of neutrons emitted into $d\alpha$ is given by $\frac{d\alpha}{2\pi}$, so if we divide the previous expression by this fraction, we obtain the probability that a neutron emitted form the source in direction $d\alpha$ about α will reach the line:

$$\begin{aligned} p(\tau) &= \frac{1}{2}\int_{0}^{\pi} e^{-\frac{\tau}{\sin\theta}} \sin\theta d\theta \\ &= \int_{0}^{\frac{\pi}{2}} e^{-\frac{\tau}{\sin\theta}} \sin\theta d\theta \\ &= Ki_2(\tau) \ , \end{aligned} \tag{A.10}$$

where $Ki_2(x)$ is the second-order Bickley-Naylor function. The Bickley-Naylor function can be defined as follows:

$$Ki_n(x) = \int_{0}^{\frac{\pi}{2}} \cos^{n-1}\theta e^{-\frac{x}{\cos\theta}} d\theta \ , \tag{A.11a}$$

$$\frac{dKi_n(x)}{dx} = -Ki_{n-1}(x) \ , \tag{A.11b}$$

$$\int_{a}^{b} Ki_n(y)\,dy = Ki_{n+1}(a) - Ki_{n+1}(b) \ . \tag{A.11c}$$

Using Equations A.10 and A.11, we can now determine the probability of a neutron which escapes from region i having its next collision in region j. This probability is given by the probability of the escaped neutron reaching the first edge of region j minus the probability of reaching the second edge of j:

$$p_{ij}(\tau,\alpha,y) = Ki_2\left(\tau_{ij} + \tau_j + \tau\right) - Ki_2\left(\tau_{ij} + \tau\right) \ , \tag{A.12}$$

where τ_{ij} is the number of mean free paths between i and j, τ_j is the number of mean free paths across j, and τ is the number of mean free paths from the neutron's point of emission to the edge of region i. The variable y is defined along an axis in the plane of the problem perpendicular to the direction of streaming. The combination of τ and y specify a specific point in region i for each angle α.

Next, we define a strip in i of length $t_i = \frac{\tau}{\Sigma_i}$ along the streaming direction with width dy. To obtain the fraction of neutrons born in this strip that collide in j, we integrate the strip and divide by its length:

$$\begin{aligned}
p_{ij}(\alpha,y) &= \frac{1}{t_i}\int_0^{t_i} p_{ij}(\tau,\alpha,y)\,dt \\
&= \frac{1}{t_i}\int_0^{t_i} Ki_2\big(\tau_{ij}+\tau\big) - Ki_2\big(\tau_{ij}+\tau_j+\tau\big)\,dt \\
&= \frac{1}{t_i}\int_0^{t_i} Ki_2\big(\tau_{ij}+\tau_i-\Sigma_i t\big) - Ki_2\big(\tau_{ij}+\tau_j+\tau_i-\Sigma_i t\big)\,dt\,. \qquad \text{(A.13)}
\end{aligned}$$

Now we apply a change of variables $x = \tau_{ij}+\tau_i-\Sigma_i t$. Doing this, we obtain

$$\begin{aligned}
p_{ij}(\alpha,y) &= -\frac{1}{\Sigma_i t_i}\int_{\tau_{ij}+\tau_i}^{\tau_{ij}} Ki_2(x) - Ki_2\big(x+\tau_j\big)\,dx \\
&= \frac{1}{\Sigma_i t_i}\Big[\Big(Ki_3\big(\tau_{ij}\big) - Ki_3\big(\tau_{ij}+\tau_i\big)\Big) - \Big(Ki_3\big(\tau_{ij}+\tau_i\big) - Ki_3\big(\tau_{ij}+\tau_i+\tau_j\big)\Big)\Big]\,. \qquad \text{(A.14)}
\end{aligned}$$

This expression can now be multiplied by the fraction of neutrons in each strip and integrated over y to obtain the total fraction of neutrons born anywhere in i that stream in direction α and collide in j:

$$\begin{aligned}
p_{ij}(\alpha) &= \int_{y_{min}(\alpha)}^{y_{max}(\alpha)} p_{ij}(\alpha,y)\,\frac{t_i}{V_i}\,dy \\
&= \frac{1}{\Sigma_i V_i}\int_{y_{min}(\alpha)}^{y_{max}(\alpha)} \Big[\Big(Ki_3\big(\tau_{ij}\big) + Ki_3\big(\tau_{ij}+\tau_i+\tau_j\big)\Big) - \Big(Ki_3\big(\tau_{ij}+\tau_i\big) + \Big(Ki_3\big(\tau_{ij}+\tau_j\big)\Big)\Big)\Big]\,dy\,.
\end{aligned}$$
$$\text{(A.15)}$$

Finally, we obtain element ij of the transport matrix by multiplying by the volume and cross section. When multiplied by ϕ_i, this gives the total contribution to ϕ_j from region i:

$$\begin{aligned}
P_{ij}(\alpha) &= \Sigma_i V_i p_{ij}(\alpha) \\
&= \int_{y_{min}(\alpha)}^{y_{max}(\alpha)} \Big[\Big(Ki_3\big(\tau_{ij}\big) + Ki_3\big(\tau_{ij}+\tau_i+\tau_j\big)\Big) - \Big(Ki_3\big(\tau_{ij}+\tau_i\big) + \Big(Ki_3\big(\tau_{ij}+\tau_j\big)\Big)\Big)\Big]\,dy\,.
\end{aligned}$$
$$\text{(A.16)}$$

Now that this probability has been derived, it can be integrated over all α to obtain the transport matrix elements for a specific geometry.

We must also handle the self-transport case, where $i = j$. Again we follow the same procedure and define the probability $p_{ii}(t,\alpha,y)$ as the probability the neutron reaches region i minus the probability it escapes region i. Since the neutron was born in region i, the first probability is 1. This gives the following expressions:

$$p_{ii}(t,\alpha,y) = 1 - Ki_2(\tau) \ , \tag{A.17a}$$

$$\begin{aligned} p_{ii}(\alpha,y) &= \frac{1}{t_i}\int_0^{t_i} p_{ii}(t,\alpha,y)\,dt \\ &= 1 - \frac{1}{\Sigma_i t_i}\left[Ki_3(0) - Ki_3(\tau_i)\right] = 1 - \frac{1}{\Sigma_i t_i}\left[\frac{\pi}{4} - Ki_3(\tau_i)\right] \ , \end{aligned} \tag{A.17b}$$

$$\begin{aligned} p_{ii}(\alpha) &= \int_{y_{min}(\alpha)}^{y_{max}(\alpha)} p_{ii}(\alpha,y)\frac{t_i dy}{V_i} \\ &= 1 - \frac{1}{\Sigma_i V_i}\int_{y_{min}(\alpha)}^{y_{max}(\alpha)}\left[\frac{\pi}{4} - Ki_3(\tau_i)\right]dy \ , \end{aligned} \tag{A.17c}$$

$$\begin{aligned} P_{ii}(\alpha) &= \Sigma_i V_i p_{ii}(\alpha) \\ &= \Sigma_i V_i - \int_{y_{min}(\alpha)}^{y_{max}(\alpha)}\left[\frac{\pi}{4} - Ki_3(\tau_i)\right]dy \ . \end{aligned} \tag{A.17d}$$

Now this self-transport kernel can be used with the kernel in A.16 to set up the full transport matrix for a problem. This matrix is dependent on the geometry of the problem, so it must be done for each unique problem being solved. The following section discusses the details of this process for a cylindrical pin cell.

A.2.2 CP in Cylindrical Coordinates

To obtain the transport matrix for a cylindrical pin cell, the pin cell must first be cylindricized. To do this, the moderator region around the outside of the fuel pin is changed to an annular ring which preserves the total volume of the cylinder. This allows the calculation to be 1D spatially. Secondly, to ensure that there is a sufficient source driving the problem, a fuel and moderator mixture can be placed in a ring beyond the moderator ring. This is especially important for using the CP method for decusping since the control rod pin cell has no fission source of its own to drive the problem.

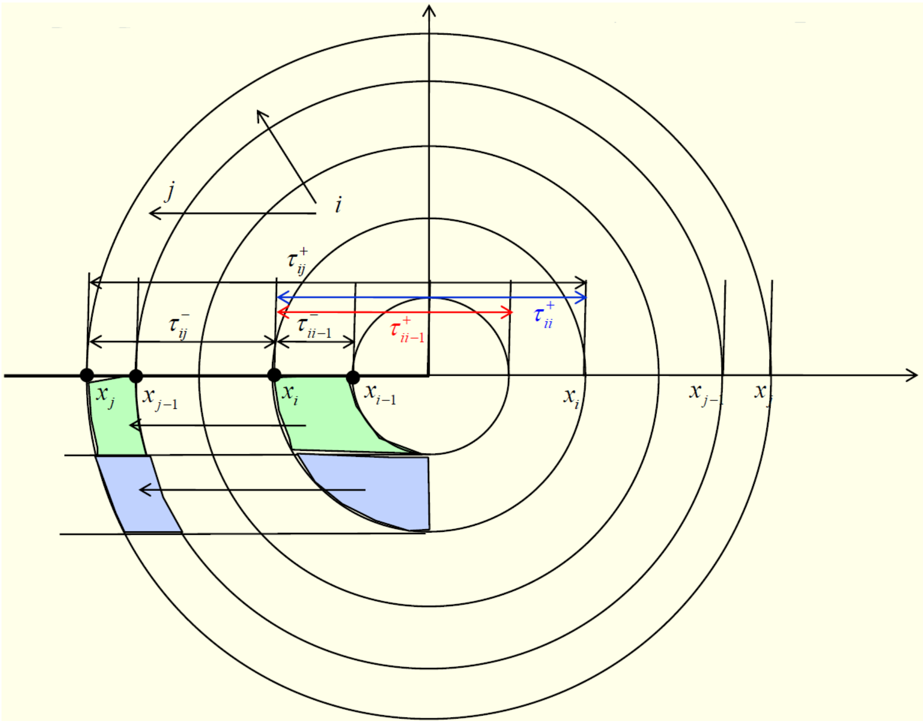


Figure A.1: Cylindrical Geometry Collision Probabilities

Now that the geometry is set up, we can take advantage of the symmetry of the problem and only model on quarter of the volume to simplify the calculation. The kernels from Equations A.16 and A.17 can be used, but some modification is required. The reason for these modifications is that from a ring i, a ring j which is outside ring i can be intersected from two different directions. Two "sub-kernels" are defined for teh positive and negative directions. Each covers only one of two directions, so they should be multiplied by $\frac{1}{2}$. However, each only accounts for $\frac{1}{4}$ the total volume, so the final expression $P_{ij}(y)$ should be multiplied by 2 to account for each of these. The sub-kernels are each multiplied by $\frac{1}{\Sigma_i V_i}$ to account for the unit source density in the volume V_i as well as a $\frac{1}{\Sigma_i}$ term that comes from the change of variables during the integration over t_i. The positive and negative τ terms are shown in the geometry in A.1.

$$P_{ij}^{-}(y) = \frac{1}{\Sigma_i V_i}\left(Ki_3\left(\tau_{ij-1}^{-}\right) + Ki_3\left(\tau_{i-1j}^{-}\right) - Ki_3\left(\tau_{ij}^{-}\right) - Ki_3\left(\tau_{i-1j-1}^{-}\right)\right) \tag{A.18a}$$

$$P_{ij}^{+}(y) = \frac{1}{\Sigma_i V_i}\left(Ki_3\left(\tau_{i-1j-1}^{+}\right) + Ki_3\left(\tau_{ij}^{+}\right) - Ki_3\left(\tau_{i-1j}^{+}\right) - Ki_3\left(\tau_{ij-1}^{+}\right)\right) \tag{A.18b}$$

$$P_{ij}(y) = 2\left(P_{ij}^{-}(y) + P_{ij}^{+}(y)\right) \; . \tag{A.18c}$$

Now we introduce notation to simplify this expression:

$$S_{ij} = \int_0^{R_i} \left(Ki_3\left(\tau_{ij}^+\right) - Ki_3\left(\tau_{ij}^-\right) \right) dy \tag{A.19a}$$

$$\begin{aligned} \Rightarrow P_{ij} &= \Sigma_i V_i \int_0^{R_i} P_{ij}(y)\, dy \\ &= 2\left(S_{ij} + S_{i-1j-1} - S_{ij-1} - S_{i-1j} \right) . \end{aligned} \tag{A.19b}$$

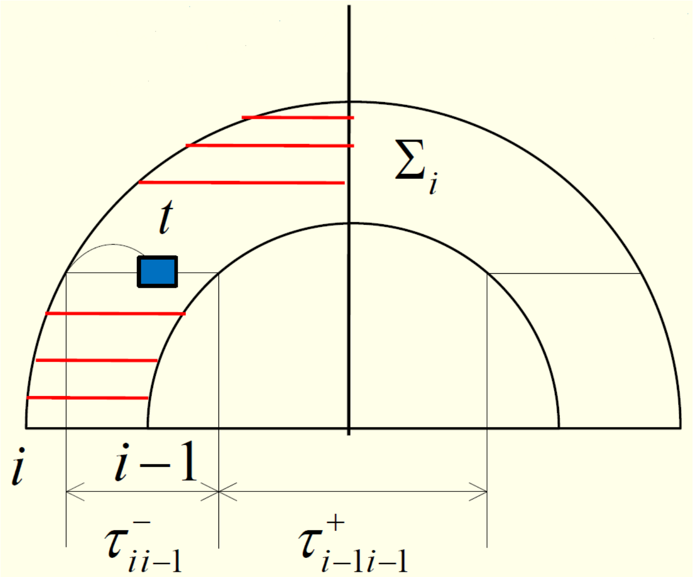


Figure A.2: Cylindrical Geometry Collision Probabilities Self-Transport

It is useful to note at this point that the principle of reciprocity exists for these kernels. What is meant by this is that $P_{ij} = P_{ji}$. This is because the S_{ij} terms are functions only of τ_{ij}^+ and τ_{ij}^-. These variables are defined as the distance between radii i and j, to $\tau_{ij}^+ = \tau_{ji}^+$ and $\tau_{ij}^- = \tau_{ji}^-$. Because of this, only half of the possible combinations of i and j must be calculated. Likewise, when the final transport matrix is set up, it will be symmetric, so only one of the upper and lower triangles must be explicitly calculated. This saves some time in the computation of the matrix elements.

A similar process is followed for the self-transport kernel. Again, positive and negative directions are set up. Additionally, the possibility of a neutron being born in i, escaping, then re-entering the other side of i must be accounted for as well. This is illustrated in

Figure A.2. This leads to the kernel in Equation A.20:

$$P^-_{ii}(y) = \frac{t_i(y)}{V_i} + \frac{1}{\Sigma_i V_i}\left(Ki_3\left(\tau^-_{ii-1}\right) + Ki_3\left(\tau^-_{i-1i}\right) - Ki_3\left(\tau^-_{ii}\right) - Ki_3\left(\tau^-_{i-1i-1}\right)\right) \tag{A.20a}$$

$$P^+_{ii}(y) = \frac{t_i(y)}{V_i} + \frac{1}{\Sigma_i V_i}\left(Ki_3\left(\tau^+_{ii}\right) + Ki_3\left(\tau^+_{i-1i-1}\right) - Ki_3\left(\tau^+_{ii-1}\right) - Ki_3\left(\tau^+_{i-1i}\right)\right) \tag{A.20b}$$

$$P_{ii}(y) = 2\left(P^-_{ii}(y) + P^+_{ii}(y)\right) \tag{A.20c}$$

$$\begin{aligned} P_{ii} &= \Sigma_i V_i \int\limits_0^{R_i} P_{ii}(y)\,dy \\ &= \Sigma_i V_i + 2\left(S\,ii + S_{\,i-1i-1} - S_{\,ii-1} - S_{\,i-1i}\right) \ . \end{aligned} \tag{A.20d}$$

P_{ij} is the probability that a neutron born in cell i has its first collision in cell j. Likewise, P_{ii} is the probability that a neutron born in cell i has its first collision in the same cell. To obtain the transport matrix elements T_{ij} and T_{ii}, we must consider the actual linear system we wish to solve. The goal is to find the reaction rates in each cell, from which we can easily find the flux in the cell ϕ_i. There are two main contributions to the reaction rates in cell i. The first contribution is from neutrons born in another cell j which have their first collision in cell i. This source is given by $\frac{Q_j}{\Sigma_{t,j}}$, which gives the contribution to the reaction rates in i when multiplied by P_{ji}. The second part of the source comes from neutrons which streamed into j and collided, scattered, then streamed into i before having their next collision. This source is given by $P_{ji}\phi_j c_j$, where c_j is the scattering ration in cell j, defined as $\frac{\Sigma_{s,j}}{\Sigma_{t,j}}$. The linear sysmem which needs to be solved is then shown in Equation A.21:

$$\Sigma_{t,i}\phi_i V_i = sum_{j=1}^{N_R} P_{ji}\left(c_j\phi_j + \frac{Q_j}{\Sigma_{t,j}}\right) \ . \tag{A.21}$$

The linear system, with the matrix and source elements, can now be explicitly defined in Equation A.22:

$$\underline{\underline{\boldsymbol{T}}}\underline{\boldsymbol{\phi}} = \underline{\boldsymbol{B}} \ , \tag{A.22a}$$

$$T_{ij} = -P_{ji}c_j = -2c_j\left(S_{\,ii} + S_{\,i-1j-1} - S_{\,ij-1} - S_{\,i-1j}\right) \tag{A.22b}$$

$$T_{ii} = \Sigma_{t,i}V_i - P_{ii}c_i = \Sigma_{t,i}V_i - \Sigma_{s,i}V_i - 2c_j\left(S_{\,ii} + S_{\,i-1j-1} - S_{\,ij-1} - S_{\,i-1j}\right) \ , \tag{A.22c}$$

$$B_i = \sum_{j=1}^{N_R} P_{ji}\frac{Q_j}{\Sigma_{t,j}} \ . \tag{A.22d}$$

The only remaining unknowns required to construct this linear system are the $S_{\,ij}$ terms that involve integrals over the third-order Bickley-Naylor functions. This will be discussed in the next section.

A.2.3 Bickely-Naylor Function Integration

In the previous sections, we defined the following functions which contain integrals over third-order Bickley-Naylor functions:

$$S_{ij} = \int_0^R \left(Ki_3\left(\tau_{ij}^{+}(y)\right) - Ki_3\left(\tau_{ij}^{-}(y)\right)\right) dy \ . \tag{A.23}$$

To numerically integrate these functions, we first break them up into regions:

$$S_{ij} = \sum_{k=1}^{i} S_{ij}^{k} \tag{A.24a}$$

$$S_{ij}^{k} = \int_{R_{k-1}}^{R_k} \left(Ki_3\left(\tau_{ij}^{+}(y)\right) - Ki_3\left(\tau_{ij}^{-}(y)\right)\right) dy \ . \tag{A.24b}$$

Next, we perform a coordinate transformation to change the bounds of integration to be on the interval $[-1,1]$. This allows us to apply a Gaussian quadrature to evaluate the integral numerically. The results of the transformation are shown in A.25, where $f(p)$ is the transformed S_{ij}^{k} and p_i and ω_i are some points and weights associated with the quadrature.

$$p = 2\frac{y - R_{k-1}}{R_k - R_{k-1}} - 1 = 2\frac{y - R_{k-1}}{\Delta_k} - 1 \ , \tag{A.25a}$$

$$dp = \frac{2}{\Delta_k} dy \ , \tag{A.25b}$$

$$\Rightarrow S_{ij}^{k} = \frac{\Delta_k}{2} \int_{-1}^{1} f(p)\, dp = \frac{\Delta_k}{2} \sum_{i} \omega_i f(p_i) \ . \tag{A.25c}$$

This can now be integrated using a standard Gaussian quadrature [54]. Because this quadrature is being applied to sections of each integral rather than the whole integral, MPACT uses a 4-point Gaussian quadrature (shown in Table A.1), which is sufficient to give accurate solutions over the entire bounds of the integral.

Table A.1: Four-Point Gaussian Quadrature

Point	Weight
$\pm\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18+\sqrt{30}}{36}$
$\pm\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18-\sqrt{30}}{36}$

APPENDIX B

C5G7 Cross-Sections

```
C5G7 macroscopic cross section data
 7 5
 2.0E+07 1.0E+06  5.0E+05 1.0E+03 1.0E+02 10. 0.0635
!
!Comments can appear after the first 3 lines and between macro/micro blocks
!
!In the second line, the first number is number of groups and the other is
!number of cross section sets.
!
!In the third line the energy group bounds are made up.
!
!Data here is derived from NEA/NSC/DOC(2003)16 or ISBN 92-64-02139-6
!Table 1 of Appendix A.
!
!The control rod cross sections come from NEA/NSC/DOC(2005)16 or
!ISBN 92-64-01069-6 Table 1 of Appendix A.
!
!For each cross-section set:
! XSMACRO <name> <scattering_order>
!  <absorption_xs> <nu-fission_xs> <kappa-fission_xs> <chi>  ! Repeats for each group
!  <scattering_matrix> !<ngroup>x<ngroup> matrix, repeated <scattering_order>+1 times
!

!Moderator
XSMACRO Moderator 0
  6.0105E-04 0.000000E+00 0.00000E+00 0.0000E+00
  1.5793E-05 0.000000E+00 0.00000E+00 0.0000E+00
  3.3716E-04 0.000000E+00 0.00000E+00 0.0000E+00
  1.9406E-03 0.000000E+00 0.00000E+00 0.0000E+00
  5.7416E-03 0.000000E+00 0.00000E+00 0.0000E+00
  1.5001E-02 0.000000E+00 0.00000E+00 0.0000E+00
  3.7239E-02 0.000000E+00 0.00000E+00 0.0000E+00
  4.44777E-02 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
  1.13400E-01 2.82334E-01 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
  7.23470E-04 1.29940E-01 3.45256E-01 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
  3.74990E-06 6.23400E-04 2.24570E-01 9.10284E-02 7.14370E-05 0.00000E+00 0.00000E+00
  5.31840E-08 4.80020E-05 1.69990E-02 4.15510E-01 1.39138E-01 2.21570E-03 0.00000E+00
  0.00000E+00 7.44860E-06 2.64430E-03 6.37320E-02 5.11820E-01 6.99913E-01 1.32440E-01
  0.00000E+00 1.04550E-06 5.03440E-04 1.21390E-02 6.12290E-02 5.37320E-01 2.48070E+00

```

```

!Guide tube
XSMACRO GuideTube 0
  5.1132E-04 0.000000E+00 0.00000E+00 0.0000E+00
  7.5801E-05 0.000000E+00 0.00000E+00 0.0000E+00
  3.1572E-04 0.000000E+00 0.00000E+00 0.0000E+00
  1.1582E-03 0.000000E+00 0.00000E+00 0.0000E+00
  3.3975E-03 0.000000E+00 0.00000E+00 0.0000E+00
  9.1878E-03 0.000000E+00 0.00000E+00 0.0000E+00
  2.3242E-02 0.000000E+00 0.00000E+00 0.0000E+00
  6.61659E-02 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
  5.90700E-02 2.40377E-01 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
  2.83340E-04 5.24350E-02 1.83297E-01 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
  1.46220E-06 2.49900E-04 9.23970E-02 7.88511E-02 3.73330E-05 0.00000E+00 0.00000E+00
  2.06420E-08 1.92390E-05 6.94460E-03 1.70140E-01 9.97372E-02 9.17260E-04 0.00000E+00
  0.00000E+00 2.98750E-06 1.08030E-03 2.58810E-02 2.06790E-01 3.16765E-01 4.97920E-02
  0.00000E+00 4.21400E-07 2.05670E-04 4.92970E-03 2.44780E-02 2.38770E-01 1.09912E+00


!Gap -- Actually just guide tube
XSMACRO Gap 0
  5.1132E-04 0.000000E+00 0.00000E+00 0.0000E+00
  7.5801E-05 0.000000E+00 0.00000E+00 0.0000E+00
  3.1572E-04 0.000000E+00 0.00000E+00 0.0000E+00
  1.1582E-03 0.000000E+00 0.00000E+00 0.0000E+00
  3.3975E-03 0.000000E+00 0.00000E+00 0.0000E+00
  9.1878E-03 0.000000E+00 0.00000E+00 0.0000E+00
  2.3242E-02 0.000000E+00 0.00000E+00 0.0000E+00
  6.61659E-02 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
  5.90700E-02 2.40377E-01 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
  2.83340E-04 5.24350E-02 1.83297E-01 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
  1.46220E-06 2.49900E-04 9.23970E-02 7.88511E-02 3.73330E-05 0.00000E+00 0.00000E+00
  2.06420E-08 1.92390E-05 6.94460E-03 1.70140E-01 9.97372E-02 9.17260E-04 0.00000E+00
  0.00000E+00 2.98750E-06 1.08030E-03 2.58810E-02 2.06790E-01 3.16765E-01 4.97920E-02
  0.00000E+00 4.21400E-07 2.05670E-04 4.92970E-03 2.44780E-02 2.38770E-01 1.09912E+00

!UO2 fuel-clad
XSMACRO UO2-3.3 0
  8.0248E-03 2.005998E-02 7.21206E-03 5.8791E-01
  3.7174E-03 2.027303E-03 8.19301E-04 4.1176E-01
  2.6769E-02 1.570599E-02 6.45320E-03 3.2990E-04
  9.6236E-02 4.518301E-02 1.85648E-02 1.0000E-07
  3.0020E-02 4.334208E-02 1.78084E-02 0.0000E+00
  1.1126E-01 2.020901E-01 8.30348E-02 0.0000E+00
  2.8278E-01 5.257105E-01 2.16004E-01 0.0000E+00
  1.27537E-01 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
  4.23780E-02 3.24456E-01 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
  9.43740E-06 1.63140E-03 4.50940E-01 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
  5.51630E-09 3.14270E-09 2.67920E-03 4.52565E-01 1.25250E-04 0.00000E+00 0.00000E+00
  0.00000E+00 0.00000E+00 0.00000E+00 5.56640E-03 2.71401E-01 1.29680E-03 0.00000E+00
  0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 1.02550E-02 2.65802E-01 8.54580E-03
  0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 1.00210E-08 1.68090E-02 2.73080E-01

!Control Rod
XSMACRO CRod 0
```

```
1.70490E-03 0.000000E+00 0.00000E+00 0.0000E+00
8.36224E-03 0.000000E+00 0.00000E+00 0.0000E+00
8.37901E-02 0.000000E+00 0.00000E+00 0.0000E+00
3.97797E-01 0.000000E+00 0.00000E+00 0.0000E+00
6.98763E-01 0.000000E+00 0.00000E+00 0.0000E+00
9.29508E-01 0.000000E+00 0.00000E+00 0.0000E+00
1.17836E+00 0.000000E+00 0.00000E+00 0.0000E+00
1.70563E-01 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
4.44012E-02 4.71050E-01 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
9.83670E-05 6.85480E-04 8.01859E-01 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
1.27786E-07 3.91395E-10 7.20132E-04 5.70752E-01 6.55562E-05 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00 1.46015E-03 2.07838E-01 1.02427E-03 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 3.81486E-03 2.02465E-01 3.53043E-03
0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 3.69760E-09 4.75290E-03 6.58597E-01
```

APPENDIX C

1D MOC Code

This appendix will show the full source code for the prototype 1D subray MOC code used to investigate the subray MOC method. Example scripts and inputs will then be given to show how the code can be used to generate subray MOC comparison data. The code was written using Matlab 2016 [58] and can be downloaded from `https://github.com/aarograh/dissertation`

C.1 Source Code

This section will provide the full source code used for all 1D MOC calculations presented in this dissertation. Each subsection will be titled with the name of the source file and include the text of the source.

C.1.1 inputClass.m

```
classdef inputClass < handle
    %INPUTCLASS Container class to hold input data
    %   Contains all data for pin geometry and materials,
    %   XS Library filename, and ray tracing and meshing
    %   parameters

    properties
        % The pin pitch in the cylindrical model
        pitch
        % The map of pin IDs for the model
        pinmap
        % The list of materials for each pin
        %   The number of rows should be equal to the highest pin ID used in pinmap
        %   The number of columns should be equal to the largest number of radial regions in any pin
        pinmats
        % The list of radii for each pin
        %   The number of rows should be the same as pinmats
        %   The number of columns should be one less than pinmats
        radii
        % The number of radial subdivisions for each material region
        %   The shape should be the same as pinmats
        pinmesh
        % Number of polar angles to use in the Gaussian quadrature
        npol
        % Name of cross section library file
        xsfilename
        % Boundary condition.  Must be a vector of strings with length 2.  Acceptable options are 'vaccum' and
        %   'refelcting'.  The first entry is for the left boundary, while the second is for the right.
        BCond
        % Maximum number of outer iterations to solve
        nouters
        % Convergence criteria.  Must be vector of doubles with length 2
        %   First element is for change in k-eff
        %   Second element is for maximum change in fission source
        conv_crit=[0, 0]
        % Option to toggle the amount of output produced by the code
        verbose=false
        % The number of mixtures to be produced by the code.  These mixtures are volume homogenizations of
        %   materials specified in the cross section library file
        nmixtures=0
        % The material IDs to be mixed
        %   The number of rows is equal to nmixtures
        %   Each row is a list of material IDs to be mixed.  The number of columns is equal to the maximum
        %   number of materials being mixed.  All 0s ending a row will be truncated.
        mixtures
        % The volume fractions used to mix the materials
        %   The shape is the same as the shape of mixtures.  Each element of mixvols gives the volume fraction
        %   for the material in the same position in mixtures.
        mixvols
        % Toggles subray MOC on or off
        %   0: No subray MOC.  Just performs a single 2D MOC sweep using homogenized cross sections where
        %   necessary.
        %   1: Performs separate sweeps during each iteration, then combines the solutions afterwards using
        %   volume fractions.  This is the same as using subray MOC everywhere.
        %   2: Performs subray MOC in the regions determined by npinSubTrack.
        subray=0
        % Determines how many neighboring pins will use subray MOC if subray is set to 2.  A value of 0 will
        %   use subray MOC only in the pin cells that have homogenized cross sections.  Larger values add pin
        %   cells to each side to resolve more effects near the partially inserted rod.  Defaults to a very
        %   large value to use subray MOC everywhere (which corresponds to subray=1).
        npinSubTrack=10000000
        % The following 4 options are unused, but left for future work and legacy input support
        diag % Not supported
        cmfd % CMFD not supported
        voleql=false % Ignored
        scattype % Only isotropic scattering is currently supported
    end

    methods
        function obj = inputClass( )
            %INPUTCLASS Creates an empty inputClass
            %   No attributes are initialized.  This
            %   simply creates the class and leaves
            %   filling the attributes to the user.

        end
    end

end
```

C.1.2 eigensolverClass.m

```
classdef eigensolverClass < handle
    %EIGENSOLVERCLASS Solves an eigenvalue problem
    %   This class contains the mesh, cross-section library,
    %   and quadrature required to solve a 1D MOC problem.
    %   It also contains methods to setup, solve, and checkConv
    %   the solution, as well as check for convergence.

    properties
        xsLib
        mesh
        quad
        solution
        fss
        accel
        cmfd
        conv_crit=[1.0e-8, 1.0e-7];
        input
        nouters=1000
        converged
        verbose=true
    end

    methods
        function obj = eigensolverClass( input )
            %EIGENSOLVERCLASS Sets up the eigensolver
            %   input - The inputClass container from which to initialize

            obj.xsLib = xsLibraryClass(input);
            obj.mesh = meshClass(input);
            obj.quad = quadratureClass(input);
            obj.fss = FixedSourceSolverClass(obj.xsLib, obj.quad, input);
            obj.solution = obj.fss.solution;
            if ~isempty(input.cmfd)
                if input.cmfd
                    obj.cmfd = cmfdClass(input, obj.mesh, obj.xsLib, obj.solution);
                    obj.accel = true;
                else
                    obj.accel = false;
                end
            else
                obj.accel = false;
            end
            if input.conv_crit(1) > 0
                obj.conv_crit(1) = input.conv_crit(1);
            end
            if input.conv_crit(2) > 0
                obj.conv_crit(2) = input.conv_crit(2);
            end
            obj.nouters = input.nouters;
            obj.converged = false;
            if ~isempty(input.verbose)
                obj.verbose = input.verbose;
            end

        end

        function obj = solve(obj)
            %SOLVE Solves the eigenvalue problem
            %   obj - The eigensolver object to solve

            for iouter=1:obj.nouters
                if obj.verbose
                    display(sprintf('Eigenvalue iteration %i',iouter));
                end
                if obj.accel
                    obj.cmfd.solve( );
                end
                obj.step();
                obj.checkConv();
                if obj.converged
                    if obj.verbose
                        display(sprintf('Converged after %i iterations...',iouter));
                    end
                    break
                elseif iouter == obj.nouters
                    if obj.verbose
                        display(sprintf('Reached maximum number of iterations...'));
                    end
                else
                    if obj.accel
                        obj.solution.keff(2) = obj.solution.keff(1);
                    end
                end
            end

        end

        function obj = step(obj)
            %STEP Performs a single iteration of the eigenvalue solve
            %   obj       - The eigensolver object to solve

            if ~obj.accel
                obj.solution.updateEig( );
            end
            obj.solution.calcFissSrc(obj.mesh, obj.xsLib);
            if obj.accel
                obj.fss.solve(1, 1);
            else
                obj.fss.solve(0, 1);
            end
            obj.solution.calcFissSrc(obj.mesh, obj.xsLib);

        end

        function obj = checkConv(obj)
            %CHECKCONV Updates eigenvalue object after each iteration
            %   obj - The eigensolver object whose convergence should be checked

            [conv_flux, conv_keff] = obj.solution.calcResidual( );
            if obj.verbose
                display(sprintf('Flux norm : %0.7f',conv_flux));
                display(sprintf('k-eff norm: %0.7f',conv_keff));
                display(sprintf('k-eff     : %0.7f\n',obj.solution.keff(1)));
            end
            if abs(conv_flux) < obj.conv_crit(2) && abs(conv_keff) < obj.conv_crit(1)
                obj.converged = true;
            end

        end
    end

end
```

C.1.3 FixedSourceSolverClass.m

```
classdef FixedSourceSolverClass < handle
    %FIXEDSOURCESOLVERCLASS Performs a fixed source calculation
    %   Uses a fixed fission source to perform a calculation for
    %   the scalar flux.

    properties
        xsLib
        mesh
        quad
        solution
        converged
        verbose=false
        relax=0.25
        subray=false
        submesh_vol
        nsubmesh=0
        npinSubTrack
        accel=false
        fipin
        bipin
    end

    methods
        function obj = FixedSourceSolverClass( varargin )
            %FIXEDSOURCESOLVERCLASS Initializes a fixedSourceSolverClass object
            %   varargin - Input arguments.  These can be "xsLib, quad, input" or
            %              "input, eig".  Anything else results in an error.
            %   xsLib - XSLibraryClass object
            %   quad  - QuadratureClass object
            %   input - inputClass object to initialize from
            %   eig   - eigensolverClass object to initialize from (optional; other arguments
            %           ignored if this one is present)

            if nargin == 2
                obj.mesh = meshClass(varargin{1});
                obj.xsLib = varargin{2}.xsLib;
                obj.quad = varargin{2}.quad;
                obj.solution = solutionClass(obj.mesh, obj.xsLib, varargin{1});
                obj.initFromEigenSolver( varargin{2});
                obj.verbose = varargin{1}.verbose;
                obj.subray = varargin{1}.subray;
                obj.npinSubTrack = varargin{1}.npinSubTrack;
            elseif nargin == 3
                obj.xsLib = varargin{1};
                obj.quad = varargin{2};
                obj.mesh = meshClass(varargin{3});
                obj.solution = solutionClass(obj.mesh, obj.xsLib, varargin{3});
                obj.verbose = varargin{3}.verbose;
                obj.subray = varargin{3}.subray;
                obj.npinSubTrack = varargin{3}.npinSubTrack;
            end
            if obj.subray > 0
                % Get number of submeshes.  Assume the same number or 0 everywhere
                for i=1:obj.mesh.nfsrcells
                    if obj.xsLib.xsSets(obj.mesh.materials(i)).nsubxs > 0
                        break
                    end
                end
                % Set submesh volume fractions
                obj.nsubmesh = length(obj.xsLib.xsSets(obj.mesh.materials(i)).subfracs);
                obj.submesh_vol = obj.xsLib.xsSets(obj.mesh.materials(i)).subfracs;
                % Split mesh into multiple meshes
                for i=1:obj.mesh.nfsrcells
                    matID = obj.mesh.materials(i);
                    nsubmesh = obj.xsLib.xsSets(matID).nsubxs;
                    if nsubmesh > 0
                        for j=1:nsubmesh
```

```
                            obj.mesh.materials(i,j) = obj.xsLib.xsSets(matID).subxs(j).ID;
                        end
                        obj.mesh.ipin(i) = -obj.mesh.ipin(i);
                    else
                        obj.mesh.materials(i,1:obj.nsubmesh) = matID;
                    end
                end
                % Set pin index flag to turn subray on and off
                if obj.nsubmesh > 1
                    pinids = zeros(max(obj.mesh.ipin),1);
                    for i=1:obj.mesh.nfsrcells
                        if obj.mesh.ipin(i) < 0
                            pinids(abs(obj.mesh.ipin(i))) = 1;
                        end
                    end
                    for i=1:obj.mesh.nfsrcells
                        if pinids(abs(obj.mesh.ipin(i)))
                            obj.mesh.ipin(i) = -abs(obj.mesh.ipin(i));
                        end
                    end
                    obj.fipin(1:obj.mesh.nfsrcells) = obj.mesh.ipin(:);
                    obj.bipin(1:obj.mesh.nfsrcells) = obj.mesh.ipin(:);
                    k = obj.mesh.nfsrcells+1;
                    fthispin = 0;
                    bthispin = max(abs(obj.bipin))+1;
                    fpinspast = obj.npinSubTrack+1;
                    bpinspast = obj.npinSubTrack+1;
                    for i=1:obj.mesh.nfsrcells
                        k = k-1;
                        flastpin = fthispin;
                        blastpin = bthispin;
                        fthispin = obj.mesh.ipin(i);
                        bthispin = obj.mesh.ipin(k);
                        if fthispin < 0
                            fpinspast = 0;
                        elseif fthispin ~= flastpin
                            fpinspast = fpinspast + 1;
                        end
                        if fpinspast <= obj.npinSubTrack
                            obj.fipin(i) = -abs(obj.fipin(i));
                        end
                        if bthispin < 0
                            bpinspast = 0;
                        elseif bthispin ~= blastpin
                            bpinspast = bpinspast + 1;
                        end
                        if bpinspast <= obj.npinSubTrack
                            obj.bipin(k) = -abs(obj.bipin(k));
                        end
                    end
                end
                % Initialize submesh flux to the scalar flux
                for j=1:obj.nsubmesh
```

```
                    obj.solution.submesh_scalflux(:,:,j) = obj.solution.scalflux(:,:,1);
                end
                obj.mesh.xstr(:,:,1:obj.nsubmesh) = 0.0;
                obj.mesh.source(:,:,1:obj.nsubmesh) = 0.0;
            % Initialize xstr since the mesh doesn't know how many groups there are
            else
                obj.nsubmesh = 1;
                obj.submesh_vol = 1.0;
                for j=1:obj.nsubmesh
                    obj.solution.submesh_scalflux(:,:,j) = obj.solution.scalflux(:,:,1);
                end
                obj.mesh.xstr(1:obj.xsLib.ngroups,1:obj.mesh.nfsrcells,1) = 0.0;
                obj.mesh.source(1:obj.xsLib.ngroups,1:obj.mesh.nfsrcells,1) = 0.0;
                obj.fipin(1:obj.mesh.nfsrcells) = 0;
                obj.bipin(1:obj.mesh.nfsrcells) = 0;
            end
        end

        function obj = initFromEigenSolver( obj, eig )
            %INITFROMEIGENSOLVER Initializes a fixedSourceSolverClass object from
            %an eigensolverClass object
            %   obj - fixedSourceSolverClass object to initialize
            %   eig - EigensolverClass object to use for initialization

            obj.solution.keff(:) = eig.fss.solution.keff(:);
            obj.solution.scalflux(:) = eig.fss.solution.scalflux(:);
            if (length(obj.solution.angflux(1,1,1,1,:)) == 1)
                obj.solution.angflux(:,:,:,:,1)=eig.fss.solution.angflux(:,:,:,:,1);
            elseif (length(eig.fss.solution.angflux(1,1,1,1,:)) == 1)
                for i=1:length(obj.solution.angflux(1,1,1,1,:))
                    obj.solution.angflux(:,:,:,:,i) = eig.fss.solution.angflux(:,:,:,:,1);
                end
            else
                obj.solution.angflux(:) = eig.fss.solution.angflux(:);
            end
            obj.solution.fisssrc(:) = eig.fss.solution.fisssrc(:);

        end

        function obj = solve( obj, wCur, ninners, convcrit )
            %SOLVE Solves a fixed source problem
            %   obj      - The FixedSourceSolverClass object to solve
            %   wCur     - Logical to tally currents (1) or not (0)
            %   ninners  - The number of inner iterations to perform
            %   convcrit - Convergence criteria for scattering source calculation

            obj.solution.updateBC();
            obj.converged = false;
            inner=0;
            if ~exist('convcrit','var')
                convcrit=1.0e-4;
            end

            while ~obj.converged
                inner = inner + 1;
                if wCur && ninners == inner
                    obj.accel = true;
                else
                    obj.accel = false;
                end
                obj.solution.scalflux(:,:,2) = obj.solution.scalflux(:,:,1);
                if obj.accel
                    obj.solution.current(:,:,2) = obj.solution.current(:,:,1);
                end
                if obj.subray
                    for i=1:obj.nsubmesh
                        obj.setup( i );
                    end
                    if obj.subray == 1
                        for i=1:obj.nsubmesh
                            obj.sweep( i );
                            obj.postprocess( i );
                        end
                        obj.postprocess( 0 );
                    elseif obj.subray == 2
                        obj.sweep_subray( );
                    end
                else
                    obj.setup( );
                    obj.sweep( );
                    obj.postprocess( );
                    obj.postprocess( 0 );
                end
                if obj.accel
                    obj.solution.scalflux(:,:,1) = obj.relax*obj.solution.scalflux(:,:,1) + ...
                        (1.0-obj.relax)*obj.solution.scalflux(:,:,2);
                    obj.solution.current(:,:,:,1) = obj.relax*obj.solution.current(:,:,:,1) + ...
                        (1.0-obj.relax)*obj.solution.current(:,:,:,2);
                end
                scatconv = obj.checkConv( );
                if obj.verbose
                    display(sprintf('Fixed Source Iteration %i - %g',inner,scatconv));
                end
                if inner == ninners || scatconv < convcrit
                    obj.converged = true;
                    if obj.verbose
                        display('Fixed Source Solve Converged.')
                    end
                end
            end

        end

        function obj = setup( obj, isubmesh )
```

```
            %SETUP_SUBRAY Sets up source and XS mesh for fixed source sub-ray MOC problem
            %   obj    - The FixedSourceSolverClass object to set up
            %   isubmesh - The submesh level being set up

            if ~exist('isubmesh','var')
                isubmesh = 1;
            end
            obj.mesh.source(:,:,isubmesh) = 0.0;
            for i=1:obj.mesh.nfsrcells
                matID = obj.mesh.materials(i,isubmesh);
                for j=1:obj.xsLib.ngroups
                    if obj.fipin(i) < 0 || obj.bipin(i) < 0
                        obj.mesh.source(j,i,isubmesh) = (obj.solution.fisssrc(i,1)*obj.xsLib.xsSets(matID).chi(j)/obj.solution.keff(1) + ...
                            obj.xsLib.xsSets(matID).scatter(j,:)*obj.solution.submesh_scalflux(:,i,isubmesh))*0.5;
                    else
                        obj.mesh.source(j,i,isubmesh) = (obj.solution.fisssrc(i,1)*obj.xsLib.xsSets(matID).chi(j)/obj.solution.keff(1) + ...
                            obj.xsLib.xsSets(matID).scatter(j,:)*obj.solution.scalflux(:,i,2))*0.5;
                    end
                    obj.mesh.xstr(j,i,isubmesh) = obj.xsLib.xsSets(matID).transport(j);
                end
            end

        end

        function obj = postprocess( obj, isubmesh )
            %POSTPROCESS_SUBRAY Post-processes the sweep result for sub-ray MOC
            %   obj     - The fixedsourcesolver object to post-process

            if ~exist('isubmesh','var')
                isubmesh = 1;
            end

            if isubmesh == 0
                obj.solution.scalflux(:,:,1) = 0.0;
                for i=1:obj.nsubmesh
                    obj.solution.scalflux(:,:,1) = obj.solution.scalflux(:,:,1) + ...
                    obj.submesh_vol(i)*obj.solution.submesh_scalflux(:,:,i);
                end
            else
                obj.solution.submesh_scalflux(:,:,isubmesh) = 0.0;
                for j=1:obj.mesh.nfsrcells
                    for k=1:obj.quad.npol
                        for g=1:obj.xsLib.ngroups
                            psibar = sum(sum(obj.solution.angflux(1:2,g,k,j:j+1,isubmesh),4),1)*0.5;
                            obj.solution.submesh_scalflux(g,j,isubmesh) = obj.solution.submesh_scalflux(g,j,isubmesh) + ...
                                psibar*obj.quad.weights(k);
                        end
                    end
                end
                if obj.accel
                    obj.solution.current(:,:,:,1) = 0.0;
                    for j=1:obj.mesh.nfsrcells+1;
                        for k=1:obj.quad.npol
                            for g=1:obj.xsLib.ngroups
                                psibar = (obj.solution.angflux(1,g,k,j,isubmesh)- ...
                                    obj.solution.angflux(2,g,k,j,isubmesh))*0.5;
                                obj.solution.current(g,j,isubmesh,1) = obj.solution.current(g,j,isubmesh,1) + ...
                                    psibar*obj.quad.cosines(k)*obj.quad.weights(k);
                            end
                        end
                    end
                end
            end

        end

        function obj = sweep( obj, isubmesh )
            %SWEEP Performs 1D MOC sweep for a single ray with multiple polars
            %   obj      - The fixedsourcesolver object to sweep
            %   isubmesh - The submesh level to sweep

            if ~exist('isubmesh','var')
                isubmesh = 1;
            end
            for i=1:obj.mesh.nfsrcells
                k = obj.mesh.nfsrcells-i+1;
                for j=1:obj.quad.npol
                    dx1 = (obj.mesh.fsredges(i+1) - obj.mesh.fsredges(i))/obj.quad.cosines(j);
                    dx2 = (obj.mesh.fsredges(k+1) - obj.mesh.fsredges(k))/obj.quad.cosines(j);
                    for igroup=1:obj.xsLib.ngroups
                        % Forward Sweep
                        exparg = exp(-obj.mesh.xstr(igroup,i,isubmesh)*dx1);
                        obj.solution.angflux(1,igroup,j,i+1,isubmesh) = ...
                            obj.solution.angflux(1,igroup,j,i,isubmesh)*exparg + ...
                            obj.mesh.source(igroup,i,isubmesh)/obj.mesh.xstr(igroup,i,isubmesh)*(1 - exparg);

                        % Backward Sweep
                        exparg = exp(-obj.mesh.xstr(igroup,k,isubmesh)*dx2);
                        obj.solution.angflux(2,igroup,j,k,isubmesh) = ...
                            obj.solution.angflux(2,igroup,j,k+1,isubmesh)*exparg + ...
                            obj.mesh.source(igroup,k,isubmesh)/obj.mesh.xstr(igroup,k,isubmesh)*(1 - exparg);
                    end
                end
            end
        end

        function obj = sweep_subray( obj )
            %SWEEP_SUBRAY Performs 1D MOC sweep using subray
            %   obj - The fixedsourcesolver object to sweep

            % Some initialization
            obj.solution.scalflux(:,:,1) = 0.0;
            obj.solution.submesh_scalflux(:) = 0.0;
            psi_in(1:obj.nsubmesh,1:2,1:obj.xsLib.ngroups,1:obj.quad.npol) = 0.0;
            source(1:obj.nsubmesh,1:2,1:obj.xsLib.ngroups) = 0.0;

            % Loop over all regions
            for i=1:obj.mesh.nfsrcells
                k = obj.mesh.nfsrcells-i+1;
                % If fipin is negative, we do subray in the forward direction
                if obj.fipin(i) < 0 || obj.bipin(i) < 0
                    lforwardSub = true;
                else
                    lforwardSub = false;
                end
                % If bipin is negative, we do subray in the backward direction
                if obj.bipin(k) < 0 || obj.fipin(k) < 0
                    lbackwardSub = true;
                else
                    lbackwardSub = false;
                end
                % Set boundary conditions and source
                for j=1:obj.quad.npol
                    for igroup=1:obj.xsLib.ngroups
                        for isubmesh=1:obj.nsubmesh
                            psi_in(isubmesh,1,igroup,j) = obj.solution.angflux(1,igroup,j,i,isubmesh);
                            psi_in(isubmesh,2,igroup,j) = obj.solution.angflux(2,igroup,j,k+1,isubmesh);
                        end
                    end
                end
                for igroup=1:obj.xsLib.ngroups
                    for isubmesh=1:obj.nsubmesh
                        source(isubmesh,1,igroup) = obj.mesh.source(igroup,i,isubmesh);
                        source(isubmesh,2,igroup) = obj.mesh.source(igroup,k,isubmesh);
                    end
                end
                % Modify source and boundary condition if subray shouldn't be used
                if ~lforwardSub
                    for j=1:obj.quad.npol
                        for igroup=1:obj.xsLib.ngroups
                            psi_in(:,1,igroup,j) = obj.submesh_vol*psi_in(:,1,igroup,j);
                        end
                    end
                    for igroup=1:obj.xsLib.ngroups
                        source(:,1,igroup) = obj.submesh_vol*source(:,1,igroup);
                    end
                end
                if ~lbackwardSub
                    for j=1:obj.quad.npol
                        for igroup=1:obj.xsLib.ngroups
                            psi_in(:,2,igroup,j) = obj.submesh_vol*psi_in(:,2,igroup,j);
                        end
                    end
                    for igroup=1:obj.xsLib.ngroups
                        source(:,2,igroup) = obj.submesh_vol*source(:,2,igroup);
                    end
                end
                for j=1:obj.quad.npol
                    dx1 = (obj.mesh.fsredges(i+1) - obj.mesh.fsredges(i))/obj.quad.cosines(j);
                    dx2 = (obj.mesh.fsredges(k+1) - obj.mesh.fsredges(k))/obj.quad.cosines(j);
                    for igroup=1:obj.xsLib.ngroups
                        % Forward sweep with submeshes
                        for isubmesh=1:obj.nsubmesh
                            % Forward Sweep
                            exparg = exp(-obj.mesh.xstr(igroup,i,isubmesh)*dx1);
                            obj.solution.angflux(1,igroup,j,i+1,isubmesh) = ...
                                psi_in(isubmesh,1,igroup,j)*exparg + ...
                                source(isubmesh,1,igroup)/obj.mesh.xstr(igroup,i,isubmesh)*(1.0-exparg);

                            psibar = sum(obj.solution.angflux(1,igroup,j,i:i+1,isubmesh),4)*0.5;
                            contribution = psibar*obj.quad.weights(j);
                            obj.solution.submesh_scalflux(igroup,i,isubmesh) = ...
                                obj.solution.submesh_scalflux(igroup,i,isubmesh) + contribution;
                            obj.solution.scalflux(igroup,i,1) = obj.solution.scalflux(igroup,i,1) + ...
                                contribution*obj.submesh_vol(isubmesh);

                            % Backward Sweep
                            exparg = exp(-obj.mesh.xstr(igroup,k,isubmesh)*dx2);
                            obj.solution.angflux(2,igroup,j,k,isubmesh) = ...
                                psi_in(isubmesh,2,igroup,j)*exparg + ...
                                source(isubmesh,2,igroup)/obj.mesh.xstr(igroup,k,isubmesh)*(1.0 - exparg);

                            psibar = sum(obj.solution.angflux(2,igroup,j,k:k+1,isubmesh),4)*0.5;
                            contribution = psibar*obj.quad.weights(j);
                            obj.solution.submesh_scalflux(igroup,k,isubmesh) = ...
                                obj.solution.submesh_scalflux(igroup,k,isubmesh) + contribution;
                            obj.solution.scalflux(igroup,k,1) = obj.solution.scalflux(igroup,k,1) + ...
                                contribution*obj.submesh_vol(isubmesh);
                        end
                    end
                end
            end

        end

        function maxdiff = checkConv( obj )
            %CHECKCONV Checks for convergence of the scattering source
            %   obj - FixedSourceSolverClass object whose convergence is being checked

            maxdiff = 0.0;
            for igroup=1:obj.xsLib.ngroups
                oldsource = zeros(obj.mesh.nfsrcells,1);
                newsource = zeros(obj.mesh.nfsrcells,1);
                for i=1:obj.mesh.nfsrcells
                    matID = obj.mesh.materials(i);
                    oldsource(i) = obj.xsLib.xsSets(matID).scatter(igroup,:)*obj.solution.scalflux(:,i,2);
                    newsource(i) = obj.xsLib.xsSets(matID).scatter(igroup,:)*obj.solution.scalflux(:,i,1);
                end
                % [oldsource,newsource]
                maxdiff = max(max(abs((oldsource - newsource)./oldsource)),maxdiff);
            end

        end
    end

end
```

C.1.4 meshClass.m

```
classdef meshClass < handle
    %MESHCLASS Object to store FSR and XS meshes and materials
    %   This object store three different types of meshes.
    %   The first is matmesh.  This contains the materials
    %   for each region.  Currently these materials are
    %   stored on the FSR mesh, but this will be switched
    %   and a method added to determine the material on
    %   the XS mesh.
    %
    %   The FSR mesh stores the position of cell edges for
    %   the mesh that will be ray traced.  The XS mesh
    %   stores cell edges for unique material regions, and
    %   is a subset of the XS mesh.

    properties
        nxscells
        nfsrcells
        materials
        ipin
        xsedges
```

```matlab
        fsredges
        xstr
        source
    end

    methods
        function obj = meshClass( input )
            %MESHCLASS Generates mesh object given geometry information
            %   input - The inputClass container from which to initialize

            % Initialize information
            npins = size(input.pinmap,2);
            nmats = size(input.pinmats,2);
            nfinecells = 0;
            ncoarsecells = 0;
            obj.fsredges(nfinecells+1) = 0.0;
            obj.xsedges(ncoarsecells+1) = 0.0;
            hpitch = input.pitch/2.0;
            if input.diag
                hpitch = sqrt(2)*hpitch;
            end

            % Loop over pins
            for i=1:npins
                % Get index of last material for this cell
                for j=nmats:-1:1
                    if input.pinmats(input.pinmap(i),j) ~=0
                        nreg = j;
                        break
                    end
                end

                % Loop out -> in over cell descriptions
                for j=nreg:-1:1
                    ncoarsecells = ncoarsecells+1;
                    % Outermost region needs to use pin hpitch
                    if isempty(input.radii)
                        width = hpitch;
                    elseif j == nreg
                        width = (1.0 - pi*input.radii(input.pinmap(i),j-1)^2.0/(input.pitch^2.0))*hpitch;
                    elseif j == 1
                        width = input.radii(input.pinmap(i),j);
                        width = pi*width^2.0/(2.0*input.pitch);
                    else
                         width = pi*(input.radii(input.pinmap(i),j)^2 - input.radii(input.pinmap(i),j-1)^2);
                         width = width/(2.0*input.pitch);
                    end
                    % Set coarse mesh
                    obj.xsedges(ncoarsecells+1,1) = obj.xsedges(ncoarsecells) + width;
                    % Set material
                    oldnfinecells = nfinecells;
                    nfinecells = nfinecells + input.pinmesh(input.pinmap(i),j);
                    obj.materials(oldnfinecells+1:nfinecells,1) = input.pinmats(input.pinmap(i),j);
                    % Set fine mesh
                    obj.fsredges(oldnfinecells+1:nfinecells+1,1) = obj.xsedges(ncoarsecells):width/...
                        input.pinmesh(input.pinmap(i),j):obj.xsedges(ncoarsecells+1);
                    obj.ipin(oldnfinecells+1:nfinecells) = i;
                end
                % Loop in -> out over cell descriptions
                for j=1:nreg
                    ncoarsecells = ncoarsecells+1;
                    % Outermost region needs to use pin hpitch
                    if isempty(input.radii)
                        width = hpitch;
                    elseif j == nreg
                        width = (1.0 - pi*input.radii(input.pinmap(i),j-1)^2.0/(input.pitch^2.0))*hpitch;
                    elseif j == 1
                        width = input.radii(input.pinmap(i),j);
                        width = pi*width^2.0/(2.0*input.pitch);
                    else
                         width = pi*(input.radii(input.pinmap(i),j)^2 - input.radii(input.pinmap(i),j-1)^2);
                         width = width/(2.0*input.pitch);
                    end
                    % Set coarse mesh
                    obj.xsedges(ncoarsecells+1,1) = obj.xsedges(ncoarsecells) + width;
                    % Set material
                    oldnfinecells = nfinecells;
                    nfinecells = nfinecells + input.pinmesh(input.pinmap(i),j);
                    obj.materials(oldnfinecells+1:nfinecells,1) = input.pinmats(input.pinmap(i),j);
                    % Set fine mesh
                    obj.fsredges(oldnfinecells+1:nfinecells+1,1) = obj.xsedges(ncoarsecells):width/...
                        input.pinmesh(input.pinmap(i),j):obj.xsedges(ncoarsecells+1);
                    obj.ipin(oldnfinecells+1:nfinecells) = i;
                end
            end
            obj.nxscells = ncoarsecells;
            obj.nfsrcells = nfinecells;

        end
    end

end
```

C.1.5 quadratureClass.m

```
classdef quadratureClass < handle
    %QUADRATURECLASS Contains quadrature information
    %   Contains the angles, their cosines, and weights
    %   for a polar quadrature for 1D MOC

    properties
        npol
        angles
        cosines
        weights
    end

    methods
        function obj = quadratureClass(input)
            %QUADRATURECLASS Sets up a quadrature class
            %   input - The inputClass container from which to intiialize

            obj.npol = input.npol/2;
            [obj.cosines, obj.weights] = createQuadrature(input.npol);
            obj.angles = acos(obj.cosines);
        end
    end

end

function [mu, w] = createQuadrature(N)
   %N is ordinate set: 2, 4, 8, ...
   %Return symmetric gauss-legendre quadrature set
   % syntax: [mu w] = createQuadrature( 4 )
   if (N == 2)
      mu = [.577350269189626]; %#ok<NBRAK>
      w  = [1.0]; %#ok<NBRAK>
   elseif (N == 4)
      mu = [.339981043584856 .861136311594053];
      w  = [.652145154862546 .347854845137454];
   elseif (N == 8)
      mu = [.183434642495650 .525532409916329 .796666477413627 .960289856497536];
      w  = [.362683783378363 .313706645877887 .222381034453374 .101228536290376];
   elseif (N == 16)
      mu = [0.989400934991650,0.944575023073233,0.865631202387832,...
         0.755404408355003,0.617876244402644,0.458016777657227,...
         0.281603550779259,0.0950125098376370;];
      w = [0.0271524594117540,0.0622535239386480,0.0951585116824930,...
         0.124628971255534,0.149595988816577,0.169156519395003,...
         0.182603415044924,0.189450610455067;];
   elseif (N == 32)
      mu = [0.0483076656877380,0.144471961582796,0.239287362252137,...
         0.331868602282128,0.421351276130635,0.506899908932229,...
         0.587715757240762,0.663044266930215,0.732182118740290,...
         0.794483795967942,0.849367613732570,0.896321155766052,...
         0.934906075937740,0.964762255587506,0.985611511545268,...
         0.997263861849482];
      w = [0.0965400885147260,...
         0.0956387200792750,0.0938443990808050,0.0911738786957640,...
         0.0876520930044040,0.0833119242269470,0.0781938957870700,...
         0.0723457941088490,0.0658222227763620,0.0586840934785360,...
         0.0509980592623760,0.0428358980222270,0.0342738629130210,...
         0.0253920653092620,0.0162743947309060,0.007018610009470];
   else
      error('Sorry, invalid N for quadrature')
   end

   assert(length(mu) == length(w));
   assert(abs(sum(w) - 1.0) < 2*eps);
   assert(all(mu > 0.0) && all(mu < 1.0));
   assert(all(w > 0));
end
```

C.1.6 solutionClass.m

```
classdef solutionClass < handle
    %SOLUTIONCLASS Stores angular and scalar flux solution variables
    %   This is mainly to make it easier to pass information around.
    %   It will also be used to check convergence for eigenvalue
    %   problems.

    properties
        BCond
        keff % First index is current, second is previous iteration
        angflux
        current
        scalflux % First index is current, second is previous iteration
        fisssrc
        submesh_scalflux
    end

    methods
        function obj = solutionClass( mesh,xsLib,input )
            %SOLUTIONCLASS Constructor for solutionClass
            %   mesh  - The mesh
            %   xsLib - The XS library
            %   input - The iput Class container from which to initialize

            if nargin == 3
                nsubmesh = 1;
                if input.subray
                    for i=1:mesh.nfsrcells
                        if xsLib.xsSets(mesh.materials(i)).nsubxs > 0
                            nsubmesh = xsLib.xsSets(mesh.materials(i)).nsubxs;
                            break
                        end
                    end
                end
                obj.keff(1:2) = 1.0;
                obj.angflux(1:2,1:xsLib.ngroups,1:input.npol/2,1:mesh.nfsrcells+1,1:nsubmesh) = 1.0;
                obj.current(1:xsLib.ngroups,1:mesh.nfsrcells+1,nsubmesh,1:2) = 0.0;
                obj.scalflux(1:xsLib.ngroups,1:mesh.nfsrcells,1:2) = 1.0;
                obj.BCond = input.BCond;
                obj.fisssrc(1:mesh.nfsrcells,1:2) = 1.0;
                if nsubmesh > 1
                    obj.submesh_scalflux(1:xsLib.ngroups,1:mesh.nfsrcells,nsubmesh) = 0.0;
                end
            end
        end

        function obj = updateBC( obj )
            %UPDATEBC Updates the solution to prepare for the next iteration
            %   obj - The solutionClass object to update

            % Set angular flux BC
            if ischar(obj.BCond(1))
                if strcmp(strtrim(obj.BCond(1,:)),'vacuum')
                    obj.angflux(1,:,:,1,:) = 0.0;
                elseif strcmp(strtrim(obj.BCond(1,:)),'reflecting')
                    obj.angflux(1,:,:,1,:) = obj.angflux(2,:,:,1,:);
                end
            end
            if ischar(obj.BCond(2))
                if strcmp(strtrim(obj.BCond(2,:)),'vacuum')
                    obj.angflux(2,:,:,end,:) = 0.0;
                elseif strcmp(strtrim(obj.BCond(2,:)),'reflecting')
                    obj.angflux(2,:,:,end,:) = obj.angflux(1,:,:,end,:);
                end
            end
            obj.angflux(1,:,:,2:end,:) = 0.0;
            obj.angflux(2,:,:,1:end-1,:) = 0.0;

        end

        function obj = updateEig( obj )
            %UPDATEEIG Calculates the new k-eff eigenvalue
            %   obj   - The solutionClass object to update

            obj.keff(2) = obj.keff(1);
            obj.keff(1) = obj.keff(2)*sum(obj.fisssrc(:,1))/sum(obj.fisssrc(:,2));

        end

        function [conv_flux, conv_keff] = calcResidual( obj )
            %CALCRESIDUAL Calculates the scalar flux and k-eff residuals
            %   obj   - The solutionClass object to check

            conv_flux = 0.0;
            for i=1:length(obj.fisssrc(:,1))
                conv_flux = max(conv_flux,abs((obj.fisssrc(i,1) - obj.fisssrc(i,2))));
            end
            conv_keff = obj.keff(1) - obj.keff(2);

        end

        function [ obj ] = calcFissSrc( obj, mesh, xsLib )
```

```
            %CALCFISSSRC Calculates the fission source in each cell
            %   obj   - The solution object to use for the FS calculation
            %   mesh  - The mesh to calculate the FS on
            %   xsLib - The XS Library to use for the calculation

            obj.fisssrc(:,2) = obj.fisssrc(:,1);
            obj.fisssrc(:,1) = 0.0;
            for i=1:mesh.nfsrcells
                matID = mesh.materials(i);
                obj.fisssrc(i,1) = xsLib.xsSets(matID).nufission*obj.scalflux(:,i,1);
            end
        end
    end

end
```

C.1.7 xsLibraryClass.m

```
classdef xsLibraryClass < handle
    %XSLIBRARYCLASS Stores cross-section data and reads XS library file
    %   Cross-section libraries must be in the same format as the
    %   MPACT user libraries.  One cross-section set will be stored for
    %   each material type.

    properties
        fileid
        name
        ngroups
        nsets
        groupBounds
        xsSets
    end

    methods
        function obj = xsLibraryClass( input )
            %XSLIBRARYCLASS Constructor for xsLibraryClass
            %   input - The inputClass container from which to initialize

            obj.openfile(input.xsfilename);

            % Get library name
            nextline = obj.getLine();
            obj.name = strrep(nextline,sprintf('\n'),'');

            % Get number of energy groups and materials
            nextline = obj.getLine();
            [obj.ngroups, obj.nsets] = strread(nextline);

            % Get energy group boundaries
            nextline = obj.getLine();
            obj.groupBounds = strread(nextline);

            for i=1:obj.nsets
                % Loop until we actually hit the XS set
                nextline = obj.getLine();
                k = strfind(nextline,'XSMACRO');
                while isempty(k) || nextline(1) == '!'
                    nextline = obj.getLine();
                    k = strfind(nextline,'XSMACRO');
                end

                % Read name and number of scattering moments
                tmpstr = strsplit(nextline);
                % Split the stupid cell array object into string and integer
                setname = strtrim(cell2mat(tmpstr(2)));
                setorder = strread(strtrim(cell2mat(tmpstr(3))));
                % This branching is needed to prevent matlab from assuming xsSets is a double array
                if i == 1
                    obj.xsSets = xsClass(i, setname, setorder);
                else
                    obj.xsSets(i) = xsClass(i, setname, setorder);
                end

                % Read in absorption, nufission, fission, and chi
                for k=1:obj.ngroups
                    nextline = obj.getLine();
                    [obj.xsSets(i).absorption(k), obj.xsSets(i).nufission(k), ...
                        obj.xsSets(i).fission(k), obj.xsSets(i).chi(k)] = strread(nextline);
                end
                % Loop over scattering orders
                for j=1:setorder+1
                    % Read in scattering matrix for order j-1
                    for k=1:obj.ngroups
                        nextline = obj.getLine();
                        obj.xsSets(i).scatter(k,1:obj.ngroups,j) = strread(nextline);
                    end
                end
                % Calculate total cross-section and transport cross-section
                obj.xsSets(i).calcTXS( input.scattype );
            end

            obj.closefile( );

            for j=1:input.nmixtures
                id = input.mixtures(j,1);
                name = sprintf('Mixture ');
                for k=2:length(input.mixtures(j,:))
                    if input.mixtures(j,k) == 0
                        break
                    else
                        if k > 2
                            name = sprintf('%s;',name);
                        end
                        name = sprintf('%s %0.3f%% %s',name,input.mixvols(j,k-1),...
                            obj.xsSets(input.mixtures(j,k)).name);
                    end
                end
                obj.xsSets(id) = xsClass(id, name);
                obj.xsSets(id).nsubxs = 0;
                obj.xsSets(id).subxs = xsClass();
                for k=2:length(input.mixtures(j,:))
                    if input.mixtures(j,k) == 0
                        break
                    else
                        obj.xsSets(id).nsubxs = obj.xsSets(id).nsubxs + 1;
                        obj.xsSets(id).subxs(k-1) = obj.xsSets(input.mixtures(j,k));
                    end
                end
                obj.xsSets(id).subfracs = input.mixvols(j,:);
                obj.mix( id );
            end
        end

        function obj = mix( obj, imix )
            %MIX Mixes sub-cross-sections
            %   obj     - the xsLibraryClass object
            %   imix    - the index of the mixture being set up

            % Initialize cross-sections and get scattering order
            obj.xsSets(imix).scatOrder = max(obj.xsSets(imix).subxs.scatOrder);
            obj.xsSets(imix).total(1:obj.ngroups) = 0;
            obj.xsSets(imix).transport(1:obj.ngroups) = 0;
            obj.xsSets(imix).absorption(1:obj.ngroups) = 0;
            obj.xsSets(imix).nufission(1:obj.ngroups) = 0;
            obj.xsSets(imix).fission(1:obj.ngroups) = 0;
            obj.xsSets(imix).chi(1:obj.ngroups) = 0;
            obj.xsSets(imix).scatter(1:obj.ngroups,1:obj.ngroups, ...
                obj.xsSets(imix).scatOrder+1) = 0;
            for i=1:obj.xsSets(imix).nsubxs

                obj.xsSets(imix).total = obj.xsSets(imix).total + ...
                    obj.xsSets(imix).subfracs(i)*obj.xsSets(imix).subxs(i).total;
                obj.xsSets(imix).transport = obj.xsSets(imix).transport + ...
                    obj.xsSets(imix).subfracs(i)*obj.xsSets(imix).subxs(i).transport;
                obj.xsSets(imix).absorption = obj.xsSets(imix).absorption + ...
                    obj.xsSets(imix).subfracs(i)*obj.xsSets(imix).subxs(i).absorption;
                obj.xsSets(imix).nufission = obj.xsSets(imix).nufission + ...
                    obj.xsSets(imix).subfracs(i)*obj.xsSets(imix).subxs(i).nufission;
                obj.xsSets(imix).fission = obj.xsSets(imix).fission + ...
                    obj.xsSets(imix).subfracs(i)*obj.xsSets(imix).subxs(i).fission;
                obj.xsSets(imix).chi = obj.xsSets(imix).chi + ...
                    obj.xsSets(imix).subfracs(i)*obj.xsSets(imix).subxs(i).chi;
                obj.xsSets(imix).scatter = obj.xsSets(imix).scatter + ...
                    obj.xsSets(imix).subfracs(i)*obj.xsSets(imix).subxs(i).scatter;
            end
        end

        function obj = openfile( obj, filename )
            %OPENFILE Opens the XS Library file
            %   obj      - The xsLibraryClass object
            %   filename - The XS Library filename

            obj.fileid = fopen(filename);

        end

        function obj = closefile( obj )
            %CLOSEFILE Closes the XS Library file
            %   obj      - The xsLibraryClass object

            fclose(obj.fileid);
            obj.fileid = 0;

        end

        function [ nextline, EOF ] = getLine( obj )
            %GETLINE Gets the next line from the XS Library file
            %   obj      - The xsLibraryClass object

            nextline = fgets(obj.fileid);
            if ischar(nextline)
                nextline = strrep(nextline,sprintf('\n'),'');
                EOF = 0;
            else
                EOF = 1;
            end

        end
    end

end
```

C.1.8 xsClass.m

```
classdef xsClass < handle
    %XSCLASS Class which hold cross-section data
    %   This class holds the multi-group XS data for
    %   a single type of material.

    properties
        ID
        name
        scatOrder
        total
        transport
        absorption
        nufission
        fission
        chi
        scatter % Scatter from column into row
        nsubxs=0
        subxs
        subfracs
    end

    methods
        function obj = xsClass( ID, name, order )
            %XSCLASS Consructor for xsClass
            %   ID    - ID for this XSSet
            %   name  - name of the XS set
            %   order - Scattering order for the xsSet

            if exist('ID')
                obj.ID = ID;
            end
            if exist('name')
                obj.name = name;
            end
            if exist('order')
                obj.scatOrder = order;
            end

        end

        function obj = calcTXS( obj, transOpt )
            %CALCTXS Calculates transport and total cross-sections
            %   obj - the XS set object
            %   transOpt - Transport correction option.  Currently only values
            %              of P0 are allowed.

            obj.total = obj.absorption + sum(obj.scatter,1);
            switch(transOpt)
                case('P0')
                    obj.transport = obj.total;
            end

        end
    end

end
```

C.2 Scripts

This section will provide a collection of scripts and an example input that will generate rod withdrawal comparisons for subray MOC. The withdrawal.m script is the top-level script and calls the other ones.

C.2.1 withdrawal.m

```
% TenPin7group_mix_rodded_unrodded
% ThreePin7group_mix_rodded_unrodded
p4a_centerAssem
percent = 10;
ncase = 0;
for i=percent:percent:100-percent
  ncase = ncase + 1;
  if ncase == 1
    getref = true;
  else
    getref = false;
  end
  input.mixvols = [i/100, (100-i)/100];
  input.subray = false;
  compare_subrayES;
    for j=1:length(fssSolver)
        solutions(j,ncase) = fssSolver(j).solution;
    end
  close all;
end

postprocess_withdrawal
```

C.2.2 p4a_centerAssem.m

```
close all; clear variables; clc;

%% General Input Data
% 1: Fuel Pin
% 2: Control Pin
% 3: Guide Tube Pin
input = inputClass();
input.pitch = 1.26;
input.diag = 0; % flat to indicate whether pin moves through narrow (0) or wide (1) water
% Mixtures
input.nmixtures = 1;
input.mixtures = [6, 1, 5];
input.mixvols = [0.5, 0.5];
% Pin information
input.pinmats = [6, 2, 1; % Mixture
    5, 2, 1; % Control Rod
    1, 2, 1; % Guide Tube
    4, 2, 1]; % Fuel

input.radii = [ 0.4, 0.475;
    0.4, 0.475;
    0.4, 0.475;
    0.4096, 0.475];
input.pinmesh = [ 3 1 2;
    3 1 2;
    3 1 2;
    3 1 2];
```

```
% Quadrature
input.npol = 16;
% XS Library Info
input.xsfilename = 'c5g7.xsl';
input.scattype = 'P0';
% Boundary Conditions
input.BCond = ['reflecting';'reflecting'];
% Convergence
input.nouters = 2000;
input.conv_crit = [1.0e-5 1.0e-5];
input.verbose = true;

%% Case 1 - 50-50 Mixutre Eigenvalue Case
 input.pinmap = [4, 4, 1, 4, 4, 1, 4, 4, 3, 4, 4, 1, 4, 4, 1, 4, 4];

%% set IDs
swappinids = [1,2,3];
```

C.2.3 compare_subrayES.m

```
% This script assumes that the inputClass input has been defined
% externally.  It also assumes that an array of pin IDs swappinids
% has been set up.  The first index is the mixed pin, the second
% the absorber pin, and the third the replacement pin
% Also assums that a getref variable has been set to true or false.
% True will cause the fully rodded/unrodded cases to run, while false
% will skip them.  This is used when this script is called repeatedly
% by another script.

%% Setup
oldmap = input.pinmap;

%% Volume-homogenized solve
eSolver(1) = eigensolverClass(input);
eSolver(1).solve();

if getref
  %% Fully rodded solve
  for i=1:length(input.pinmap)
      if oldmap(i) == 1
          input.pinmap(i) = swappinids(2);
      end
  end
  eSolver(2) = eigensolverClass(input);
  eSolver(2).solve();

  %% Fully unrodded solve
  for i=1:length(input.pinmap)
      if oldmap(i) == 1
          input.pinmap(i) = swappinids(3);
      end
  end
  eSolver(3) = eigensolverClass(input);
  eSolver(3).solve();
end

%% Sub-ray fixed source solve
for i=1:length(input.pinmap)
    if oldmap(i) == 1
        input.pinmap(i) = swappinids(1);
    end
end
input.subray = 1;
eSolver(4) = eigensolverClass(input);
eSolver(4).solve();

input.subray = 2;
input.npinSubTrack = 0;
eSolver(5) = eigensolverClass(input);
eSolver(5).solve();
input.npinSubTrack = 1;
eSolver(6) = eigensolverClass(input);
eSolver(6).solve();
input.npinSubTrack = 2;
eSolver(7) = eigensolverClass(input);
eSolver(7).solve();

% Post-process
for i=1:length(eSolver)
    fssSolver(i) = eSolver(i).fss;
end
```

C.2.4 postprocess_withdrawal.m

```
close all;
ncases = length(solutions(1,:));

for i=1:ncases+2
  keffs(2,i) = solutions(2,1).keff(1);
  keffs(3,i) = solutions(3,1).keff(1);
  if i == 1
    vfrac(i) = 0.0;
    keffs(1,i) = solutions(2,1).keff(1);
    keffs(4:7,i) = keffs(1,i);
  elseif i == ncases+2
    vfrac(i) = 100.0;
    keffs(1,i) = solutions(3,i-2).keff(1);
    keffs(4:7,i) = keffs(1,i);
  else
    vfrac(i) = (i-1)/(ncases+1)*100;
    keffs(1,i) = solutions(1,i-1).keff(1);
    keffs(4,i) = solutions(4,i-1).keff(1);
    keffs(5,i) = solutions(5,i-1).keff(1);
    keffs(6,i) = solutions(6,i-1).keff(1);
    keffs(7,i) = solutions(7,i-1).keff(1);
  end
end



isol = 5;
xgrid = 0.5*(fssSolver(1).mesh.fsredges(2:end)+fssSolver(1).mesh.fsredges(1:end-1));
for igroup=1:7
    figure(igroup)
    plot(xgrid,solutions(1,isol).scalflux(igroup,:,1),'k',xgrid,solutions(4,isol).scalflux(igroup,:,1),'r-s',...
        xgrid,solutions(5,isol).scalflux(igroup,:,1),'b:*',xgrid,solutions(6,isol).scalflux(igroup,:,1),'g-.+',...
        xgrid,solutions(7,isol).scalflux(igroup,:,1),'k--o','linewidth',2)
    title(sprintf('Scalar Flux at 50%% Rod Withdrawal, Group %i',igroup))
    legend('Volume Homog.','Subray w/o Recomb.','Subray w/ Recomb. - 0','Subray w/ Recomb. - 1',...
        'Subray w/ Recomb. - 2')
    ax = gca;
    ax.XAxis.TickValues = min(fssSolver(1).mesh.xsedges):input.pitch:max(fssSolver(1).mesh.xsedges);
    ax.XAxis.MinorTickValues = fssSolver(1).mesh.xsedges;
    ax.XTickLabelRotation = 45;
    ax.FontSize = 28;
    xlabel('Position (cm)');
    xlim([0,21.42]);
    ylabel('Scalar Flux');
    grid on;
end
figure(8)
plot(vfrac,keffs(1,:),'k',vfrac,keffs(4,:),'r-s',vfrac,keffs(5,:),'b:*',...
    vfrac,keffs(6,:),'g-.+',vfrac,keffs(7,:),'k--o','linewidth',2)
    legend('Volume Homog.','Subray w/o Recomb.','Subray w/ Recomb. - 0','Subray w/ Recomb. - 1',...
        'Subray w/ Recomb. - 2')
title('K-eff vs. Control Rod Withdrawal');
ax = gca;
ax.FontSize = 28;
xlabel('Control Rod Percent Withdrawn');
ylabel('k-eff');
```

BIBLIOGRAPHY

[1] M. Halsall. *CACTUS, a Characteristics Solution to the Neutron Transport Equations in Complicated Geometries*. *Technical report*, UKAEA Atomic Energy Establishment (1980).

[2] H. Finnemann, F. Bennewitz, and M. Wagner. "Interface current techniques for multidimensional reactor calculations." *Atomkernenergie*, **30(2)**: pp. 123–128 (1977).

[3] C. L. Hoxie. *Application of nodal equivalence theory to the neutronic analysis of PWRs*. Ph.D. thesis, Massachusetts Institute of Technology (1982).

[4] H. S. Khalil. *The application of nodal methods to PWR analysis*. Ph.D. thesis, Massachusetts Institute of Technology (1983).

[5] A. Y. Cheng. *Homogenization of BWR assemblies by response matrix methods*. Ph.D. thesis, Massachusetts Institute of Technology (1981).

[6] P. J. Finck. *Homogenization and dehomogenization schemes for BWR assemblies*. Ph.D. thesis, Massachusetts Institute of Technology (1982).

[7] F. Bennewitz, H. Finnemann, and M. Wagner. "Higher order corrections in nodal reactor calculations." *Transactions of the American Nuclear Society*, **22** (1975).

[8] H. Finnemann and W. Gundlach. "Space-time kinetics code iqsbox for pwr and bwr." *Atomkernenergie Kerntechnik*, **37** (1981).

[9] S. Langenbuch, W. Maurer, and W. Werner. "Coarse-mesh flux-expansion method for the analysis of space-time effects in large light water reactor cores." *Nuclear Science and Engineering*, **63(4)**: pp. 437–456 (1977).

[10] H.-S. Joo. *Resolution of the control rod cusping problem for nodal methods*. Ph.D. thesis, Massachusetts Institute of Technology (1984).

[11] J. C. Gehin. *A quasi-static polynomial nodal method for nuclear reactor analysis*. *Technical report*, Oak Ridge Inst. for Science and Education, TN (United States); Massachusetts Inst. of Tech., Cambridge, MA (United States) (1992).

[12] K. Smith *et al.* "Enhancements of the studvick core management system." In: *Proceedings of the Topical Meeting on Advances in Reactor Physics. Charleston SC USA* (1992).

[13] K. B. Lee *et al.* "Correction of the control rod cusping effect using one-dimensional fine mesh flux profiles." In: *Proceedings of the KNS Spring Meeting*, volume 2 (1998).

[14] T. Downar *et al.* "Theory manual for the parcs neutronics core simulator." *School of Nuclear Engineering, Purdue University, West Lafayette, Indiana*, **47907** (2004).

[15] Y. H. Kim and N. Z. Cho. "A flux-and adjoint flux-weighting method for the control rod cusping problem in nodal methods." In: *Proceedings of the KNS Spring Meething* (1990).

[16] Z. R. de Lima *et al.* "Correcting the cusping problem in three-dimensional transients through nem modification." *Nuclear Science and Engineering*, **170(1)**: pp. 66–74 (2012).

[17] A. Martinez, V. Pereira, and E. DA SILVA. "A system for the prediction and determination of the subcritical multiplication condition." *Kerntechnik*, **64(4)**: pp. 230–234 (1999).

[18] A. Dall'Osso. "Reducing rod cusping effect in nodal expansion method calculations." In: *Proceedings of the International Conference on the New Frontiers of Nuclear Technology: Reactor Physics, Safety and High-Performance Computing, PHYSOR, Seoul, Korea* (2002).

[19] A. Yamamoto. "A simple and efficient control rod cusping model for three-dimensional pin-by-pin core calculations." *Nuclear technology*, **145(1)**: pp. 11–17 (2004).

[20] L. Yu, D. Lu, and S. Zhang. "Further development of the ciama nodal method for kinetic applications." In: *Proceedings of the Reactor Physics Asia 2015 (RPHA15) Conference*. Jeju, South Korea (2016).

[21] D. Lu, L. Yu, and S. Zhang. "A three-dimensional nodal method with channel-wise intrinsic axial mesh adaptation." *Annals of Nuclear Energy*, **79**: pp. 152–157 (2015).

[22] B. Cho and N. Z. Cho. "A nonoverlapping local/global iterative method with 2-d/1-d fusion transport kernel and p-cmfd wrapper for transient reactor analysis." *Annals of Nuclear Energy*, **85**: pp. 937–957 (2015).

[23] Y. S. Jung and H. G. Joo. "Control rod decusping treatment based on local 3-d cmfd calculation for direct whole core transport solvers." In: *Proceedings of the International Congress on Advances in Nuclear Power Plants (ICAPP)* (2014).

[24] M. Ryu and H. G. Joo. "ntracer whole core transport solutions to c5g7-td benchmark." In: *Proceedings of the International Conference on Mathematics and Computation (M&C 2017)*. Jeju, Korea (2017).

[25] A. K. Prinja and E. W. Larsen. *Handbook of Nuclear Engineering*, chapter Principles of Neutron Transport. D. G. Cacuci (Ed.) (2010).

[26] A. Hebert. *Applied Reactor Physics*. Presses inter Polytechnique (2009).

[27] MPACT Team. *MPACT Theory Manual*. *Technical report*, Oak Ridge National Laboratory and the University of Michigan (2015).

[28] S. Stimpson *et al. Improved diffusion coefficents for SPN axial solvers in the MPACT 2D/1D method applied to the AP1000® PWR start-up core models*, volume 3, (pp. 1702–1716). American Nuclear Society (2015).

[29] B. C. Yee, E. W. Larsen, and B. Kochunas. "An ananalytic derivation of transport-corrected p_0 cross sections and diffusion coefficients." In: *Proceesings of the Physics of Reactors (PHYSOR) conference*. Sun Valley, ID (2016).

[30] B. R. Herman *et al. Improved diffusion coefficients generated from Monte Carlo codes*. *Technical report*, American Nuclear Society, 555 North Kensington Avenue, La Grange Park, IL 60526 (United States) (2013).

[31] M. Ryu *et al.* "Incorporation of anisotropic scattering in ntracer." In: *Transactions of the Korean Nuclear Society Autumn Meeting*. Pyeongchang, Korea (2014).

[32] A. Yamamoto, Y. Kitamura, and Y. Yamane. "Simplified treatments of anisotropic scattering in lwr core calculations." *Journal of Nuclear Science and Technology*, **45(3)**: pp. 217–229. URL `http://dx.doi.org/10.1080/18811248.2008.9711430` (2008).

[33] J. Askew. *A characteristics formulation of the neutron transport equation in complicated geometries*. *Technical report*, United Kingdom Atomic Energy Authority (1972).

[34] E. Masiello, R. Clemente, and S. Santandrea. "High-order method of characteristics for 2-d unstructured meshes." In: *Proc. Int. Conf. on Mathematics, Computational Methods & Reactor Physics (M&C 2009)*. Saratoga Springs, NY, USA (2009).

[35] K. Smith. "Nodal method storage reduction by nonlinear iteration." *Trans. Am. Nucl. Soc.*, **44**: p. 265 (1983).

[36] R. G. McClarren. "Theoretical aspects of the simplified pn equations." *Transport Theory and Statistical Physics*, **39(2-4)**: pp. 73–109 (2011).

[37] D. Knott and A. Yamamoto. *Handbook of Nuclear Engineering*, chapter Lattice Physics Computations. D. G. Cacuci (Ed.) (2010).

[38] B. M. Kochunas. *A Hybrid Parallel Algorithm for the 3-D Method of Characteristics Solution of the Boltzmann Transport Equation on High Performance Compute Clusters*. Ph.D. thesis, University of Michigan (2013).

[39] N. Z. Cho, G. S. Lee, and C. J. Park. "A fusion technique of 2-d/1-d methods for three-dimensional whole-core transport calculations." In: *Trans. Am. Nucl. Soc.*

[40] N. Z. Cho, G. S. Lee, and C. J. Park. “Fusion method of characteristics for 3d whole core transport calculation.” In: *Trans. Am. Nucl. Soc.*, volume 86, (pp. 322–324) (2002).

[41] J. Y. Cho *et al.* “Three-dimensional heterogeneous whole core transport calculations employing planar moc solutions.” In: *Trans. Am. Nucl. Soc.*, volume 87, (pp. 234–236) (2002).

[42] M. Hursin, B. Kochunas, and T. Downar. *DeCART Theory Manual. Technical report*, University of Michigan (2008).

[43] H. G. Joo *et al.* “Methods and performance of a three-dimensional whole-core transport code decart.” In: *PHYSOR 2004 – The Physics of Fuel Cycles and Advanced Nuclear Systems: Global Developments*. Chicago, Illinois (2004).

[44] B. W. Kelley and E. W. Larsen. “2d/1d approximations to the 3d neutron transport equation. i:theory.” In: *Trans. Am. Nucl. Soc.* Sun Valley, ID (2013).

[45] B. S. Collins *et al.* “Stability and accuracy of 3d neutron transport simulations using the 2d/1d method in mpact.” *Journal of Computational Physics*, **326**: pp. 612–628 (2016).

[46] B. W. Kelley. *An Investigation of 2D/1D Approximations to the 3D Boltzmann Transport Equation.* Ph.D. thesis, University of Michigan. URL `https://deepblue.lib.umich.edu/bitstream/handle/2027.42/113576/kelleybl_1.pdf?sequence=1&isAllowed=y` (2015).

[47] S. G. Stimpson. *An Azimuthal, Fourier Moment-Based Axial S_N Solver for the 2D/1D Scheme.* phdthesis, University of Michigan. URL `https://deepblue.lib.umich.edu/bitstream/handle/2027.42/111446/sgstim_1.pdf?sequence=1&isAllowed=y` (2015).

[48] J. Y. Cho *et al.* “Axial sp$_n$ and radial MOC coupled whole core transport calculation.” *J. Nucl. Sci. Technol.*, **44(9)**: pp. 1156–1171 (2007).

[49] M. Ryu *et al.* “Solution of the beavrs benchmark using the ntracer direct whole core calculation code.” *Journal of Nuclear Science and Technology*, **52(7-8)**: pp. 961–969. URL `http://dx.doi.org/10.1080/00223131.2015.1038664` (2015).

[50] S. Stimpson, B. S. Collins, and B. Kochunas. *MOC Efficiency Improvements Using a Jacobi Inscatter Approximation. techreport CASL-U-2016-1056-002*, Oak Ridge National Laboratory. URL `http://info.ornl.gov/sites/publications/files/Pub69335.pdf` (2016).

[51] D. E. Cullen. “Application of the probability table method to multigroup calculations of neutron transport.” *Nucl. Sci. Eng.*, **55**: p. 387 (1974).

[52] Y. Liu *et al.* “Resonance self-shielding methodology in mpact.” In: *Proceedings of the Conference on Mathematics and Computation (M&C 2013)*. Sun Valley, ID, USA. URL `http://www.casl.gov/docs/CASL-U-2015-0008-000.pdf` (2013).

[53] A. Yamamoto *et al.* “Derivation of optimum polar angle quadrature set for the method of characteristics based on approximation for the bickley function.” *J. Nucl. Sci. Technol.*, **44(2)** (2007).

[54] P. J. Davis and I. Polonsky. *A Handbook of Mathematical Functions*, chapter Numerical Interpolation, Differentiation and Integration. Dover (1972).

[55] K. S. Smith. *Spatial homogenization methods for light water reactor analysis*. Ph.D. thesis, Massachusetts Institute of Technology (1980).

[56] K. S. Smith. *An Analytic Nodal Method for Solving the Two-Group, Multidimensional, Static and Transient Neutron Diffusion Equation*. mathesis, Massachusetts Institute of Technology (1979).

[57] B. Bradie. *A Friendly Introduction to Numerical Analysis*. Pearson Education. ISBN 9788131709429. URL `https://books.google.com/books?id=oaaayS9azQUC` (2006).

[58] MATLAB. *version 9.0 (R2016a)*. The MathWorks Inc., Natick, Massachusetts (2016).

[59] E. E. Lewis *et al. Benchmark on Deterministic Transport Calculations without Spatial Homogenization. Technical report*, Nuclear Energy Agency Organisation for Economic Cooperation and Development (NEA-OECD) (2003).

[60] E. E. Lewis *et al. Benchmark on Deterministic Transport Calculations without Spatial Homogenization: MOX Fuel Assembly 3-D Extension Case. Technical report*, Nuclear Energy Agency Organisation for Economic Cooperation and Development (NEA-OECD) (2005).

[61] A. T. Godfrey. *VERA Core Physics Benchmark Progression Problem Specifications. Technical report*, Oak Ridge National Laboratory. URL `http://www.casl.gov/docs/CASL-U-2012-0131-004.pdf` (2014).

[62] W. B. Marshall, B. T. Rearden, and E. L. Jones. *Validation of SCALE 6.2 Criticality Calculations Using KENO V.A and KENO-VI*. URL `http://www.osti.gov/scitech/servlets/purl/1215589` (2015).