AN ABSTRACT OF THE THESIS OF

Andrew J. Hummel for the degree of Master of Science in Nuclear Engineering presented on December 17, 2010.

Title: Accelerating the Convergence of the k-Eigenvalue Problem using a Coarse Mesh Finite Differencing Scheme in Cylindrical Geometry

Abstract approved:

_____

Todd S. Palmer

Reactor modeling is largely limited by the computational time required to perform accurate full core calculations. There are many different methods and techniques employed in different reactor simulation codes, but properly modeling all of the physics that takes place in the system requires extensive computational effort. The Coarse Mesh Finite Differencing (CMFD) technique was proposed in 1983 by Kord Smith as a spatial acceleration scheme to combat this problem. It is a nonlinear iterative method that reduces the storage requirement of the problem by reducing the number of unknowns in the system. It is a diffusion based method that can be applied to diffusion and transport problems. The macroscopic cross sections and diffusion coefficients are homogenized accordingly to a coarser mesh. The reaction rates in the new coarse mesh cells are preserved along with the higher order surface currents. A current correction coefficient is introduced to maintain these currents. The finite differencing numerical approximation can then be applied to the 3-Dimensional

steady-state neutron diffusion equation resulting in a linear system of equations that is readily solvable.

This project has involved implementing the CMFD Acceleration into the reactor simulation code PARCS. PARCS was developed jointly between Purdue University and the University of Michigan to model Pebble Bed Modular Reactors. Although PARCS contains numerous numerical techniques, the focus of this research has been to accelerate the Fine Mesh Finite Differencing approximation in cylindrical geometry. Cylindrical coordinates prevents a higher order nodal method from being used as the primary scheme, but it allows for a more accurate representation of the core. The CMFDA was employed using a 2 group cross section library for fast and thermal neutrons.

Accelerating the Convergence of the k-Eigenvalue Problem using a Coarse Mesh Finite Differencing Scheme in Cylindrical Geometry

by

Andrew J. Hummel

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented December 17, 2010
Commencement June 2011

Master of Science thesis of Andrew J. Hummel presented on December 17, 2010.

APPROVED:

_____

Major Professor, representing Nuclear Engineering

_____

Head of the Department of Nuclear Engineering and Radiation Health Physics

_____

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

_____

Andrew J. Hummel, Author

ACKNOWLEDGEMENTS

I would first like to thank my family, especially my parents John and Marcia, for their continual support of my graduate studies. I would also like to thank my academic adviser, Dr. Todd Palmer, for working patiently with me throughout the course of this research. He has been instrumental in guiding me through this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDIX TABLES

Accelerating the Convergence of the k-Eigenvalue Problem using a Coarse Mesh Finite Differencing Scheme in Cylindrical Geometry

Chapter 1 Introduction

One of the pitfalls of modern day nuclear reactor analysis is the amount of computational time and storage needed to properly simulate all of the physics that takes place in the reactor system. The equations that model different phenomena such as neutron transport and heat transfer are well understood, and operational procedures have been developed to safely generate nuclear electricity. This is clearly evidenced by the abundance of commercial nuclear power and research reactors. Real time data can be acquired from a reactor's operational history and used to help generate improved computational models. These models serve as a basis for future reactor designs and for the simulation of existing reactor behavior under off-normal and accident conditions.

The evolution of reactor modeling began with simple experiments to determine the parameters of the four and six factor formulas. With the information gained, scaled reactors were built. Additional data such as energy-dependent cross section libraries could then be tabulated and the iterative process between modeling and experimentation continued. These early models were rudimentary compared to today's three-dimensional coupled physics codes because of advances both in physics and the capability of computers to process data. Over the years, empirically-derived parameters such as cross-sections have been determined to a great degree of accuracy through reactor operating experience and improved experimental capabilities. With increasing insight into the

physical behavior of nuclear systems, more accurate simulation tools have been developed.

Today's three-dimensional reactor physics models are based on one of three general mathematical formulations. When choosing which physical model to use, it is important to consider the type of reactor being modeled, the appropriate coordinate system, the desired accuracy of the solution, and the time available to obtain the solution. The most accurate treatment of neutron behavior in a reactor is the transport model. The transport equation cannot be analytically solved except in very simple, unrealistic conditions. Several different approaches have been developed to generate approximate solutions. One common method in Cartesian geometry is to transverse integrate the transport equation to obtain three coupled 1-D equations (Ougouag & Terry, 2002). These 1-D solutions are averaged over the other two directions leading to 3-D nodal solutions. However, if a different coordinate system is used, such as spherical (r, θ, ϕ) or cylindrical (r, θ, z), the transverse integration no longer becomes a viable method for separating the spatial directions and other more complex methods must be employed (Ougouag & Terry, 2002). The second model used for reactor analysis is based on the diffusion equation, an approximation to the transport equation. Finite differencing can be easily applied to the diffusion equation, leading to a set of linear equations that can be directly solved. This process has been used in a variety of coordinate systems and mesh types. A third and most intensive model for simulating neutron behavior involves Monte Carlo (MC) techniques. MC uses random statistical sampling from cumulative probability distribution functions (CPDF) to determine a

neutron's path length, type of interaction, and position, energy, and direction at birth. MC can be utilized in any coordinate system and does not require a spatial mesh. However, its results are subject to statistical error. Therefore, very large amounts of sampling may be needed to reduce the standard deviation of the results to appropriate levels.

All three models described above are employed in a wide range of reactor physics codes, and they are all burdened by the enormous number of unknowns present when performing a full core calculation. Based on the number of assemblies, the number of pins per assembly, the number of axial planes, and other factors depending on the model used, it is estimated that it would be 2030 at the earliest before a full core MC calculation could be completed on a single CPU in 1 hour (Smith, 2003). Deterministic transport calculations are not constrained by statistical error, but they have even more unknowns. This has lead to the development of acceleration schemes to improve computational performance. One of the most widely used acceleration techniques is Coarse Mesh Finite Differencing (CMFD). Although applicable to transport problems, this is a diffusion based method that accelerates the convergence of the eigenvalue calculation. It involves representing the interface current at nodal intersections with the average fluxes of the two adjacent nodes (Downer, Lee, Xu, Kozlowski, & Staudenmier, 2007). This method has been employed in the code PARCS (Purdue Advanced Reactor Core Simulator) which is used to analyze light water, heavy water, and gas-cooled reactor systems.

The inherent benefit of the CMFD method stems from the fact that it reduces the number of unknowns in the system. These unknowns include coupling coefficients and nodal fluxes that must be computationally stored throughout the iteration process (Smith, 1984). By spatially homogenizing cross section data to a coarser mesh, the size of the problem can be drastically reduced. However, reconstruction of the heterogeneous properties associated with a given zone or region that has been homogenized is nearly impossible, so preservation of the spatial integrals over the homogenized region is generally the goal (Smith, 1986). The quantities preserved are then the eigenvalue, node averaged reaction rates, and surface averaged currents. The node and surface averaging comes from the homogenization process described in section 2.3.

The eigenvalue, k, is the multiplication factor of the system and is defined by:

$$k = \frac{neutrons\ produced\ in\ current\ generation}{neutrons\ produced\ in\ previous\ generation}.$$

If k=1 then the system is critical and will sustain a chain reaction. Supercritical and subcritical systems are defined when k is greater or less than 1 respectively. The reaction rate density is the number of collisions of a particular type *x* per second per cubic centimeter. In terms of the macroscopic cross section and neutron flux, this is defined by the following integral:

$$\int_0^\infty \Sigma_x(E)\varphi(E)dE.$$

The macroscopic cross section is the probability of interaction of type *x* per cm at a given energy (Lewis, 2008). Example interactions include fission, absorption, or

scattering. The neutron current or leakage is the net number of neutrons passing through a surface per second per unit area.

With many of today's nuclear power plants reaching the end of their designed lifetime, there has been much research into the design of the next generation of nuclear plants. In the early 1990's a renewed interest in Very High Temperature Reactors (VHTR) surfaced due to a growing need for improved reactor standards (Huda & Obara, 2008). These gas-cooled, graphite-moderated reactors come in two general designs as shown in Figure 1.1. The prismatic core design uses graphite fuel blocks embedded with TRISO (tri-structural isotropic) fuel particles while the Pebble Bed Modular Reactor (PBMR) design involves circulating fuel pebbles containing the TRISO fuel particles in an annular core (INL, 2004). An inherent safety feature of the latter design stems from the fact that the spherical fuel particles act as tiny pressure vessels that contain the fission products (Huda & Obara, 2008). PARCS has been modified to model the behavior of this reactor to help determine its viability as a next generation alternative. Typical operating parameters for PBMRs are given in Table 1.1.

Table 1.1 – Typical PBMR specifications taken from Next Generation Nuclear Plant – Design Methods Development and Validation Research and Development Program Plan, INL

Power Output (MWt)	600
Plant Design Life (Years)	60
Thermal Efficiencies	45
Fuel Enrichment (%)	8
Fuel Burnup (MWd/ton)	90,000
Inlet Temperature (°C)	490
Outlet Temperature (°C)	1000
Maximum Fuel Temperature (°C)	1028

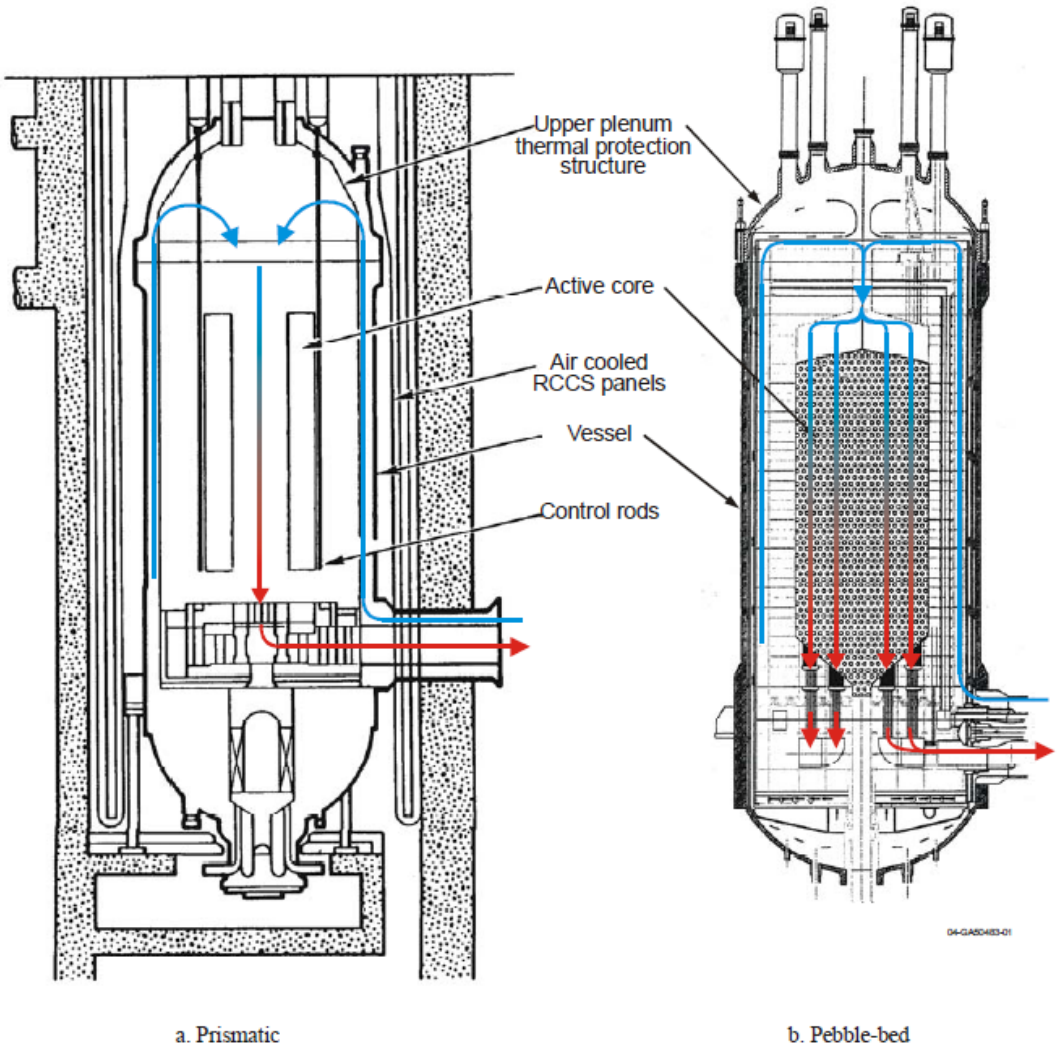

Figure 1.1 – Typical VHTR core configurations [INL, 2004]

Unlike typical boiling water reactors (BWR) and pressurized water reactors (PWR) that use water as a coolant and moderator, PBMRs use graphite as a moderator and helium as the gas coolant. The moderator is needed to slow neutrons down from fast to thermal energies. Most all neutrons resulting from fission are born in the fast energy range with greater than 1 mega electron-volt (MeV) of kinetic energy. Through collisions with the moderator medium, structural materials, and fuel, the neutrons will

lose energy until they become thermalized. A thermal neutron is in thermal equilibrium with its surroundings with an average energy dictated by the material temperature. Thermal neutrons are desired because uranium has a high thermal fission cross section. The coolant is used to transfer heat out of the core. Helium is used due to its chemical inertness, low absorption cross section, and ability to efficiently transfer heat (Huda & Obara, 2008).

To model the neutronic behavior of the PBMR, PARCS solves the diffusion approximation in cylindrical geometry. Fine Mesh Finite Differencing (FMFD) is used to obtain the heterogeneous solution, and the mesh size used leads to a slow convergence rate of the iterative solvers. The goal of this research was to develop and implement a CMFD solver within PARCS to improve the computational efficiency of the solution of PBMR FMFD diffusion problems. In Chapter 2 we introduce the diffusion equation in (r, θ, z) geometry and derive the CMFD equations. Several test problems were used to evaluate the accuracy and efficiency of the implementation, and Chapter 3 reviews the results of those problems and their implications. Chapter 4 contains a discussion of the results, conclusions, and offers suggestions for possible future work.

Chapter 2 Methods

2.1 Diffusion Equation:

The diffusion equation is shown with ϕ, v, J, Σ, υ, k, and S representing the scalar flux, neutron speed, neutron current, macroscopic cross section, neutrons per fission, and source respectively. Q is an external source. The independent variables for spatial location, energy, and time are $\underline{r}$, E, and t respectively.

$$\frac{\mathbf{1}}{\boldsymbol{v}(\boldsymbol{E})}\frac{\boldsymbol{\partial\phi}(\boldsymbol{r},\boldsymbol{E},\boldsymbol{t})}{\boldsymbol{\partial t}} + \underline{\boldsymbol{\nabla}}\cdot\underline{\boldsymbol{J}}\left(\underline{\boldsymbol{r}},\boldsymbol{E},\boldsymbol{t}\right) + \boldsymbol{\Sigma_a}\left(\underline{\boldsymbol{r}},\boldsymbol{E}\right)\boldsymbol{\phi}\left(\underline{\boldsymbol{r}},\boldsymbol{E},\boldsymbol{t}\right) = \boldsymbol{S}\left(\underline{\boldsymbol{r}},\boldsymbol{E},\boldsymbol{t}\right) \qquad (\mathbf{2.1})$$

$$\boldsymbol{S}\left(\underline{\boldsymbol{r}},\boldsymbol{E},\boldsymbol{t}\right) = \boldsymbol{\upsilon\Sigma_f}\left(\underline{\boldsymbol{r}},\boldsymbol{E}\right)\boldsymbol{\phi}\left(\underline{\boldsymbol{r}},\boldsymbol{E},\boldsymbol{t}\right) + \int_{\mathbf{0}}^{\infty}\boldsymbol{\Sigma_s}\left(\underline{\boldsymbol{r}},\boldsymbol{E'}\rightarrow\boldsymbol{E}\right)\boldsymbol{\phi}\left(\underline{\boldsymbol{r}},\boldsymbol{E'}\rightarrow\boldsymbol{E},\boldsymbol{t}\right)\boldsymbol{dE'} + \boldsymbol{Q}$$

Diffusion theory relies on the approximation that neutrons in a given system flow from areas of high concentration to areas of low concentration. In fact, neutrons travel in straight lines and only experience changes in kinetic energy and direction upon direct collision with nuclei. Fick's law given by Eq. 2.2a governs this diffusion process, and the quantity, D, is the diffusion coefficient which is introduced in the derivation of the diffusion equation and is defined in Eq. 2.2b.

$$\underline{\boldsymbol{J}}\left(\underline{\boldsymbol{r}},\boldsymbol{E}\right) = -\boldsymbol{D}\left(\underline{\boldsymbol{r}},\boldsymbol{E}\right)\cdot\underline{\boldsymbol{\nabla}}\boldsymbol{\phi}\left(\underline{\boldsymbol{r}},\boldsymbol{E}\right) \qquad (\mathbf{2.2a})$$

$$\boldsymbol{D}(\underline{\boldsymbol{r}}, \boldsymbol{E}) = \frac{\mathbf{1}}{\mathbf{3}\boldsymbol{\Sigma}_{\boldsymbol{tr}}(\underline{\boldsymbol{r}}, \boldsymbol{E})} \qquad (\mathbf{2.2}\boldsymbol{b})$$

In the case of isotropic scattering, $\Sigma_{tr}(\underline{r}, E) = \Sigma_t(\underline{r}, E)$. A similar expression exists for anisotropic scattering. Applying the steady-state assumption to Eq. 2.1 removes the temporal dependence from the other parameters and makes the time derivative term equal to zero. The eigenvalue, *k*, is introduced by replacing *v* with *v*/*k* and setting the external source to zero (Lewis & Miller, 1993). The resultant equation is the 3-D diffusion approximation in k-eigenvalue form.

$$\underline{\boldsymbol{\nabla}} \cdot \underline{\boldsymbol{J}}(\underline{\boldsymbol{r}}, \boldsymbol{E}) + \boldsymbol{\Sigma}_{\boldsymbol{a}}(\underline{\boldsymbol{r}}, \boldsymbol{E})\boldsymbol{\phi}(\underline{\boldsymbol{r}}, \boldsymbol{E})$$

$$= \frac{\mathbf{1}}{\boldsymbol{k}}\left[\boldsymbol{v}\boldsymbol{\Sigma}_{\boldsymbol{f}}(\underline{\boldsymbol{r}}, \boldsymbol{E})\boldsymbol{\phi}(\underline{\boldsymbol{r}}, \boldsymbol{E})\right] + \int_{\mathbf{0}}^{\infty} \boldsymbol{\Sigma}_{\boldsymbol{s}}\left(\underline{\boldsymbol{r}}, \boldsymbol{E}' \rightarrow \boldsymbol{E}\right)\boldsymbol{\phi}(\underline{\boldsymbol{r}}, \boldsymbol{E}' \rightarrow \boldsymbol{E})\boldsymbol{dE}' \qquad (\mathbf{2.3})$$

PARCS then employs FMFD to numerically solve this equation. Using an initial guess for the source and multiplication factor on the right hand side of equation 2.3, the only unknowns are the fluxes. A preconditioned Krylov subspace method is then used to solve the linear system (Downer et al, 2007).

2.2 Finite Differencing:

In this section we derive the discretized form of the diffusion equation in (r, θ, z) geometry in a single energy group. We divide the spatial domain into nodes with the subscripts *i, j,* and *k* indicating r, θ, z subdivisions. Figure 2.1 shows that neighboring

cells in the radial direction are coupled together through the interface currents at the cell boundaries.

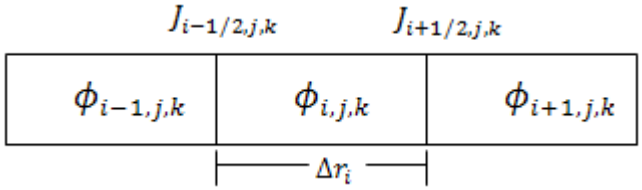

Figure 2.1: Neighboring cells shown coupled by their currents.

By integrating equation 2.3 over the volume of cell *i, j, k* the new equation can be written as

$$\boldsymbol{J}^{\boldsymbol{n+1/2}}_{\boldsymbol{g,(i+1/2,j,k)}} - \boldsymbol{J}^{\boldsymbol{n+1/2}}_{\boldsymbol{g,(i-1/2,j,k)}} + \boldsymbol{J}^{\boldsymbol{n+1/2}}_{\boldsymbol{g,(i,j+1/2,k)}} - \boldsymbol{J}^{\boldsymbol{n+1/2}}_{\boldsymbol{g,(i,j-1/2,k)}} + \boldsymbol{J}^{\boldsymbol{n+1/2}}_{\boldsymbol{g,(i,j,k+1/2)}} - \boldsymbol{J}^{\boldsymbol{n+1/2}}_{\boldsymbol{g,(i,j,k-1/2)}} + \boldsymbol{\Sigma}_{\boldsymbol{t(,i,j,k)}}\boldsymbol{\Delta V}_{\boldsymbol{(i,j,k)}}\boldsymbol{\phi}^{\boldsymbol{n+1/2}}_{\boldsymbol{g,(i,j,k)}}$$

$$= \boldsymbol{\chi_g}\frac{\mathbf{1}}{\boldsymbol{k}}\sum_{\mathbf{g'=1}}^{\mathbf{G}}\left[\boldsymbol{\upsilon\Sigma}_{\boldsymbol{f,g',(i,j,k)}}\boldsymbol{\Delta V}_{\boldsymbol{(i,j,k)}}\boldsymbol{\phi}^{\boldsymbol{n}}_{\boldsymbol{g',(i,j,k)}}\right] + \sum_{\boldsymbol{g'=1}}^{\boldsymbol{G}}\boldsymbol{\Sigma}_{\boldsymbol{s,g'g,(i,j,k)}}\,\boldsymbol{\Delta V}_{\boldsymbol{(i,j,k)}}\boldsymbol{\phi}^{\boldsymbol{n}}_{\boldsymbol{g',(i,j,k)}} \qquad (\mathbf{2.4})$$

with

$$\boldsymbol{\Delta V}_{\boldsymbol{(i,j,k)}} = \bar{\boldsymbol{r}}_{\boldsymbol{i}}\boldsymbol{\Delta r}_{\boldsymbol{i}}\boldsymbol{\Delta\theta}_{\boldsymbol{j}}\boldsymbol{\Delta z}_{\boldsymbol{k}} \quad \text{and} \quad \bar{\boldsymbol{r}}_{\boldsymbol{i}} = \frac{\boldsymbol{r}_{\boldsymbol{i+1/2}} + \boldsymbol{r}_{\boldsymbol{i-1/2}}}{\mathbf{2}}.$$

Here n refers to the iteration index, g the energy group, and the scattering term includes only down scatter into the energy group of interest. Finite differencing is applied to Eq. (2.4) to obtain two approximations for each of the surface currents in the r, θ, and z directions. $\phi_{k\pm1/2}$ are the fluxes at the respective surfaces.

$$\boldsymbol{a)}\ \boldsymbol{J}_{\boldsymbol{k+1/2}} = -\boldsymbol{D}_{\boldsymbol{k}}\frac{\boldsymbol{\phi}_{\boldsymbol{k+1/2}} - \boldsymbol{\phi}_{\boldsymbol{k}}}{\boldsymbol{\Delta z}_{\boldsymbol{k}}/\mathbf{2}} \qquad \boldsymbol{b)}\ \boldsymbol{J}_{\boldsymbol{k+1/2}} = -\boldsymbol{D}_{\boldsymbol{k+1}}\frac{\boldsymbol{\phi}_{\boldsymbol{k+1}} - \boldsymbol{\phi}_{\boldsymbol{k+1/2}}}{\boldsymbol{\Delta z}_{\boldsymbol{k+1}}/\mathbf{2}} \qquad (\mathbf{2.5})$$

$$\boldsymbol{a)}\ J_{k-1/2} = -D_{k-1}\frac{\phi_{k-1/2} - \phi_{k-1}}{\Delta z_{k-1}/2} \qquad \boldsymbol{b)}\ J_{k-1/2} = -D_k\frac{\phi_k - \phi_{k-1/2}}{\Delta z_k/2} \qquad (2.6)$$

Setting 2.5a equal to 2.5b and 2.6a to 2.6b will yield two expressions for the interface fluxes as shown below.

$$\phi_{k+1/2} = \frac{\frac{D_k\phi_k}{\Delta z_k/2} + \frac{D_{k+1}\phi_{k+1}}{\Delta z_{k+1}/2}}{\frac{D_k}{\Delta z_k/2} + \frac{D_{k+1}}{\Delta z_{k+1}/2}} \qquad (2.7)$$

$$\phi_{k-1/2} = \frac{\frac{D_k\phi_k}{\Delta z_k/2} + \frac{D_{k-1}\phi_{k-1}}{\Delta z_{k-1}/2}}{\frac{D_k}{\Delta z_k/2} + \frac{D_{k-1}}{\Delta z_{k-1}/2}} \qquad (2.8)$$

Equation (2.7) can be substituted back into Eq. (2.5a) yielding an expression for the current as:

$$J_{k+1/2} = \frac{-D_k}{\frac{\Delta z_k}{2}} * \left(\frac{\frac{D_k\phi_k}{\frac{\Delta z_k}{2}} + \frac{D_{k+1}\phi_{k+1}}{\frac{\Delta z_{k+1}}{2}}}{\frac{D_k}{\frac{\Delta z_k}{2}} + \frac{D_{k+1}}{\frac{\Delta z_{k+1}}{2}}} - \phi_k \right) \qquad (2.9)$$

A similar substitution can be made with Equations (2.8) and (2.6b) to generate the current at $J_{k-1/2}$. Following some minor algebra, the fine mesh currents are given below. The quantity $\widetilde{D}$ represents the base nodal coupling coefficient.

$$J_{k+1/2} = \frac{-D_k}{\frac{\Delta z_k}{2}} * \left(\frac{\frac{D_{k+1}(\phi_{k+1} - \phi_k)}{\frac{\Delta z_{k+1}}{2}}}{\frac{D_k}{\frac{\Delta z_k}{2}} + \frac{D_{k+1}}{\frac{\Delta z_{k+1}}{2}}} \right) = -\frac{\widetilde{D}_{k+1}}{\Delta z_k}(\phi_{k+1} - \phi_k) \qquad (2.10)$$

$$J_{k-1/2} = -\frac{\widetilde{D}_{k-1}}{\Delta z_k}(\phi_k - \phi_{k-1}) \qquad (2.11)$$

$$J_{i+1/2} = -\frac{\widetilde{D}_{i+1}}{\Delta r_i}(\phi_{i+1} - \phi_i) \tag{2.12}$$

$$J_{i-1/2} = -\frac{\widetilde{D}_{i-1}}{\Delta r_i}(\phi_i - \phi_{i-1}) \tag{2.13}$$

$$J_{j+1/2} = -\frac{\widetilde{D}_{j+1}}{\Delta \theta_j}(\phi_{j+1} - \phi_j) \tag{2.14}$$

$$J_{j-1/2} = -\frac{\widetilde{D}_{j-1}}{\Delta \theta_j}(\phi_j - \phi_{j-1}) \tag{2.15}$$

These current definitions can be used in Eq. (2.4) to form the following FMFD equation.

$$\phi^{n+1/2}_{g,(i+1,j,k)}(-\widetilde{D}_{g,(i+1,j,k)})\Delta\theta_j\Delta z_k + \phi^{n+1/2}_{g,(i-1,j,k)}(-\widetilde{D}_{g,(i-1,j,k)})\Delta\theta_j\Delta z_k +$$

$$\phi^{n+1/2}_{g,(i,j+1,k)}(-\widetilde{D}_{g,(i,j+1,k)})\Delta r_i\Delta z_k + \phi^{n+1/2}_{g,(i,j-1,k)}(-\widetilde{D}_{g,(i,j-1,k)})\Delta r_i\Delta z_k +$$

$$\phi^{n+1/2}_{g,(i,j,k+1)}(-\widetilde{D}_{g,(i,j,k+1)})\Delta r_i\Delta\theta_j + \phi^{n+1/2}_{g,(i,j,k-1)}(-\widetilde{D}_{g,(i,j,k-1)})\Delta r_i\Delta\theta_j +$$

$$\phi^{n+1/2}_{g,(i,j,k)}\left[\left(\widetilde{D}_{g,(i+1,j,k)} + \widetilde{D}_{g,(i-1,j,k)}\right)\Delta\theta_j\Delta z_k + \left(\widetilde{D}_{g,(i,j+1,k)} + \widetilde{D}_{g,(i,j-1,k)}\right)\Delta r_i\Delta z_k +\right.$$

$$\left.\left(\widetilde{D}_{g,(i,j,k+1)} + \widetilde{D}_{g,(i,jk-1)}\right)\Delta r_i\Delta\theta_j + \Sigma_{t,(i,j,k)}\Delta V_{(i,j,k)}\right] =$$

$$\chi_g\frac{1}{k}\sum_{g=1}^{G}\upsilon\Sigma_{f,g',(i,j,k)}\Delta V_{(i,j,k)}\phi^{n}_{g',(i,j,k)} + \sum_{g'=1}^{G}\Sigma_{s,g'g,(i,j,k)}\Delta V_{(i,j,k)}\phi^{n}_{g',(i,j,k)} \tag{2.16}$$

PARCS then employs power iteration to calculate the eigenvalue. Using an initial guess for the source and multiplication factor on the right hand side of Eq. (2.3), the only unknowns are the fluxes. A preconditioned Krylov subspace method is then used to solve the linear system (Downer et al, 2007).

2.3 Homogenization:

Once a fine mesh (FM) flux is generated through power iteration, it can be used to homogenize this cross section data on a coarse mesh (CM). The homogenization process must preserve the net reaction rate over all fine mesh nodes that lie in each coarse mesh node. This is achieved by imposing the following two constraints:

$$\int_{V,cm} \boldsymbol{\Sigma}_{x,g}^{cm}(\boldsymbol{r})\boldsymbol{\phi}_g^{cm}(\boldsymbol{r})\boldsymbol{dr} = \int_{V,fm} \boldsymbol{\Sigma}_{x,g}^{fm}(\boldsymbol{r})\boldsymbol{\phi}_g^{fm}(\boldsymbol{r})\boldsymbol{dr} \qquad (\mathbf{2.17})$$

$$\int_{S,cm} \boldsymbol{\nabla}\cdot\boldsymbol{J}_g^{cm}(\boldsymbol{r})\cdot\boldsymbol{dS} = \int_{S,fm} \boldsymbol{\nabla}\cdot\boldsymbol{J}_g^{fm}(\boldsymbol{r})\cdot\boldsymbol{dS} \qquad (\mathbf{2.18})$$

Here *x* refers to a nuclear reaction (t, s, f) for a specific cross section, *cm* the coarse mesh node index, *fm* the FM node index, *g* the energy group, *V* the volume, and *S* the surface. The homogenized CM data are then defined by:

$$\boldsymbol{\Sigma}_{x,g}^{cm} \equiv \frac{\int_{V,fm} \boldsymbol{\Sigma}_{x,g}^{fm}(\boldsymbol{r})\,\boldsymbol{\phi}_g^{fm}(\boldsymbol{r})\boldsymbol{dr}}{\int_{V,cm} \boldsymbol{\phi}_g^{cm}(\boldsymbol{r})\boldsymbol{dr}} \qquad (\mathbf{2.19})$$

$$\boldsymbol{D}_g^{cm} \equiv \frac{\int_{S,fm} \boldsymbol{J}_g^{fm}(\boldsymbol{r})\cdot\boldsymbol{dS}}{-\int_{S,cm} \boldsymbol{\nabla}\boldsymbol{\phi}_g^{cm}(\boldsymbol{r})\cdot\boldsymbol{dS}} \qquad (\mathbf{2.20})$$

It is important to note that every coarse mesh node has six surfaces and only one diffusion coefficient. In order to preserve the fine mesh current on the surface of each coarse mesh cell, another parameter, $\widehat{D}$, must be introduced. Equation 2.17 defines $\widehat{D}$ in terms of known node average fluxes. For example, in the arbitrary z-direction, we find

$$\boldsymbol{J}_g^{cm+1/2} = -\frac{\widetilde{\boldsymbol{D}}_g^{cm}}{\boldsymbol{\Delta z}^{cm}}\left(\boldsymbol{\phi}_g^{cm+1} - \boldsymbol{\phi}_g^{cm}\right) - \frac{\widehat{\boldsymbol{D}}_g^{cm}}{\boldsymbol{\Delta z}^{cm}}\left(\boldsymbol{\phi}_g^{cm+1} + \boldsymbol{\phi}_g^{cm}\right) \qquad (\mathbf{2.21})$$

$$with \qquad \boldsymbol{\phi}_g^{cm} = \frac{\int_{V,fm} \boldsymbol{\phi}_g^{fm}(\boldsymbol{r})\boldsymbol{dr}}{\int_{V,cm} \boldsymbol{dr}}$$

This new variable called the Corrective Nodal Coupling Coefficient is derived from the solution of a higher order problem, in this case the FMFD method in PARCS. It simply forces the current on the surface of the CM to equal that of the FM by adding a degree of freedom in the CMFD.

2.4 CMFD Formulation:

Once the FM flux solution is obtained, all of the CM parameters can be calculated. Equation (2.21) can then be used to replace the current in terms of unknown CM fluxes, $\widetilde{D}$, $\widehat{D}$, and mesh spacing. Using Eq. (2.21) in Eq. (2.4) yields a linear system of equations for the CM flux. Each node *m* is coupled to six other nodes designated east, west, north, south, top, and bottom (e, w, n, s, t, b).

$$\boldsymbol{\phi}_{g,e}^{l+\frac{1}{2}}(-\widetilde{\boldsymbol{D}}_{g,e} - \widehat{\boldsymbol{D}}_{g,e})\boldsymbol{\Delta\theta\Delta z} + \boldsymbol{\phi}_{g,w}^{l+\frac{1}{2}}(-\widetilde{\boldsymbol{D}}_{g,w} + \widehat{\boldsymbol{D}}_{g,w})\boldsymbol{\Delta\theta\Delta z} +$$

$$\boldsymbol{\phi}_{g,n}^{l+\frac{1}{2}}(-\widetilde{\boldsymbol{D}}_{g,n} + \widehat{\boldsymbol{D}}_{g,n})\boldsymbol{\Delta r\Delta z} + \boldsymbol{\phi}_{g,s}^{l+\frac{1}{2}}(-\widetilde{\boldsymbol{D}}_{g,s} - \widehat{\boldsymbol{D}}_{g,s})\boldsymbol{\Delta r\Delta z} +$$

$$\boldsymbol{\phi}_{g,t}^{l+\frac{1}{2}}(-\widetilde{\boldsymbol{D}}_{g,t} - \widehat{\boldsymbol{D}}_{g,t})\boldsymbol{\Delta r\Delta\theta} + \boldsymbol{\phi}_{g,b}^{l+\frac{1}{2}}(-\widetilde{\boldsymbol{D}}_{g,b} + \widehat{\boldsymbol{D}}_{g,b})\boldsymbol{\Delta r\Delta\theta} +$$

$$\boldsymbol{\phi}_{g,m}^{l+\frac{1}{2}}\left[\left(\widetilde{\boldsymbol{D}}_{g,e} - \widehat{\boldsymbol{D}}_{g,e}\right)\boldsymbol{\Delta\theta\Delta z} + \left(\widetilde{\boldsymbol{D}}_{g,w} + \widehat{\boldsymbol{D}}_{g,w}\right)\boldsymbol{\Delta\theta\Delta z} + \left(\widetilde{\boldsymbol{D}}_{g,n} + \widehat{\boldsymbol{D}}_{g,n}\right)\boldsymbol{\Delta r\Delta z} +\right.$$

$$\left.\left(\widetilde{\boldsymbol{D}}_{g,s} - \widehat{\boldsymbol{D}}_{g,s}\right)\boldsymbol{\Delta r\Delta z} + \left(\widetilde{\boldsymbol{D}}_{g,t} - \widehat{\boldsymbol{D}}_{g,t}\right)\boldsymbol{\Delta r\Delta\theta} + \left(\widetilde{\boldsymbol{D}}_{g,b} + \widehat{\boldsymbol{D}}_{g,b}\right)\boldsymbol{\Delta r\Delta\theta} + \boldsymbol{\Sigma}_{a,g,m}\boldsymbol{\Delta V}_m\right]$$

$$= \boldsymbol{\chi}_g \frac{\mathbf{1}}{\boldsymbol{k}} \boldsymbol{\upsilon\Sigma}_{f,g',m}\boldsymbol{\Delta V}_m\boldsymbol{\phi}_{g',m}^{l} + \sum_{g'=1}^{G} \boldsymbol{\Sigma}_{s,g'g,m}\,\boldsymbol{\Delta V}_m\boldsymbol{\phi}_{g',m}^{l}\Big] \qquad (\mathbf{2.22})$$

This leads to a hepta-diagonal matrix and is solved using exactly the same procedure as on the FM. The eigenvalue and fission source are checked for convergence after every iteration. Each must meet the convergence criteria for the iteration procedure to stop. Each iteration solves for the flux and fission source of the linear equation for all spatial nodes in a given energy group. The new coarse mesh flux is then used to update the fine mesh flux with the following relation:

$$\boldsymbol{\phi}_{fm}^{l+1} = \frac{\boldsymbol{\phi}_{fm}^{l}}{\boldsymbol{\phi}_{cm}^{l}} \boldsymbol{\phi}_{cm}^{l+1/2} \qquad (\mathbf{2.23})$$

2.5 Implementation:

The CMFD was integrated into a special version of PARCS containing only the cylindrical geometry solver, PARCS-CYL. Several new routines were created, and several existing routines were modified. Although PARCS utilizes a Graphical User Interface (GUI), it was strictly the source code and input deck that were modified.

PARCS reads in cross-section data, which includes all material properties, the number of FM nodes, and the FM node dimensions from the input deck. It does this by first incorporating a problem-dependent number of material regions. The number of FM nodes that compromise the region are then specified. Finally, the dimensions of each material region are specified.

The first modification required specifying in the code the number of CM nodes to be used and the number of FM nodes in each CM node. A good reasonable starting point for these values would be to use the material regions as the CM nodes. This definition or choice would lead to homogenized nodes with uniform fine-mesh material

properties. This choice is arbitrary, and the material mesh does not necessarily lead to the most efficient acceleration. Below is an example graphical representation of the relationship between the material regions (MR), FM and CM nodes.

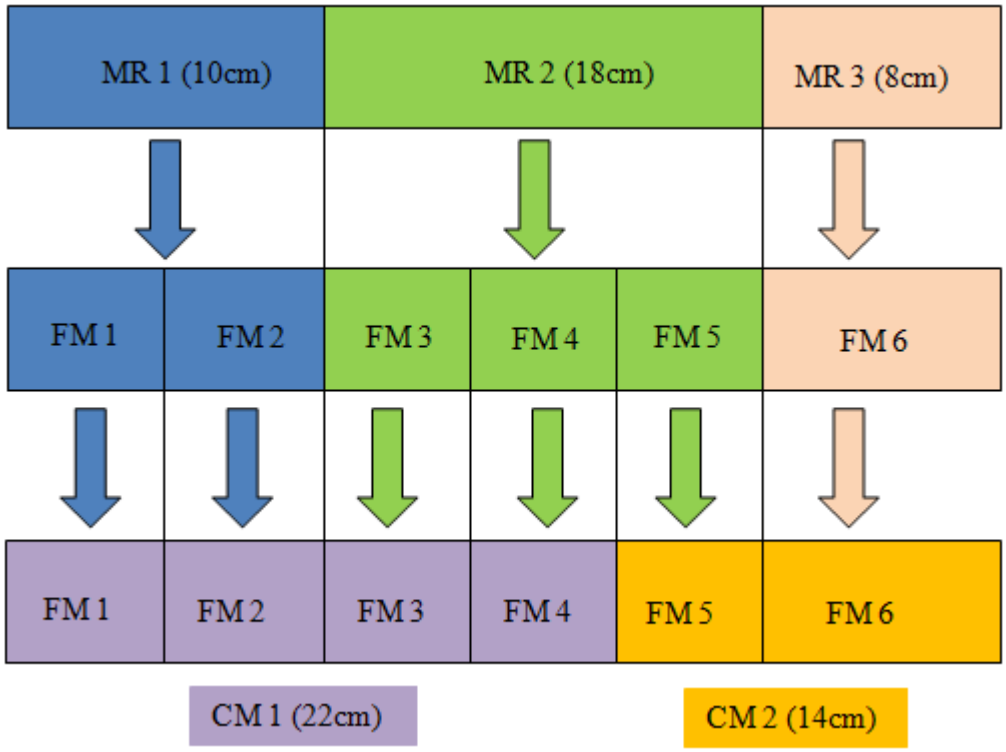

Figure 2.2 – Schematic of how information is passed from the MM to the FM to the CM.

This shows three different material regions of specified dimensions being divided into six FM nodes that are then arbitrarily grouped into two CM nodes. The FM dimensions determine the dimensions of the CM nodes. Every CM node boundary must coincide with a FM node boundary, and this leads to CM nodes of varying dimension. We will consider problems where the CM boundary overlaps the MR nodes, but this does not guarantee the CM nodes will be of equal dimension.

Node averaged fluxes and coupling coefficients are used in the CMFD, and they are obtained from the FMFD calculation. The first complete power iteration provides an

estimate of fluxes and the eigenvalue that will be used for the first iteration of the CMFD. Each iteration on both mesh sizes yields more accurate solutions, but the information gained per iteration compared to the computation time required favors doing as much of the work as possible with the CMFD.

After each power iteration on the FM problem, the information is used to perform a converged CM solution. Once converged, the CM flux is propagated back to the FM along with the eigenvalue, and the process is repeated until the fission source and eigenvalue are converged to the specified tolerance shown in Table 2.1.

Table 2.1 – Convergence Criteria

Local Fission Source	1.0e-5
Global Fission Source	1.0e-6
Eigenvalue	1.0e-6

2.6 PARCS Control Logic:

A diagram showing the entire control flow logic of PARCS is shown in Figure 2.3. The initialization process consists of reading information from the input deck, establishing cross section data, and initializing computational parameters to their appropriate settings. The FMFD coupling coefficients are then determined, and the FMFD linear system is setup. The preconditioners needed for the Bi-Conjugate Gradient Stabilized (BiCGSTAB) algorithm are then constructed (Downer et al, 2007) . The BiCGSTAB method is used because it has proven successful for nonlinear problems and because the conjugate gradient (CG) method only works for symmetric positive definite linear systems (SPD) (Downer et al, 2007). While the linear system for the FMFD problem is SPD, the CMFD equations are not. Specific criteria must be met on the

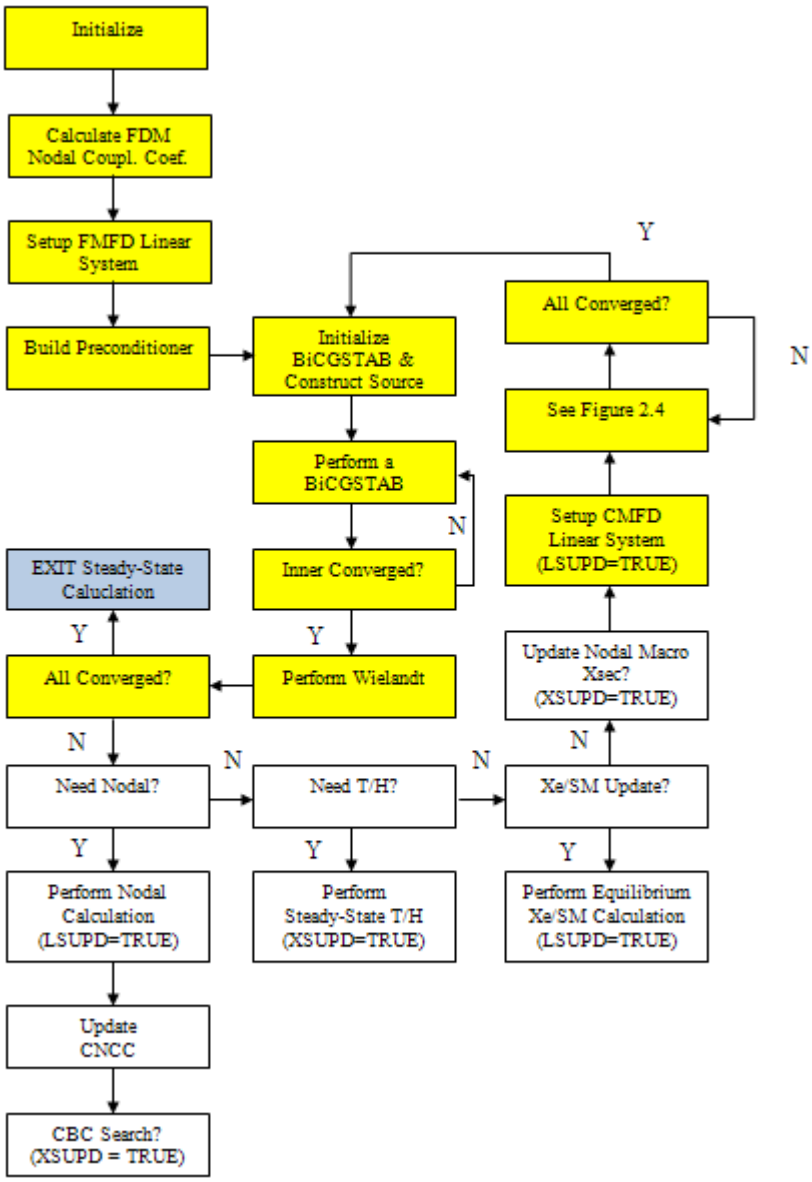

Figure 2.3 – Schematic of the control logic used in PARCS for the eigenvalue calculation

residual vector before the FM flux is updated. The Wielandt Shift option, if turned on, will calculate the new eigenvalue. This option serves to speed up the fission source iteration. Tests for the convergence criteria in Table 2.1 then determine whether the eigenvalue calculation is complete.

Only the yellow boxes in Figure 2.3 are relevant to this project. The Need Nodal box refers to updating the flux with a higher order nodal method. There is no higher order nodal method in PARCS for cylindrical geometry. The Need T/H, Xe/Sm, and XSUPD boxes can only be invoked during a transient. These steps are important in normalizing the power distribution, updating the Xenon/Samarium number densities, and updating the cross sections (Downer et al, 2007). The flow logic for setting up the CMFD linear system is shown in Figure 2.4.

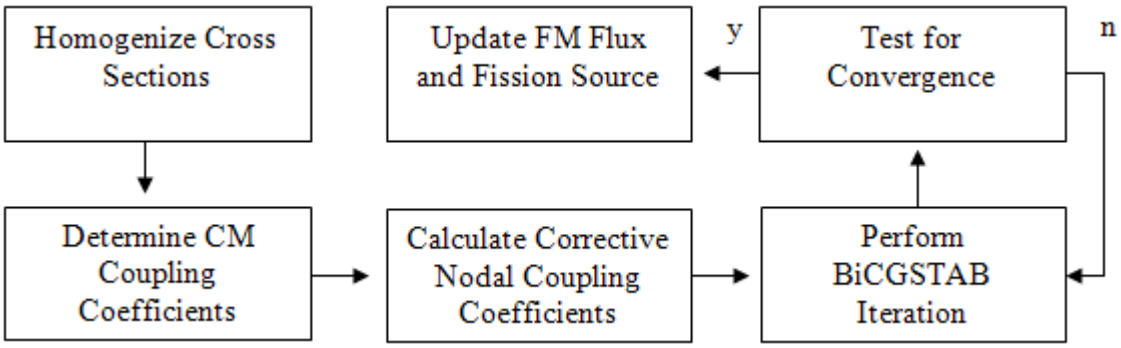

Figure 2.4 – CMFD control logic

Upon convergence of the CM problem, another FMFD iteration takes place. This process continues until the FM problem passes all convergence tests, at which time the eigenvalue calculation is complete.

Chapter 3 Results

In this Chapter we present the results of several test problems designed to evaluate the performance of the CMFD implementation in cylindrical geometry.

3.1 Homogeneous Infinite Medium Problem:

Our first test problem is a homogeneous medium with reflecting conditions on all external boundaries. These boundary conditions model an infinite medium (no leakage of neutrons from the system). The material properties are from material region 1 in the OECD-PBMR400 cross section library and are given in Table 3.1. For the complete library refer to Frederik Reitsma's paper in the references.

Table 3.1 – Homogeneous Material Properties

Parameter	Fast Group	Thermal Group
D	2.104250e+00	1.56790e+00
Σ_a	6.267630e-04	2.969320e-03
$\nu\Sigma_f$	1.808830e-04	4.529160e-03
Σ_f	7.382980e-05	1.848637e-03
$\Sigma_s(1 \rightarrow 2 \; or \; 2 \rightarrow 1)$	1.287270e-03	0

The system is governed by the two diffusion equations representing the fast and thermal energy groups as shown in Eqs. (3.1) and (3.2).

$$-\boldsymbol{\nabla \cdot D_1 \nabla \phi_1 + \Sigma_{r,1} \phi_1 = \frac{1}{k}\left[\nu\Sigma_{f,1}\phi_1 + \nu\Sigma_{f,2}\phi_2\right]} \qquad (\mathbf{3.1})$$

$$-\boldsymbol{\nabla \cdot D_2 \nabla \phi_2 + \Sigma_{a,2} \phi_2 = \Sigma_{s,1\rightarrow 2}\phi_1} \qquad (\mathbf{3.2})$$

Eliminating the leakage terms, the infinite medium multiplication factor for the system can then be calculated as follows:

$$\boldsymbol{k_\infty = \left(\frac{\nu\Sigma_{f,2}}{\Sigma_{a,2}}\right)\left(\frac{\Sigma_{s,1\rightarrow 2}}{\Sigma_{s,1\rightarrow 2} + \Sigma_{a,1}}\right)\left(\frac{\nu\Sigma_{f,1}\phi_1 + \nu\Sigma_{f,2}\phi_2}{\nu\Sigma_{f,2}\phi_2}\right)} \qquad (\mathbf{3.3})$$

Using the data in Table 3.1 yields a k_∞ of 1.1203466. Table 3.2 shows that for all the geometries tested, the solution was achieved in three fine mesh iterations. The CMFD

Table 3.2 – Homogeneous Problem Results

Test Problem	Cells		Fine Mesh Iterations	k	Net Acceleration Factor
	R	Z			
Geometry 1	20	29	3	1.1203466	0.726
Geometry 2	45	29	3	1.1203466	0.733
Geometry 3	20	45	3	1.1203466	0.736
Geometry 4	45	45	3	1.1203466	0.699
Geometry 5	10	29	3	1.1203466	0.740
Geometry 6	20	15	3	1.1203466	0.741
Geometry 7	10	15	3	1.1203466	0.749
Geometry 8	105	70	3	1.1203466	0.636
Fine Mesh Answer	213	145	3	1.1203466	

actually slows down the solution to this problem, but this is expected. The initial eigenvalue and fission source guess make it impossible for the FM solution to converge in the first two iterations. This is because the first iteration yields the correct solution to the problem, but this solution is much different than the initial guess, so there is no convergence. The second iteration on the FM also yields the correct solution, and all convergence tests are now passed. However, on the FM the convergence tests must be passed in two consecutive iterations to complete the eigenvalue problem. Three iterations are therefore the minimum needed, and that is how many it takes for this problem. The CMFD is unnecessary and causes the calculation to be slower than the FMFD solver on its own for all versions of this test problem. Appendix A has the dimensions of each CM geometry including the FM cell distribution. The net acceleration factor is the total improvement in computational speed including all FM

and CM solves, the homogenization process, calculation of the CNCCs, and convergence testing. It is calculated by dividing the total time the FMFD solver takes to converge the eigenvalue by the time required when the CMFD is used.

3.2 Homogeneous Medium Problem with Top Vacuum Boundary Condition:

This problem is exactly like the one described in section 3.1 except that there is a vacuum boundary condition imposed on the top axial plane, allowing neutron leakage through this boundary. The analytic solution can be obtained by solving the partial differential equation.

The geometric buckling is the solution to a Helmholtz equation, which characterizes the spatial dependence of the neutron flux (Lewis). This equation is shown below along with the formula for the geometric buckling for a cylindrically shaped reactor (Lewis, 2008) .

$$\boldsymbol{\nabla^2\phi + B^2\phi = 0} \qquad (\mathbf{3.5})$$

$$\boldsymbol{B^2 = (\frac{2.405}{\tilde{R}})^2 + (\frac{\pi}{\tilde{H}})^2} \qquad (\mathbf{3.6})$$

Here R is the radius and H is the height of the core. For the problem at hand, the radial direction extends infinitely because of the reflecting boundaries making the radial term become zero. The geometric buckling only depends on $\tilde{H}$, the extrapolated height which is $H + 2\lambda$, where λ is the extrapolation distance. $\tilde{R}$ is the extrapolated radius, and it is equal to $R + 2\lambda$. This extrapolation distance is the distance beyond the boundary of the core at which the flux theoretically goes to zero.

Equations (3.7) and (3.8) define the fast and thermal diffusion lengths respectively. The diffusion length is the distance between where a neutron is born and where it is absorbed (Lewis).

$$L_1 = \sqrt{D_1/\Sigma_{r,1}} \tag{3.7}$$

$$L_2 = \sqrt{D_2/\Sigma_{a,2}} \tag{3.8}$$

Using these expressions in equations 3.1 and 3.2 respectively yields the following equations:

$$L_1^2\Sigma_{r,1}\mathrm{B}^2\phi_1 + \Sigma_{r,1}\phi_1 = \frac{1}{k}\left[\nu\Sigma_{f,1}\phi_1 + \nu\Sigma_{f,2}\phi_2\right] \tag{3.9}$$

$$L_2^2\Sigma_{a,2}\mathrm{B}^2\phi_2 + \Sigma_{a,2}\phi_2 = \Sigma_{s,1\to2}\phi_1 \tag{3.10}$$

Both sides of equations 3.9 and 3.10 can be divided by the thermal flux, and equation 3.9 will then yield an expression for the fast to thermal flux ratio which can be substituted into equation 3.10. This results in the expression for k shown below.

$$k = \frac{\nu\Sigma_{f,1}}{\left[-L_1^2\Sigma_{r,1}B^2 + \Sigma_{r,1}\right]} + \frac{\nu\Sigma_{f,2}\Sigma_{s,1\to2}}{\left[-L_1^2\Sigma_{r,1}B^2 + \Sigma_{r,1}\right]\left[-L_2^2\Sigma_{a,2}B^2 + \Sigma_{a,2}\right]} \tag{3.11}$$

Using the data in Table 3.1, k = 1.1182219. This analytic solution is also achieved by PARCS to within 1.0e-05, and the results are shown in Table 3.4.

Table 3.3 – Results for the homogenous problem with a vacuum boundary on the top axial plane

Test Problem	Cells		Fine Mesh Iterations	k	Average Iteration Time (sec)	Iteration Improvement	Net Acceleration Factor
	R	Z					
Geometry 1	20	29	10	1.1182686	0.086	64.1	24.7
Geometry 2	45	29	10	1.1182686	0.133	41.6	16.3
Geometry 3	20	45	24	1.1182686	0.095	57.9	7.81
Geometry 4	45	45	24	1.1182686	0.215	25.7	3.53
Geometry 5	10	29	10	1.1182686	0.061	90.2	34.6
Geometry 6	20	15	45	1.1182686	0.049	113	6.89
Geometry 7	10	15	45	1.1182686	0.041	135	8.22
Geometry 8	105	70	8	1.1182686	0.962	5.74	3.34
Fine Mesh Answer	213	145	577	1.1182686	5.518		

In each case, the iteration converges regardless of the choice of coarse mesh. The leakage of neutrons requires many iterations on the fine mesh to reach convergence. The iteration improvement is the average iteration time on the FM (5.518 sec) divided by the average iteration time for each CM geometry. On the coarsest mesh, the iteration improvement is a factor of 135, but there are many more FM iterations needed to converge the solution making the net acceleration minimal for this case. This results because the heterogeneous properties of each FM node are not preserved in the homogenization process, and as the CM nodes increase in size, these properties become more spatially smeared. For all the geometries, coarsening the radial mesh while holding the axial mesh fixed results in an increased acceleration. The flux is constant along each radial node since the material properties are the same everywhere and the system has an outer reflecting boundary. The flux is not flat in the axial direction and is greatest at $z = -H/2$. It appears that using the material mesh size of 29 in the axial direction yields the greatest benefit. The default number of FM nodes in each of the 29 material regions

is five. Varying the number of FM nodes in each CM caused poorer performance. Because the size of the core is so much greater in the axial direction, maintaining an even distribution of FM cells per CM converges the solution most effectively. This assertion is corroborated by the flux errors seen in Table 3.5.

Table 3.4 – Flux errors for the homogeneous problem with vacuum boundary

Test Problem	Cells		Max Flux % Error
	R	Z	
Geometry 1	20	29	0.125
Geometry 2	45	29	0.122
Geometry 3	20	45	5.66
Geometry 4	45	45	5.66
Geometry 5	10	29	0.122
Geometry 6	20	15	0.278
Geometry 7	10	15	0.278
Geometry 8	105	70	1.54
Fine Mesh Answer	213	145	

Geometries 3 and 4 both yield the largest errors in the flux. The last CM node at the top of the core in both of these cases measures 130 centimeters compared to the other coarse mesh nodes which are 10 cm. These values result from the arbitrary placement of FM nodes in each CM and could stem from homogenizing too coarsely. However, even the most refined case produces a high error, and it has only 4 FMs in the top CM.

3.3 Heterogeneous Infinite Medium Problem:

This heterogeneous problem contains two different materials. These materials alternate each axial plane and are constant across any given radial direction. One is the

fissile material used in the previous two problems, and the other is a non-fissionable material with the properties shown in Table 3.6.

Table 3.5 –Cross-section data for the non-fissionable material

Parameter	Fast Group	Thermal Group
D	9.800900e-01	8.013910e-01
Σ_a	8.054710e-06	1.795710e-04
$\nu\Sigma_f$	0	0
Σ_f	0	0
$\Sigma_s(1 \rightarrow 2 \; or \; 2 \rightarrow 1)$	9.862810e-03	0

Although an analytic solution is not readily obtainable for this problem, it is important to observe the behavior of the CMFD under such conditions. The two materials that are used in the homogenization process have very different properties. The stark difference between the two materials along with the size of the axial mesh, make it difficult for the CMFD operator to converge. Table 3.7 shows that some of the geometries fail to offer any acceleration, and some yield negative fluxes, causing the solution to diverge.

Table 3.6 – Results for the heterogeneous infinite medium problem

Test Problem	Cells		Fine Mesh Iterations	k	Average Iteration Time (sec)	Iteration Improvement	Net Acceleration Factor
	R	Z					
Geometry 1	20	29	13	1.2294141	3.91	2.40	0.849
Geometry 2	45	29	13	1.2294141	6.44	1.46	0.540
Geometry 3	20	45	-	-	-	-	-
Geometry 4	45	45	-	-	-	-	-
Geometry 5	10	29	13	1.2294141	0.0377	249	23.2
Geometry 6	20	15	19	1.229414	1.85	5.08	0.644
Geometry 7	10	15	19	1.229414	0.0934	101	12.2
Geometry 8	105	70	10	1.229414	1.38	6.82	0.805
Fine Mesh Answer	213	145	324	1.2294141	9.39		

Defining the coarse axial mesh such that it spans material boundaries causes the solution to diverge in cases 3 and 4. The most refined CM (case 8) converged in less

CPU time than cases 1 and 2. Case 8 has more unknowns, which results in a bigger matrix. Case 5 has a coarser radial mesh than cases 1 and 2, but they all have the same axial mesh. However, cases 1 and 2 offer no acceleration, and case 5 reduces the convergence time by a factor of 23. All 3 cases required 13 FM iterations to converge. Case 7 has a coarser radial grid by a factor of two compared with case 6. This leads to an improvement by a factor of nearly 20 for both the iteration time and net acceleration factor. This leads to the conclusion that because the flux remains relatively flat in the radial direction and the material is uniform, a very coarse grid a very coarse radial grid can be used. The flux errors are shown in Table 3.7.

Table 3.7 – heterogeneous infinite medium flux

Test Problem	Cells		Fine Mesh Iterations	Flux % Difference
	R	Z		
Geometry 1	20	29	13	0.05481039
Geometry 2	45	29	13	0.054856
Geometry 3	20	45	-	-
Geometry 4	45	45	-	-
Geometry 5	10	29	13	0.05479441
Geometry 6	20	15	19	0.07865781
Geometry 7	10	15	19	0.07866124
Geometry 8	105	70	10	1.44515886
Fine Mesh Answer	213	145	324	

In this test problem, the results for all differ by less than a tenth of a percent compared to the FMFD solution, aside from case 8. This same behavior is observed in the homogeneous problem with a vacuum boundary. Geometries 3, 4, and 8 for both problems have flux solutions that deviate greatly from the FM solution. All three have the most refined axial mesh. We might expect them to yield no acceleration, but they

should be the least likely to have results that do not agree with the actual solution. It appears that homogenizing over material regions is the best approach for the CMFD.

3.4 OECD-PBMR400 Benchmark Problem:

The problem description is given in Table 3.8. The total number of active mesh cells does not include the baffle region around the core or the outer and bottom reflector regions for they are not included in the steady-state calculation.

Table 3.8 - PBMR400 Description

Radial Meshes	213
Theta Meshes	1
Axial Meshes	145
Total Active Meshes	30885
Core Radius	4.62 m
Core Height	17 m
Radial Material Zones	23
Theta Material Zones	1
Axial Material Zones	34

The geometry of the core consists of concentric rings which are not necessarily equidistant from each other. Figure 3.1 shows how the core is separated into different nodes in cylindrical geometry.

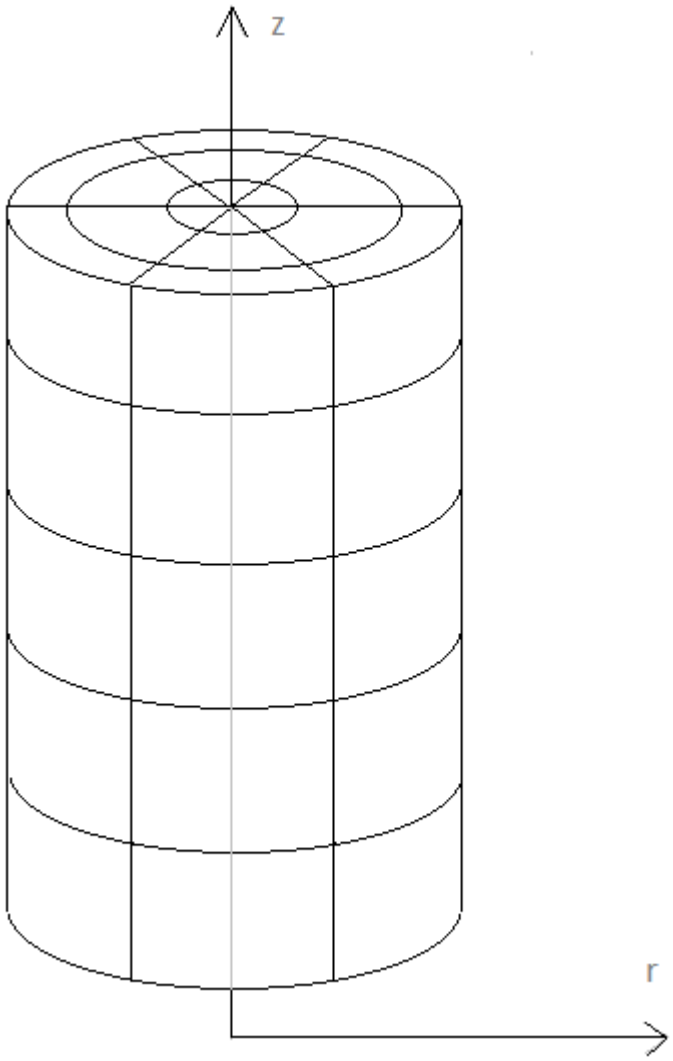

Figure 3.1 – Diagram of node formation in cylindrical geometry

Since there is only one cell in the theta dimension for the PBMR400, it is essentially a 2 dimensional problem. Figures 3.2 and 3.3 give detailed descriptions of the core layout. In figure 3.2, CC is graphite, F is the fuel, V is a void, and TR, BR, and SR are the top, bottom and side reflectors respectively. In an accident scenario, small borated graphite balls can drop into the Reserve Shutdown System (RSS). Another

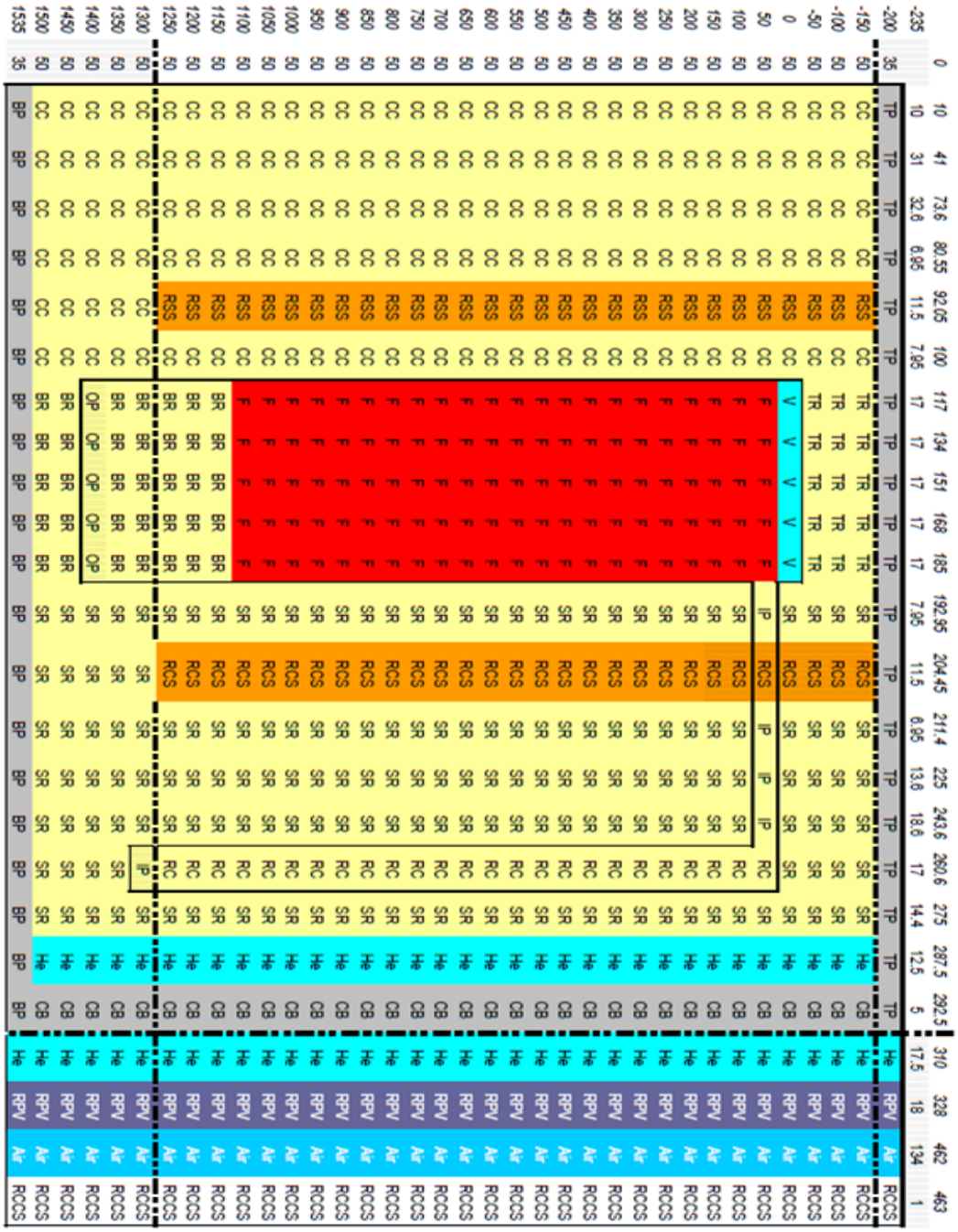

Figure 3.2 – Layout of the PBMR400 core

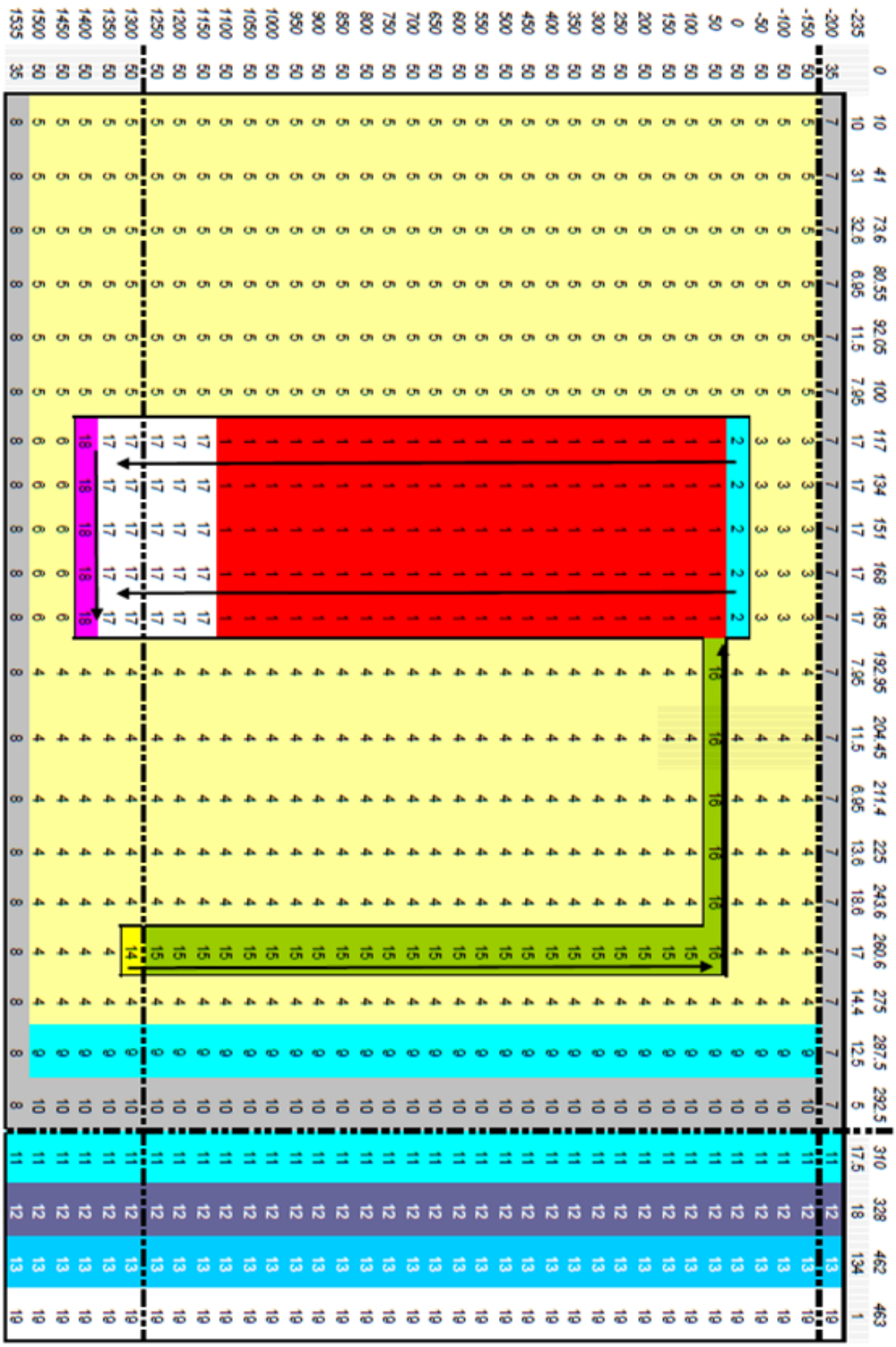

Figure 3.3 –Material distribution in the PBMR400

safety feature of the PBMR design is the Reactor Core Cooling System. If the He coolant is no longer able to remove heat from the system, the decay heat will reach the RCCS and slowly heat vertical water columns. These columns will passively transfer the heat upward out of the core where the water is cooled naturally by air. The core barrel, reactor pressure vessel, and inlet and outlet plena can also be seen in figure 3.2. The numbers outside of the outlined figure are the dimensions in cm in the radial and axial directions and represent the material regions. Figure 3.3 shows the material distribution in the PBMR400 design. The cross section library with all material information is included in Appendix B.

Table 3.9 shows the results of the PBMR400 problem. The eigenvalue, k, is identical for all the variations of the CM used. The only case that fails to yield physical results is geometry 7. This solution initially behaves like the others for about 10

Table 3.9 – PBMR400 Results

Test Problem	Cells		Fine Mesh Iterations	k	Iteration Time (sec)	Iteration Improvement	Net Acceleration Factor
	R	Z					
Geometry 1	20	29	11	1.0046159	0.0382	132	16.1
Geometry 2	45	29	11	1.0046159	0.175	28.8	9.46
Geometry 3	20	45	8	1.0046159	0.0390	129	19.5
Geometry 4	45	45	9	1.0046159	0.137	36.9	9.35
Geometry 5	10	29	13	1.0046159	0.0139	362	17.1
Geometry 6	20	15	419	1.0046159	0.00890	565	0.501
Geometry 7	10	15	-	-	-	-	-
Geometry 8	105	70	8	1.0046159	1.06	4.73	2.14
Geometry 9	40	29	11	1.0046159	0.179	28.1	10.4
Geometry 10	40	58	8	1.0046159	0.185	27.2	7.96
Geometry 11	20	58	8	1.0046159	0.0566	88.9	16.9
Fine Mesh Answer	213	145	266	1.0046159	5.03		

iterations before producing negative fluxes. Using too coarse a grid makes it harder to maintain the higher order FMFD problem characteristics. It can also be seen from geometry 6 that even when the CMFD yields physical results, it may actually slow the convergence rate down. Here, more FM iterations are required with the acceleration than without it.

The benefit of the CMFD modification is evident from the net acceleration factors above. These include both the FM and CM loops used to obtain the final solution. The FM solution converges without the CMFD after 266 iterations. Some of the CM solutions are obtained after over 300 iterations, but these iterations are significantly less computationally expensive. The net acceleration is also the key determinant for gauging the optimal mesh size. In case 1, the coarse mesh is the material region; some cases are based on the material region dimensions while others are completely arbitrary. Case 8 roughly cuts the number of FM cells in half and yields just over a factor of 2 improvement. Using a constant number of radial cells, refining the number of axial cells from case 1 to 6 increases the convergence rate, but then from case 6 to 4 the convergence rate decreases. This demonstrates the balance that exists between the inherent reduction in computational speed in solving a smaller, coarser problem and the lack of accuracy associated with the solution. A much more detailed analysis would need to be carried out to find the optimal mesh sizes, but considering geometry 3 leads to a solution nearly 20 times faster, the CMFD appears to be very successful.

As the mesh size becomes coarser the calculation time is drastically reduced. This is due to a fewer number of unknowns and the smaller matrix that results.

However, this is not seen between geometries 2 and 9. Even though the problems differ by only 5 radial CMs, the FM node distribution in each is quite different. Again, this results from the fact that the number and dimension of the FM nodes cannot be altered in order to maintain the PBMR400 benchmark solution, and this might explain the discrepancy. The same behavior is seen in cases 2 and 4. Also, the variation in the number of radial mesh nodes has a greater impact on the iteration time than the variation in the number of axial meshes.

One other phenomenon that was observed was a loss of diagonal dominance in some of the rows of the matrix. Diagonal dominance is defined algebraically by equation 3.1 for a square matrix A.

$$|\boldsymbol{A_{ii}}| \geq \left|\sum_{\boldsymbol{j \neq i}} \boldsymbol{A_{ij}}\right| \qquad (\mathbf{3.1})$$

The matrix is diagonally dominant if the absolute value of the diagonal element is greater than or equal to the absolute value of the sum of the off diagonal elements in a given row. This quality guarantees that the solution to the matrix equation $Ax = b$ will always be positive. The FMFD numerical approximation of the diffusion equation will always lead to a diagonally dominant matrix. However, the $\widetilde{D}$ that is introduced in the CMFD formulation can destroy diagonal dominance. Table 3.10 shows the number of times diagonal dominance was lost in each variant of the OECD-PBMR400 benchmark problem for the fast and thermal energy groups.

This loss of diagonal dominance arises in regions where homogenization has taken place over coolant channels. These channels are filled with He gas and lead to a

neutron streaming effect (Kim, Cho, Lee, Noh, & Zee, 2007). The neutrons enter these channels, and due to the low density of He, the chance of interaction is extremely small. Because of the directional-dependence of diffusion coefficients, this creates regions susceptible to anisotropic diffusion. PARCS accounts for this by multiplying the diffusion coefficients in these regions by scalars to obtain values comparable to the surrounding material. The computer code PEBBED also does this for the diffusion

Table 3.10 – Loss of diagonal dominance is lost in each energy group.

Test Problem	Cells		Loss of Diagonal Dominance	
	R	Z	Fast Group	Thermal Group
Geometry 1	20	29	0	46
Geometry 2	45	29	2	70
Geometry 3	20	45	0	77
Geometry 4	45	45	131	165
Geometry 5	10	29	0	2
Geometry 6	20	15	0	21
Geometry 7	10	15	-	-
Geometry 8	105	70	149	297
Geometry 9	40	29	1	41
Geometry 10	40	58	18	68
Geometry 11	20	58	3	88
Fine Mesh Answer	213	145	0	0

coefficients allowing for directional dependence (Gouger, 2006). There seems to be a correlation between the anisotropic regions and the loss of diagonal dominance, but the solution is unaffected except on the coarsest mesh, case 7.

Although the convergence criteria on the fission source and eigenvalue is reached in 10 of the 11 cases, it is important to compare the flux solutions from all the cases to that obtained from the higher order FMFD calculation. One of the most

important factors for the safe operation of a reactor is the peak pin power. This allows one to determine the minimum Departure from Nucleate Boiling Ratio (DNBR), which is an operational safety limit set to make sure pool boiling does not occur in the core for LWRs (Todreas). For gas reactors such as the PBMR400, it is still important to know the peak temperature to ensure the structural integrity of the fuel and moderator. The maximum difference in the fluxes generated from the CMFD and standard FMFD for all 30,885 FM nodes is less than 1% except in case 6. Table 3.11 shows the maximum error for each geometry over both energy groups. It appears that homogenizing the meshes over different material regions yields slightly more error.

Table 3.11 – Flux Errors for PBMR400 Problem

Test Problem	Cells		Number of Cells	Flux % Difference
	R	Z		
Geometry 1	20	29	580	0.0938
Geometry 2	45	29	1305	0.129
Geometry 3	20	45	900	0.202
Geometry 4	45	45	2025	0.177
Geometry 5	10	29	290	0.355
Geometry 6	20	15	300	5.45
Geometry 7	10	15	150	-
Geometry 8	105	70	7350	0.316
Geometry 9	40	29	1160	0.0649
Geometry 10	40	58	2320	0.919
Geometry 11	20	58	1160	0.892
Fine Mesh Answer	213	145	30885	0

Chapter 4 Conclusion and Future Work

4.1 Conclusions:

The goal of this work was to integrate the Coarse Mesh Finite Differencing Acceleration technique in cylindrical geometry into the reactor simulation code PARCS. Multiple test problems were used to assess the accuracy and efficiency of this method. The analytic solutions for a homogenous infinite medium problem and homogenous medium with a vacuum boundary condition on the top axial plane were obtained and compared to the solution generated by PARCS. The flux and eigenvalue solutions were in excellent agreement. The benchmarked PBMR400 solution generated with the FMFD method was also achieved with the CMFD. Coarsening the grid eventually led to the production of negative fluxes, but an acceleration was achieved for all of the PBMR400 cases and for most of the others.

4.2 Future Work:

The CMFD was tested on problems with two energy groups representative of fast and thermal neutrons. Future work will include testing CMFD on multi-group problems. This will allow for both spatial and energy acceleration. A 23 group cross section library has already been generated for use in PARCS, and the PARCS_CYL version of the code is being modified to utilize this library. The CMFD will then be able to spatially accelerate the solution for each energy group, or the multi-group parameters can be collapsed from 23 groups down to two. Both accelerations can be used together or separately.

Another area of future work should include analysis of the stability of the CMFD method. Work has been published showing the convergence behavior in 1-D, and there is the possibility that this analysis could be applied to a 3-D problem (Lee, Downer, & Kim, 2004). The results for the 4 problems presented here definitely show that the solution is highly dependent on the number of CMs used and the number of FM nodes per CM. Further investigation into the stability of the method in 3-D might reveal limiting conditions on the CM node size.

Finally, the homogenization of cells with different material properties should be more closely examined, specifically when this homogenization occurs over coolant channels. The neutron streaming effect is apparent, and it is accounted for in PARCS and VSOP94 by multiplying the diffusion coefficients here by scalars. There might be a better way to represent these regions when using CMFD, even though the solutions seem to be unaffected. The homogenization process does not retain the heterogeneities associated with the FM cells, and homogenizing over like materials may prove to be the most robust option. Analyzing the solutions to other complex benchmarked problems may shed more light on this.

References

Cho, J. Y., Joo, H. G., J., Kim, K. S., & Zee, S. Q. (2002). Cell based CMFD formulation for acceleration of whole-core method of characteristics calculations. Journal of the Korean Nuclear Society, 34, 250-258.

Downer, T., Lee, D., Xu, Y., Kozlowski, T. & Staudenmier, J. (2007). PARCS v2.6 U.S. NRC Core Neutronics Simulator Theory Manual. U.S. NRC. Retrieved from https://engineering.purdue.edu/PARCS/Code/Manual/Theory/PDF/PARCS_TheoryManual.pdf

Gouger, H. The Application of the PEBBED Code Suite to the PBMR-400 Coupled Code Benchmark – FY 2006 Annual Report. Prepared for the U.S. Department of Energy Office of Nuclear Energy Under DOE Idaho Operating Office Contract DE-AC07-05ID14517, September, 2006.

Hébert, A. (2009). Applied reactor physics. Montréal: Presses internationales, Polytechnique.

Huda, M.Q., & Obara, T. (2008). Development and testing of analytic models for the pebble bed type HTRs. Annals of Nuclear Energy, 35, 1994-2005.

Idaho National Laboratory. (2004). Next generation nuclear plant – design methods development and validation research and development program plan. Retrieved from http://nuclear.inl.gov/deliverables/docs/ngnp-methods-dev-programc-04-02293.pdf

Ilas, G., Hudson, N. H., Rahnema, F., Ougouag, A. M., & Gouger, H. G. (2006). On the few-group cross-section generation methodology for PBR analysis. Annals of Nuclear Energy, 33, 1058-1070.

Kim, D. S., & Cho, N. Z.(2003). The analytic function expansion nodal (AFEN) method with half-interface averaged fluxes in mixed geometry nodes for analysis of pebble bed modular reactor (PBMR) Cores. Journal of Nuclear Science and Technology, 40, 291-297.

Kim, K., Cho, J., Lee, H. C., Noh, J. M., & Zee, S. Q. (2007). Development of a physics analysis procedure for the prismatic very high temperature gas-cooled reactors. Annals of Nuclear Energy, 34, 849-860.

Lee, D., Downer, T. J., & Kim, Y. (2004). Convergence analysis of the nonlinear coarse-mesh finite difference method for one-dimensional fixed-source neutron diffusion problem. Nuclear Science and Engineering, 147, 127-147.

Lewis, E. E. (2008). Fundamentals of nuclear reactor physics. Boston, MA: Academic Press: Elsevier Inc.

Lewis, E. E., & Miller Jr., W.F. (1993). Computational Methods of Neutron Transport. La Grange Park, Illinois: American Nuclear Society, Inc.

Ougouag, A. M., & Terry, W. K. (2002). Development of a nodal method for the solution of the neutron diffusion equation in general cylindrical geometry. Idaho National Laboratory Bechtel BWXT Idaho, LLC . Retrieved from http://www.inl.gov/technicalpublications/Documents/2670751.pdf

Reitsma, F (2004). The Pebble Bed Modular Reactor Layout and Neutronics Design of

the Equilibrium Cycle. PHYSOR 2004, Chicago, Illinois, April 25-29, on CD -ROM,American Nuclear Society, Lagrange, Park, IL.

Smith, K. S. (2003, April). A Century in Review – A Century Anew. A paper presented at the Nuclear Mathematical and Computational Sciences topical meeting, Gatlinburg, TN.

Smith, K. S. (1986). Assembly homogenization techniques for light water Reactors. Progress in Nuclear Energy, 17, 303-335.

Smith, K. S. (1984). Nodal method storage reduction by nonlinear acceleration. Transactions of the American Nuclear Society, 44, 265-266.

Smith, K. S., Henry, A. F.,& Lorentz, R. A. The determination of homogenized diffusion theory parameters for coarse mesh nodal analysis. (1980). American Nuclear Society, 190, 294-308.

Todreas, N. E., & Kazimi, M. S. (1990). Nuclear systems I: Thermal hydraulic fundamentals. United States: Taylor & Francis Group, LLC.

Woods, B. G., Jackson, R. B.,& Nelson B. L. Scaling analysis for the Very High Temperature Reactor Test Facility at Oregon State University. Prepared for New and Advanced Reactors Branch Office of Nuclear Regulatory Research, U. S. Nuclear Regulatory Commission.

Appendices

Appendix A- Cell Dimensions and Distributions

Table A.1 – Fine mesh cell dimensions

Radial FM cells	Radial Dimension per cell (cm)	Radial FM cells	Radial Dimension per cell (cm)	Radial FM cells	Radial Dimension per cell (cm)	Axial FM cells	Axial Dimension per cell (cm)
1-4	2.5	83-102	.85	185-191	2.4285714	1-5	50
5-12	3.875	103-122	.85	192-198	2.0571429	6-150	10
13-20	4.075	123-142	.85	199-205	1.7857142		
21-26	1.1583333	143-152	.795	206-213	.625		
27-32	1.9166667	153-162	1.15	214	17.5		
33-42	.795	163-170	.86875	215	18		
43-62	.85	171-177	1.9428571	216	134		
63-82	.85	178-184	2.6571428				

Table A.2 – Geometry 1 and material region dimensions

Radial CM cell	Radial Dimension per cell (cm)	Number of FM cells in each Radial CM cell	Axial CM cell	Axial Dimension per cell (cm)	Number of FM cells in each Axial CM cell
1	10.0	4	1-5	50	1
2	31.0	8	6-34	50	5
3	32.6	8			
4	6.95	6			
5	11.5	6			
6	7.95	10			
7	17.0	20			
8	17.0	20			
9	17.0	20			
10	17.0	20			
11	17.0	20			
12	7.95	10			
13	11.5	10			

14	6.95	8			
15	13.6	7			
16	18.6	7			
17	17.0	7			
18	14.4	7			
19	12.5	7			
20	5.0	8			
21	17.5	1			
22	18.0	1			
23	134.0	1			

Table A.3 – Geometry 2 dimensions

Radial CM cells	Radial Dimension per cell (cm)	Number of FM cells in each Radial CM cell	Axial CM cells	Axial Dimension per cell (cm)	Number of FM cells in each Axial CM cell
1	10	4	1-5	50	1
2	15.5	4	6-34	50	5
3	15.5	4			
4	16.3	4			
5	16.3	4			
6	4.63333333	4			
7	6.15	4			
8	7.66666667	4			
9	3.18	4			
10	3.18	4			
11	3.29	4			
12-30	3.4	4			
31	5.1	6			
32	5.1	6			
33	5.1	6			
34	4.99	6			
35	4.77	6			
36	6.19	6			
37	6.9	6			
38	6.95	8			
39	13.6	7			
40	18.6	7			

41	17.0	7			
42	14.4	7			
43	12.5	7			
44	2.5	4			
45	2.5	4			
46	17.5	1			
47	18.0	1			
48	134.0	1			

Table A.4 – Geometry 3 dimensions

Radial CM cells	Radial Dimension per cell (cm)	Number of FM cells in each Radial CM cell	Axial CM cells	Axial Dimension per cell (cm)	Number of FM cells in each Axial CM cell
1	10.0	4	1-5	50	1
2	31.0	8	6-49	30	3
3	32.6	8	50	130	13
4	6.95	6			
5	11.5	6			
6	7.95	10			
7	17.0	20			
8	17.0	20			
9	17.0	20			
10	17.0	20			
11	17.0	20			
12	7.95	10			
13	11.5	10			
14	6.95	8			
15	13.6	7			
16	18.6	7			
17	17.0	7			
18	14.4	7			
19	12.5	7			
20	5.0	8			
21	17.5	1			
22	18.0	1			
23	134.0	1			

Table A.5 – Geometry 4 dimensions

Radial CM cells	Radial Dimension per cell (cm)	Number of FM cells in each Radial CM cell	Axial CM cells	Axial Dimension per cell (cm)	Number of FM cells in each Axial CM cell
1	17.75	6	1-5	50	1
2	23.25	6	6-49	30	3
3	24.45	6	50	130	13
4	12.7833333	6			
5	9.98333333	6			
6	7.01333333	6			
7	4.77	6			
8-23	5.1	6			
24	4.99	6			
25	4.77	6			
26	6.19	6			
27	6.9	6			
28	5.2125	6			
29	9.50892857	6			
30	13.8	6			
31	25.2	10			
32	20.4	10			
33	1.78571429	1			
34	1.78571429	1			
35	1.78571429	1			
36	1.78571429	1			
37	1.78571429	1			
38	0.625	1			
39	0.625	1			
40	0.625	1			
41	0.625	1			
42	0.625	1			
43	0.625	1			
44	0.625	1			
45	0.625	1			
46	17.5	1			
47	18.0	1			

48	134.0	1			

Table A.6 – Geometry 5 dimensions

Radial CM cells	Radial Dimension per cell (cm)	Number of FM cells in each Radial CM cell	Axial CM cells	Axial Dimension per cell (cm)	Number of FM cells in each Axial CM cell
1	79.3916667	25	1-5	50	1
2	27.4083333	25	6-34	50	5
3	21.25	25			
4	21.25	25			
5	21.25	25			
6	20.81	25			
7	29.7542857	25			
8	57.45714286	25			
9	9.553571429	6			
10	4.375	7			
11	17.5	1			
12	18.0	1			
13	134.0	1			

Table A.7 – Geometry 6 dimensions

Radial CM cells	Radial Dimension per cell (cm)	Number of FM cells in each Radial CM cell	Axial CM cells	Axial Dimension per cell (cm)	Number of FM cells in each Axial CM cell
1	10.0	4	1-5	50	1
2	31.0	8	6-19	100	10
3	32.6	8	20	50	5
4	6.95	6			
5	11.5	6			
6	7.95	10			
7	17.0	20			
8	17.0	20			
9	17.0	20			
10	17.0	20			
11	17.0	20			

12	7.95	10			
13	11.5	10			
14	6.95	8			
15	13.6	7			
16	18.6	7			
17	17.0	7			
18	14.4	7			
19	12.5	7			
20	5.0	8			
21	17.5	1			
22	18.0	1			
23	134.0	1			

Table A.8 – Geometry 7 dimensions

Radial CM cells	Radial Dimension per cell (cm)	Number of FM cells in each Radial CM cell	Axial CM cells	Axial Dimension per cell (cm)	Number of FM cells in each Axial CM cell
1	79.3916667	25	1-5	50	1
2	27.4083333	25	6-19	100	10
3	21.25	25	20	50	5
4	21.25	25			
5	21.25	25			
6	20.81	25			
7	29.7542857	25			
8	57.45714286	25			
9	9.553571429	6			
10	4.375	7			
11	17.5	1			
12	18.0	1			
13	134.0	1			

Table A.9 – Geometry 8 dimensions

Radial CM cells	Radial Dimension per cell (cm)	Number of FM cells in each Radial CM cell	Axial CM cells	Axial Dimension per cell (cm)	Number of FM cells in each Axial CM cell
1-2	5.0	2	1-5	50	1

3-6	7.75	2	6-73	20	2
7-10	8.15	2	74	50	5
11-13	2.31666667	2	75	40	4
14-16	3.83333333	2			
17-21	1.59	2			
22-71	1.7	2			
72-76	1.59	2			
77-81	2.3	2			
82-85	1.7375	2			
86-88	3.88571429	2			
90-92	5.31428571	2			
93-95	4.85714286	2			
96	4.48571429	2			
97-99	4.11428571	2			
100-102	3.57142857	2			
103	2.41071429	2			
104	1.25	2			
105	3.125	5			
106	17.5	1			
107	18.0	1			
108	134.0	1			

Table A.10 – Geometry 9 dimensions

Radial CM cells	Radial Dimension per cell (cm)	Number of FM cells in each Radial CM cell	Axial CM cells	Axial Dimension per cell (cm)	Number of FM cells in each Axial CM cell
1	17.75	6	1-5	50	1
2	23.25	6	6-34	50	5
3	24.45	6			
4	12.7833333	6			
5	9.98333333	6			
6	7.01333333	6			
7	4.77	6			
8-23	5.1	6			
24	4.99	6			
25	4.77	6			

26	6.19	6			
27	6.9	6			
28	5.2125	6			
29	9.50892857	6			
30	13.8	6			
31	10.6285714	4			
32	9.71428571	4			
33	9.34285714	4			
34	8.22857142	4			
35	5.9	3			
36	5.35714285	3			
37	5.35714285	3			
38	1.875	3			
39	1.875	3			
40	1.25	2			
41	17.5	1			
42	18.0	1			
43	134.0	1			

Table A.11 – Geometry 10 dimensions

Radial CM cells	Radial Dimension per cell (cm)	Number of FM cells in each Radial CM cell	Axial CM cells	Axial Dimension per cell (cm)	Number of FM cells in each Axial CM cell
1	17.75	6	1-5	50	1
2	23.25	6	6-36	30	3
3	24.45	6	37-62	20	2
4	12.7833333	6			
5	9.98333333	6			
6	7.01333333	6			
7	4.77	6			
8-23	5.1	6			
24	4.99	6			
25	4.77	6			
26	6.19	6			
27	6.9	6			

28	5.2125	6			
29	9.50892857	6			
30	13.8	6			
31	10.6285714	4			
32	9.71428571	4			
33	9.34285714	4			
34	8.22857142	4			
35	5.9	3			
36	5.35714285	3			
37	5.35714285	3			
38	1.875	3			
39	1.875	3			
40	1.25	2			
41	17.5	1			
42	18.0	1			
43	134.0	1			

Table A.12 – Geometry 11 dimensions

Radial CM cell	Radial Dimension per cell (cm)	Number of FM cells in each Radial CM cell	Axial CM cell	Axial Dimension per cell (cm)	Number of FM cells in each Axial CM cell
1	10.0	4	1-5	50	1
2	31.0	8	6-36	30	3
3	32.6	8	37-62	20	2
4	6.95	6			
5	11.5	6			
6	7.95	10			
7	17.0	20			
8	17.0	20			
9	17.0	20			
10	17.0	20			
11	17.0	20			
12	7.95	10			
13	11.5	10			
14	6.95	8			

15	13.6	7			
16	18.6	7			
17	17.0	7			
18	14.4	7			
19	12.5	7			
20	5.0	8			
21	17.5	1			
22	18.0	1			
23	134.0	1			

Appendix B – PBMR400 Material Distribution by Axial Planes

Numbers correspond to material zones which can be found in Frederik Reitsma's paper in the references.

```
Planar_Reg 1
  132 132 132 132 163 124 115  115  115  115  115 143 188 151 151
  154 154 154 189 190
Planar_Reg 2
  132 132 132 132 163 124 115  115  115  115  115 143 188 151 151
  154 154 154 189 190
Planar_Reg 3
  132 132 132 132 163 124 114  114  114  114  114 143 188 151 151
  154 154 154 189 190
Planar_Reg 4
  134 134 134 131 162 123 22   44   66   88   110 142 187 151 151
  153 153 153 189 190
Planar_Reg 5
  134 134 134 131 162 123 21   43   65   87   109 142 186 151 151
  153 153 153 189 190
Planar_Reg 6
  134 134 134 130 161 122 20   42   64   86   108 141 185 150 150
  153 153 153 189 190
Planar_Reg 7
  134 134 134 130 161 122 19   41   63   85   107 141 184 150 150
  153 153 153 189 190
Planar_Reg 8
  134 134 134 130 161 122 18   40   62   84   106 141 183 150 150
  153 153 153 189 190
Planar_Reg 9
  134 134 134 129 160 121 17   39   61   83   105 140 182 149 149
  153 153 153 189 190
Planar_Reg 10
  134 134 134 129 160 121 16   38   60   82   104 140 181 149 149
  153 153 153 189 190
Planar_Reg 11
  134 134 134 129 160 121 15   37   59   81   103 140 180 149 149
  153 153 153 189 190
Planar_Reg 12
  134 134 134 129 160 121 14   36   58   80   102 140 179 149 149
  153 153 153 189 190
Planar_Reg 13
  134 134 134 128 159 120 13   35   57   79   101 139 178 148 148
  153 153 153 189 190
Planar_Reg 14
  134 134 134 128 159 120 12   34   56   78   100 139 177 148 148
  153 153 153 189 190
Planar_Reg 15
  134 134 134 128 159 120 11   33   55   77   99  139 176 148 148
  153 153 153 189 190
Planar_Reg 16
```

```
  134 134 134 128 159 120 10   32   54   76   98  139 175 148 148
  153 153 153 189 190
Planar_Reg 17
  134 134 134 127 158 119 9    31   53   75   97  138 174 147 147
  153 153 153 189 190
Planar_Reg 18
  134 134 134 127 158 119 8    30   52   74   96  138 173 147 147
  153 153 153 189 190
Planar_Reg 19
  134 134 134 127 158 119 7    29   51   73   95  138 172 147 147
  153 153 153 189 190
Planar_Reg 20
  134 134 134 127 158 119 6    28   50   72   94  138 171 147 147
  153 153 153 189 190
Planar_Reg 21
  134 134 134 126 157 118 5    27   49   71   93  137 170 146 146
  153 153 153 189 190
Planar_Reg 22
  134 134 134 126 157 118 4    26   48   70   92  137 169 146 146
  153 153 153 189 190
Planar_Reg 23
  134 134 134 126 157 118 3    25   47   69   91  137 168 146 146
  153 153 153 189 190
Planar_Reg 24
  134 134 134 125 156 117 2    24   46   68   90  136 167 145 145
  153 153 153 189 190
Planar_Reg 25
  134 134 134 125 156 117 1    23   45   67   89  136 166 145 145
  153 153 153 189 190
Planar_Reg 26
  133 133 133 133 155 116 111  111  111  111  111 135 165 144 144
  152 152 152 189 190
Planar_Reg 27
  133 133 133 133 155 116 112  112  112  112  112 135 164 144 144
  152 152 152 189 190
Planar_Reg 28
  133 133 133 133 155 116 113  113  113  113  113 135 164 144 144
  152 152 152 189 190
Planar_Reg 29
  133 133 133 133 155 116 113  113  113  113  113 135 164 144 144
  152 152 152 189 190
```

Appendix C – Code

```
SUBROUTINE cmfdaccel2
      USE impl
      USE params
      USE itrcntl
      USE xsec
      USE geovar
      USE coupl
      USE lscoef
      USE solvec
      USE cntl, ONLY: linearsolver
      IMPLICIT NONE
      INTEGER(ivp) :: i, j, k, lf, l, m, kb, kt, le, lw, ls, ln,
jp, lp, md, mc, ib, ie, js
      INTEGER(ivp) :: icm, jcm, kcm, lcm, l3, ii, jj, kk, ncmxyz,
ncmxy, &
                      m1, z, kbot, ktop, kbeg, kend, ibc, ka, ja,
la, ia, &
                      icomp, ipr, lcmold
      REAL(rvp) :: albf, bcf
      REAL(rvp) :: hcmxi, hcmxiw, hcmyj, hcmyjn, hcmzk, hcmzkb,
hcmxie, &
                   hcmxy, hcmxz, hcmyz, hcmyzL, hcmyzR, hzk, hyj,
hxi, &
                   hyz, hxz, hxy, hyzL, hyzR, difl, diflw, difln,
diflb, diflz, diflzb, &
                   dddccfz1, dddccfr1
      LOGICAL, SAVE :: betapinit=.TRUE.

      ncmxy = (ncmxe(1)-ncmxs(1)+1)*ncmy
      ncmxyz = ncmxy*(ncmze-ncmzs+1)

!homogenize parameters to coarse mesh
      cmphicm=zero     !homogenized flux
      cmxstfcm=zero    !homogenized total x-sec
      cmxscatcm=zero   !homogenized scattering x-sec
      cmscatvcm=zero
      cmxstrfcm=zero   !homogenized transport x-sec
      cmrvdeltcm=zero  !homogenized 1/v*dt
      cmxsnffcm=zero   !homogenized fission x-sec
      cmxschicm=zero   !homogenized fission distribution
      cmbetapcm=zero   !homogenized prompt neutron fraction
      cmafcm=zero      !homogenized nu fission x-sec
      cmpsicm=zero     !homogenized source term
      cmxsdfcmold=zero
      cmxsdzcmold=zero


      DO k = nzs, nze
         kcm = ktokcm(k)
           DO j = 1, ny
```

```
            jcm = jtojcm(j)
            DO i = nxs(j), nxe(j)
             icm = itoicm(i)
             l = nodel(i,j)
                  lcm = nodelcm(icm,jcm)
                  DO m = 1, ng

cmphicm(m,lcm,kcm)=cmphicm(m,lcm,kcm)+phi(m,l,k)*volnode(l,k)
                    cmxstfcm(m,lcm,kcm)=cmxstfcm(m,lcm,kcm)+
&

xstf(m,l,k)*phi(m,l,k)*volnode(l,k)
                    cmxstrfcm(m,lcm,kcm)=cmxstrfcm(m,lcm,kcm)+
&

xstrf(m,l,k)*phi(m,l,k)*volnode(l,k)
                    cmxsnffcm(m,lcm,kcm)=cmxsnffcm(m,lcm,kcm)+
&

xsnff(m,l,k)*phi(m,l,k)*volnode(l,k)
                    cmxschicm(m,lcm,kcm)=cmxschicm(m,lcm,kcm)+
&

xschi(m,l,k)*phi(m,l,k)*volnode(l,k)
                    cmrvdeltcm(m,lcm,kcm)=cmrvdeltcm(m,lcm,kcm)+
&

rvdelt(m,l,k)*phi(m,l,k)*volnode(l,k)
       !            cmbetapcm(lcm,kcm)=cmbetapcm(lcm,kcm)+
&
       !
betap(l,k)*phi(m,l,k)*volnode(l,k)
                    cmxsdfcmold(m,lcm,kcm)=cmxsdfcmold(m,lcm,kcm)
+ (1/xsdf(m,l,k))*volnode(l,k)
                    cmxsdzcmold(m,lcm,kcm)=cmxsdzcmold(m,lcm,kcm)
+ (1/xsdz(m,l,k))*volnode(l,k)
                    DO m1 = 1, ng
                       IF(m1.NE.m) THEN

cmxscatcm(m,m1,lcm,kcm)=cmxscatcm(m,m1,lcm,kcm)+            &

xscat(m,m1,l,k)*phi(m,l,k)*volnode(l,k)
                          END IF
                    END DO
                  END DO
            END DO
          END DO
      END DO

        DO k = ncmzs, ncmze
```

```
           DO j = 1, ncmy
              DO i = ncmxs(j), ncmxe(j)
                 l = nodelcm(i,j)
                 DO m = 1, ng
                    cmphicm(m,l,k)=cmphicm(m,l,k)/volcm(l,k)
                    fmtocmphi(m,l,k)=cmphicm(m,l,k)

cmxstfcm(m,l,k)=cmxstfcm(m,l,k)/(cmphicm(m,l,k)*volcm(l,k))

cmxstrfcm(m,l,k)=cmxstrfcm(m,l,k)/(cmphicm(m,l,k)*volcm(l,k))

cmxsnffcm(m,l,k)=cmxsnffcm(m,l,k)/(cmphicm(m,l,k)*volcm(l,k))

cmxschicm(m,l,k)=cmxschicm(m,l,k)/(cmphicm(m,l,k)*volcm(l,k))

cmrvdeltcm(m,l,k)=cmrvdeltcm(m,l,k)/(cmphicm(m,l,k)*volcm(l,k))
       !
cmbetapcm(l,k)=cmbetapcm(l,k)/(cmphicm(m,l,k)*volcm(l,k))
                    cmxsdfcm(m,l,k)=one/(cmxstrfcm(m,l,k) *
3.0_rvp) !r/theta diffusion coefficeint
                    cmxsdzcm(m,l,k)=one/(cmxstrfcm(m,l,k) *
3.0_rvp) !z diffusion coefficient

cmxsdfcmold(m,l,k)=volcm(l,k)/cmxsdfcmold(m,l,k)     !to fix void

cmxsdzcmold(m,l,k)=volcm(l,k)/cmxsdzcmold(m,l,k)     !to fix void
                    cmxsdfcm(m,l,k)=cmxsdfcmold(m,l,k)  !to fix
void
                    cmxsdzcm(m,l,k)=cmxsdzcmold(m,l,k)  !to fix
void
                    DO m1 = 1, ng
                       IF(m1.NE.m) THEN

cmxscatcm(m,m1,l,k)=cmxscatcm(m,m1,l,k)/(cmphicm(m,l,k)*volcm(l,k)
)
                       END IF
                    END DO
!
cmbetapcm(l,k)=cmbetapcm(l,k)/((cmphicm(1,l,k)+cmphicm(2,l,k))*vol
cm(l,k))
                 END DO
              END DO
           END DO
        END DO



!calculate D-tildes on coarse mesh
           DO k = ncmzs, ncmze
              kb = k-1
```

```
            hcmzk = hcmz(k)
            hcmzkb = hcmz(kb)
            DO j = 1, ncmy
               hcmyj = hcmy(j)
               hcmyjn = hcmy(j-1)
               IF(j.EQ.1 .AND. ibcy(1).EQ.0 .AND. hcmyj.NE.hcmyjn)
hcmyjn=hcmyj
               DO i = ncmxs(j), ncmxe(j)
                  hcmxi = hcmx(i)
                  hcmxiw = hcmx(i-1)
                  l = nodelcm(i,j)
                  lw = nodelcm(i-1,j)
                  ln = nodelcm(i,j-1)
                  DO m =1, ng
                     difl=cmxsdfcm(m,l,k)
                     diflw=cmxsdfcm(m,lw,k)
                     difln=cmxsdfcm(m,ln,k)
                     diflb=cmxsdfcm(m,l,kb)
                     IF(hcmxiw.EQ.0) THEN
                       dfwcm(m,l,k)=0
                     ELSE

dfwcm(m,l,k)=2.0_rvp*difl*diflw/(difl*hcmxiw+diflw*hcmxi)
                     END IF

dfncm(m,l,k)=2.0_rvp*difl*difln/(difl*hcmyjn+difln*hcmyj)
                     dfncm(m,l,k)=dfncm(m,l,k)/centerxcm(i)
                     diflz=cmxsdzcm(m,l,k)
                     diflzb=cmxsdzcm(m,l,kb)

dfbcm(m,l,k)=2.0_rvp*diflz*diflzb/(diflz*hcmzkb+diflzb*hcmzk)
                  END DO
               END DO
            END DO
         END DO
!           incorporate external boundary conditions
!---------------------x/r - direction
         DO ib = 1, 2
            DO k = ncmzs, ncmze
               DO j = 1, ncmy
                  IF(ibcx(ib) .EQ. 0) THEN
                    albf = zero
                  ELSE IF(ibcx(ib) .EQ. 1) THEN
                    albf = big
                  ELSE IF(ibcx(ib) .EQ. 2) THEN
                    albf = half
                  END IF
                  IF(ib.EQ.1) THEN
                    i = ncmxs(j)
                    ie = ncmxs(j)
```

```
                  ELSE
                    i = ncmxe(j)
                    ie = i + 1
                  END IF
                  hcmxi = hcmx(i)
                  l = nodelcm(i,j)
                  le = nodelcm(ie,j)
                  DO m = 1, ng
                     dfwcm(m,le,k)=bcf(cmxsdfcm(m,l,k),albf,hcmxi)
                  END DO
                  IF(ib.EQ.1 .AND. r_inner.EQ.zero) THEN
                    DO m = 1, ng
                       dfwcm(m,le,k)=zero
                    END DO
                  END IF
               END DO
            END DO
         END DO
!--------------------y/theta direction
         DO ib = 1, 2
            DO k = ncmzs, ncmze
               DO i = ncmxs(1), ncmxe(1)
                  IF(ibcy(ib).EQ.0) THEN
                     albf = zero
                  ELSE IF(ibcy(ib) .EQ. 1) THEN
                     albf = big
                  ELSE IF(ibcy(ib) .EQ. 2) THEN
                     albf = half
                  END IF
                  IF(ib.EQ.1) THEN
                     j = ncmys(i)
                     js = ncmys(i)
                     jp = ncmye(i)
                  ELSE
                     j = ncmye(i)
                     js = j + 1
                     jp = ncmys(i)
                  END IF
                  hcmyj = hcmy(j)
                  hcmyj = hcmyj * centerxcm(i)
                  hcmyjn = hcmy(jp) * centerxcm(i)
                  l = nodelcm(i,j)
                  ls = nodelcm(i,js)
                  lp = nodelcm(i,jp)
                  IF(ibcy(ib) == 3) THEN
                     DO m = 1, ng
                        difl=cmxsdfcm(m,l,k)
                        difln=cmxsdfcm(m,lp,k)

dfnpcm(m,lp,k)=2.0_rvp*difl*difln/(difl*hcmyjn+difln*hcmyj)
```

```
                        dfncm(m,ls,k) = zero
                     END DO
                  ELSE
                     DO m = 1, ng
                        dfncm(m,ls,k) =
bcf(cmxsdfcm(m,l,k),albf,hcmyj)
                     END DO
                  END IF
               END DO
            END DO
         END DO
!----------------- z-direction
         DO ib = 1, 2
            IF(ibcz(ib).EQ.0) THEN
               albf = zero
            ELSE IF(ibcz(ib) .EQ. 1) THEN
               albf = big
            ELSE IF(ibcz(ib) .EQ. 2) THEN
               albf = half
            END IF
            IF(ib.EQ.1) THEN
               k = ncmzs
               kt = ncmzs
            ELSE
               k = ncmze
               kt = k + 1
            END IF
            hcmzk = hcmz(k)
            DO l = 1, ncmxy
               DO m = 1, ng
                  dfbcm(m,l,kt)=bcf(cmxsdzcm(m,l,k),albf,hcmzk)
               END DO
            END DO
         END DO



!calculate fm currents
         DO k=nzs, nze
            kt=k+1
            hzk=hz(k)
            DO j=1,ny
               hyj=hy(j)
               hyz=hzk*hyj
               js=j+1
               DO i=nxs(j), nxe(j)
                  hxi=hx(i)
                  hyzL=(centerx(i)-hx(i)/2.0_rvp)*hyz
                  hyzR=(centerx(i)+hx(i)/2.0_rvp)*hyz
                  hxy=hxi*hyj
```

```
            hxy=hxy*centerx(i)
            hxz=hxi*hzk
            l=nodel(i,j)
            le=nodel(i+1,j)
            ls=nodel(i,js)
            DO m=1,ng
               accw(m,l,k)=dfw(m,l,k)*hyzL
               accn(m,l,k)=dfn(m,l,k)*hxz
               accb(m,l,k)=dfb(m,l,k)*hxy
               acce(m,l,k)=dfw(m,le,k)*hyzR
               accs(m,l,k)=dfn(m,ls,k)*hxz
               acct(m,l,k)=dfb(m,l,kt)*hxy
               accbbb(m,l,k)=dnb(m,l,k)*hxy
               acctbb(m,l,k)=dnb(m,l,kt)*hxy
            END DO
         END DO
      END DO
   END DO

   jw=zero
   je=zero
   jn=zero
   jss=zero
   jb=zero
   jt=zero
   kbot=nzs
   ktop=nze
   DO k=kbot, ktop
      DO j=nys(1), nye(1)
         i=nxs(j)
         l=nodel(i,j)
         lw=nodel(i-1,j) !lm
         le=nodel(i+1,j) !lp
         ibc=ibcx(1)
         IF(ibc.EQ.0) THEN
           jw(1,l,k)=zero
           jw(2,l,k)=zero
         ELSE IF(ibc.EQ.1 .OR. ibc.EQ.2) THEN
           jw(1,l,k)=-(accw(1,l,k))*phi(1,l,k)
           jw(2,l,k)=-(accw(2,l,k))*phi(2,l,k)
         END IF
         je(1,l,k)=acce(1,l,k)*(phi(1,l,k)-phi(1,le,k))
         je(2,l,k)=acce(2,l,k)*(phi(2,l,k)-phi(2,le,k))

         i=nxe(j)
         l=nodel(i,j)
         lw=nodel(i-1,j)
         le=nodel(i+1,j)
         ibc=ibcx(2)
         IF(ibc.EQ.0) THEN
```

```
      je(1,l,k)=zero
      je(2,l,k)=zero
    ELSE IF(ibc.EQ.1 .OR. ibc.EQ.2) THEN
      je(1,l,k)=acce(1,l,k)*phi(1,l,k)
      je(2,l,k)=acce(2,l,k)*phi(2,l,k)
    END IF
    jw(1,l,k)=-accw(1,l,k)*(phi(1,l,k)-phi(1,lw,k))
    jw(2,l,k)=-accw(2,l,k)*(phi(2,l,k)-phi(2,lw,k))

    DO i=nxs(j)+1, nxe(j)-1
       l=nodel(i,j)
       lw=nodel(i-1,j)
       le=nodel(i+1,j)
       jw(1,l,k)=-accw(1,l,k)*(phi(1,l,k)-phi(1,lw,k))
       jw(2,l,k)=-accw(2,l,k)*(phi(2,l,k)-phi(2,lw,k))
       je(1,l,k)=acce(1,l,k)*(phi(1,l,k)-phi(1,le,k))
       je(2,l,k)=acce(2,l,k)*(phi(2,l,k)-phi(2,le,k))
    END DO
  END DO

  DO i=nxs(1), nxe(1)
    j=nys(i)
    l=nodel(i,j)
    ln=nodel(i,j-1)
    ls=nodel(i,j+1)
    ibc=ibcy(1)
    IF(ibc.EQ.0) THEN
      jn(1,l,k)=zero
      jn(2,l,k)=zero
    ELSE IF(ibc.EQ.1 .OR. ibc.EQ.2) THEN
      jn(1,l,k)=-accn(1,l,k)*phi(1,l,k)
      jn(2,l,k)=-accn(2,l,k)*phi(2,l,k)
    END IF
    IF(ny .GT. 1) THEN
      jss(1,l,k)=accs(1,l,k)*(phi(1,l,k)-phi(1,ls,k))
      jss(2,l,k)=accs(2,l,k)*(phi(2,l,k)-phi(2,ls,k))
    END IF
    j=nye(i)
    l=nodel(i,j)
    ls=nodel(i,j+1)
    ln=nodel(i,j-1)
    ibc=ibcy(2)
    IF(ibc.EQ.0) THEN
      jss(1,l,k)=zero
      jss(2,l,k)=zero
    ELSE IF(ibc.EQ.1 .OR. ibc.EQ.2) THEN
      jss(1,l,k)=accs(1,l,k)*phi(1,l,k)
      jss(2,l,k)=accs(2,l,k)*phi(2,l,k)
    END IF
    IF(ny .GT. 1) THEN
```

```
                jn(1,l,k)=-accn(1,l,k)*(phi(1,l,k)-phi(1,ln,k))
                jn(2,l,k)=-accn(2,l,k)*(phi(2,l,k)-phi(2,ln,k))
              END IF
              DO j=nys(i)+1, nye(i)-1
                 l=nodel(i,j)
                 ln=nodel(i,j-1)
                 ls=nodel(i,j+1)
                 jn(1,l,k)=-accn(1,l,k)*(phi(1,l,k)-phi(1,ln,k))
                 jn(2,l,k)=-accn(2,l,k)*(phi(2,l,k)-phi(2,ln,k))
                 jss(1,l,k)=accs(1,l,k)*(phi(1,l,k)-phi(1,ls,k))
                 jss(2,l,k)=accs(2,l,k)*(phi(2,l,k)-phi(2,ls,k))
              END DO
           END DO
         END DO

         k=nzs
         kt=k+1
         IF(ibcz(1).EQ.0) THEN
           DO l=1,nxy
              jb(1,l,k)=zero
              jb(2,l,k)=zero
           END DO
         ELSE IF(ibcz(1).EQ.1 .OR. ibcz(1).EQ.2) THEN
           DO l=1,nxy
              jb(1,l,k)=-(accb(1,l,k)+accbbb(1,l,k))*phi(1,l,k)
              jb(2,l,k)=-(accb(2,l,k)+accbbb(2,l,k))*phi(2,l,k)
           END DO
         END IF
         DO l=1,nxy
            jt(1,l,k)=acct(1,l,k)*(phi(1,l,k)-phi(1,l,kt)) -
acctbb(1,l,k)*(phi(1,l,k)+phi(1,l,kt))
            jt(2,l,k)=acct(2,l,k)*(phi(2,l,k)-phi(2,l,kt)) -
acctbb(2,l,k)*(phi(2,l,k)+phi(2,l,kt))
         END DO

         kbeg=nzs+1
         k=nze
         kb=k-1
         IF(ibcz(2).EQ.0) THEN
           DO l=1,nxy
              jt(1,l,k)=zero
              jt(2,l,k)=zero
           END DO
         ELSE IF(ibcz(2).EQ.1 .OR. ibcz(2).EQ.2) THEN
           DO l=1,nxy
              jt(1,l,k)=(acct(1,l,k)-acctbb(1,l,k))*phi(1,l,k)
              jt(2,l,k)=(acct(2,l,k)-acctbb(2,l,k))*phi(2,l,k)
           END DO
         END IF
         DO l=1,nxy
```

```
            jb(1,l,k)=-accb(1,l,k)*(phi(1,l,k)-phi(1,l,kb)) -
accbbb(1,l,k)*(phi(1,l,k)+phi(1,l,kb))
            jb(2,l,k)=-accb(2,l,k)*(phi(2,l,k)-phi(2,l,kb)) -
accbbb(2,l,k)*(phi(2,l,k)+phi(2,l,kb))
         END DO

         kend=nze-1
         DO k=kbeg,kend
            kb=k-1
            kt=k+1
            DO l=1, nxy
               jb(1,l,k)=-accb(1,l,k)*(phi(1,l,k)-phi(1,l,kb)) -
accbbb(1,l,k)*(phi(1,l,k)+phi(1,l,kb))
               jb(2,l,k)=-accb(2,l,k)*(phi(2,l,k)-phi(2,l,kb)) -
accbbb(2,l,k)*(phi(2,l,k)+phi(2,l,kb))
               jt(1,l,k)=acct(1,l,k)*(phi(1,l,k)-phi(1,l,kt)) -
acctbb(1,l,k)*(phi(1,l,k)+phi(1,l,kt))
               jt(2,l,k)=acct(2,l,k)*(phi(2,l,k)-phi(2,l,kt)) -
acctbb(2,l,k)*(phi(2,l,k)+phi(2,l,kt))
            END DO
         END DO

         jwcm=zero
         jecm=zero
         jncm=zero
         jscm=zero
         jbcm=zero
         jtcm=zero
!z cm current
         DO kcm = ncmzs, ncmze
            kb = ktokmap(kcm)
            kt = ktokmap(kcm+1)-1
            IF(kcm.EQ.ncmze) kt=nze
            DO j = 1, ny
               jcm = jtojcm(j)
               DO i = nxs(j), nxe(j)
                  icm = itoicm(i)
                  l = nodel(i,j)
                  lcm = nodelcm(icm,jcm)
                  DO m = 1, ng
                     jbcm(m,lcm,kcm)=jbcm(m,lcm,kcm) + jb(m,l,kb)
                     jtcm(m,lcm,kcm)=jtcm(m,lcm,kcm) + jt(m,l,kt)
                  END DO
               END DO
            END DO
         END DO
!theta cm current
         DO k = nzs, nze
            kcm = ktokcm(k)
            DO jcm = 1, ncmy
```

```
                j = jtojmap(jcm)
                jp = jtojmap(jcm+1)-1
                IF(j==1) jp=ncmy
                IF(j==ncmy) jp=1
                DO i = nxs(j), nxe(j)
                   icm = itoicm(i)
                   l = nodel(i,j)
                   lp = nodel(i,jp)
                   lcm = nodelcm(icm,jcm)
                   DO m = 1, ng
                      jncm(m,lcm,kcm)=jncm(m,lcm,kcm) + jn(m,l,k)
                      jscm(m,lcm,kcm)=jscm(m,lcm,kcm) + jss(m,lp,k)
                   END DO
                END DO
             END DO
          END DO
!r cm current
          DO k = nzs, nze
             kcm = ktokcm(k)
             DO j = 1, ny
                jcm = jtojcm(j)
                DO icm = ncmxs(jcm), ncmxe(jcm)
                   i = itoimap(icm)
                   ii = itoimap(icm+1)-1
                   IF(icm.EQ.ncmxe(jcm)) ii=nxe(1)
                   l = nodel(i,j)
                   le = nodel(ii,j)
                   lcm = nodelcm(icm,jcm)
                   DO m = 1, ng
                      jwcm(m,lcm,kcm)=jwcm(m,lcm,kcm) + jw(m,l,k)
                      jecm(m,lcm,kcm)=jecm(m,lcm,kcm) + je(m,le,k)
                   END DO
                END DO
             END DO
          END DO

!jwcm=zero
!jecm=zero
!jncm=zero
!jscm=zero
!jbcm=zero
!jtcm=zero

!define current correction coefficient
          dhatw=zero
          dhate=zero
          dhatn=zero
          dhats=zero
          dhatb=zero
          dhatt=zero
```

```
          dhatpn=zero
          dhatps=zero
          jpn=zero
          jps=zero
          DO k = ncmzs, ncmze
             kb = k-1
             kt = k+1
           hcmzk = hcmz(k)
           DO j = 1, ncmy
              IF(j==1) jp=ncmy       !BC
                IF(j==ncmy) jp=1          !BC
                hcmyj = hcmy(j)
                hcmyz = hcmyj * hcmzk
              DO i = ncmxs(j), ncmxe(j)
!                  IF(i == 1) THEN
!                    hcmyzL = centerxcm(i) * hcmyz
!                    hcmyzR = centerxcm(i) * hcmyz
!                  ELSE
                   hcmyzL = (centerxcm(i) - hcmx(i)/2.0_rvp) * hcmyz
                   hcmyzR = (centerxcm(i) + hcmx(i)/2.0_rvp) * hcmyz
!                  END IF
                   hcmxi = hcmx(i)
                   hcmxy = hcmxi * hcmyj
                   hcmxy = hcmxy * centerxcm(i)
                   hcmxz = hcmxi * hcmzk
                 l = nodelcm(i,j)
                   lw = nodelcm(i-1,j)
                   ln = nodelcm(i,j-1)
                 le = nodelcm(i+1,j)
                 ls = nodelcm(i,j+1)
                   IF(j==1 .OR. j==ncmy) lp=nodelcm(i,jp)
!                   IF(hcmyzL.EQ.0) hcmyzL=1
!                   IF(hcmyzR.EQ.0) hcmyzR=1
!                   IF(hcmxy.EQ.0)  hcmxy=1
!                   IF(hcmxz.EQ.0)  hcmxz=1
                 DO m=1, ng
                     IF(hcmyzL == zero) THEN
                       jwcm(m,l,k) = zero
                     ELSE
                       jwcm(m,l,k)=jwcm(m,l,k)/hcmyzL
                     END IF
                     jecm(m,l,k)=jecm(m,l,k)/hcmyzR
                     jncm(m,l,k)=jncm(m,l,k)/hcmxz
                     jscm(m,l,k)=jscm(m,l,k)/hcmxz
                     jbcm(m,l,k)=jbcm(m,l,k)/hcmxy
                     jtcm(m,l,k)=jtcm(m,l,k)/hcmxy
                     dhatw(m,l,k)=(-jwcm(m,l,k)-
(dfwcm(m,l,k)*(cmphicm(m,l,k)-cmphicm(m,lw,k))))/  &
                                    (cmphicm(m,l,k)+cmphicm(m,lw,k))
```

```
                  dhatn(m,l,k)=(-jncm(m,l,k)-
(dfncm(m,l,k)*(cmphicm(m,l,k)-cmphicm(m,ln,k))))/  &
                                (cmphicm(m,l,k)+cmphicm(m,ln,k))
                  dhatb(m,l,k)=(-jbcm(m,l,k)-
(dfbcm(m,l,k)*(cmphicm(m,l,k)-cmphicm(m,l,kb))))/  &
                                (cmphicm(m,l,k)+cmphicm(m,l,kb))
                  dhate(m,l,k)=(-jecm(m,l,k)-
(dfwcm(m,le,k)*(cmphicm(m,le,k)-cmphicm(m,l,k))))/ &
                                (cmphicm(m,le,k)+cmphicm(m,l,k))
                  dhats(m,l,k)=(-jscm(m,l,k)-
(dfncm(m,ls,k)*(cmphicm(m,ls,k)-cmphicm(m,l,k))))/ &
                                (cmphicm(m,ls,k)+cmphicm(m,l,k))
                  dhatt(m,l,k)=(-jtcm(m,l,k)-
(dfbcm(m,l,kt)*(cmphicm(m,l,kt)-cmphicm(m,l,k))))/ &
                                (cmphicm(m,l,kt)+cmphicm(m,l,k))
               IF(j==1) dhatpn(m,l,k)= &
                                (-jscm(m,lp,k)-
(dfnpcm(m,lp,k)*(cmphicm(m,l,k)-cmphicm(m,lp,k))))/  &
                                (cmphicm(m,l,k)+cmphicm(m,lp,k))
                  IF(j==ncmy) dhatps(m,l,k)= &
                                (-
jscm(m,lp,k)+(dfnpcm(m,lp,k)*(cmphicm(m,l,k)-cmphicm(m,lp,k))))/
&
                                (cmphicm(m,l,k)+cmphicm(m,lp,k))
               END DO
            END DO
         END DO
      END DO

!       dhatn=zero
!       dhats=zero
!       dhate=zero
!       dhatw=zero
!       dhatt=zero
!       dhatb=zero

!define matrix coefficients for unknown coarse mesh fluxes
!        IF(betapinit) THEN
!          cmbetapcm=one
!          betapinit=FALSE
!        END IF
        cmbetapcm=one !for ss the prompt neutron fraction is the
same

        DO k = ncmzs, ncmze
           kt = k+1
           hcmzk = hcmz(k)
           DO j = 1, ncmy
              hcmyj = hcmy(j)
              hcmyz = hcmyj * hcmzk
```

```
               IF(j==1) jp=ncmy
               IF(j==ncmy) jp=1
               DO i = ncmxs(j), ncmxe(j)
                  hcmyzL = (centerxcm(i) - hcmx(i)/2.0_rvp)*hcmyz
                  hcmyzR = (centerxcm(i) + hcmx(i)/2.0_rvp)*hcmyz
                  hcmxi = hcmx(i)
                  hcmxy = hcmxi * hcmyj
                  hcmxy = hcmxy * centerxcm(i)
                  hcmxz = hcmxi * hcmzk
                  l = nodelcm(i,j) !$$
                  le = nodelcm(i+1,j) !$$
                  ls = nodelcm(i,j+1) !$$
                  IF(j==1 .OR. j==ncmy) lp=nodelcm(i,jp)
                  DO m =1, ng
                     cmccw(m,l,k)=(-
dfwcm(m,l,k)+dhatw(m,l,k))*hcmyzL
                     cmccn(m,l,k)=(-
dfncm(m,l,k)+dhatn(m,l,k))*hcmxz
                     cmccb(m,l,k)=(-
dfbcm(m,l,k)+dhatb(m,l,k))*hcmxy
                     cmcce(m,l,k)=(-dfwcm(m,le,k)-
dhate(m,l,k))*hcmyzR
                     cmccs(m,l,k)=(-dfncm(m,ls,k)-
dhats(m,l,k))*hcmxz
                     cmcct(m,l,k)=(-dfbcm(m,l,kt)-
dhatt(m,l,k))*hcmxy
                     IF(j==1) cmccpn(m,l,k)=(-dfnpcm(m,lp,k)-
dhatpn(m,lp,k))*hcmxz
                     IF(j==ncmy) cmccps(m,l,k)=(-dfnpcm(m,lp,k)-
dhatpn(m,lp,k))*hcmxz
                     cmamcc(m,l,k)=
(dfwcm(m,l,k)+dhatw(m,l,k))*hcmyzL &

+(dfncm(m,l,k)+dhatn(m,l,k))*hcmxz &

+(dfbcm(m,l,k)+dhatb(m,l,k))*hcmxy &
                                    +(dfwcm(m,le,k)-
dhate(m,l,k))*hcmyzR &
                                    +(dfncm(m,ls,k)-
dhats(m,l,k))*hcmxz &
                                    +(dfbcm(m,l,kt)-
dhatt(m,l,k))*hcmxy
                     IF(j==1 .OR. j==ncmy)
cmamcc(m,l,k)=cmamcc(m,l,k)  &
                                    +(dfnpcm(m,lp,k)-
dhatpn(m,lp,k))*hcmxz

cmam(m,l,k)=cmamcc(m,l,k)+(cmxstfcm(m,l,k)+cmrvdeltcm(m,l,k))*volc
m(l,k)
```

```
                    cmafcm(m,l,k) =
cmbetapcm(l,k)*cmxsnffcm(m,l,k)*volcm(l,k)
                    cmpsicm(l,k) = cmpsicm(l,k) +
cmafcm(m,l,k)*cmphicm(m,l,k) !calculate fission source
                    DO m1 = 1, ng
                       cmscatvcm(m,m1,l,k) = cmxscatcm(m,m1,l,k) *
volcm(l,k)
                    END DO
                 END DO
              END DO
           END DO
        END DO

!zero out unnecessary boundary coupling coefficients
        kb = ncmzs
        kt = ncmze
        DO l = 1, ncmxy
           DO m = 1, ng
              cmccb(m,l,kb) = zero
              cmcct(m,l,kt) = zero
           END DO
        END DO
        DO k = ncmzs, ncmze
           DO i = ncmxs(1), ncmxe(1)
              ln = nodelcm(i,ncmys(i))
              ls = nodelcm(i,ncmye(i))
              DO m = 1, ng
                 cmccn(m,ln,k) = zero
                 cmccs(m,ls,k) = zero
              END DO
           END DO
           DO j = ncmys(1), ncmye(1)
              lw = nodelcm(ncmxs(j),j)
              le = nodelcm(ncmxe(j),j)
              DO m = 1, ng
                 cmccw(m,lw,k) = zero
                 cmcce(m,le,k) = zero
              END DO
           END DO
        END DO


END SUBROUTINE cmfdaccel2
```