UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

PhD THESIS

Development of a 3D Modal Neutron Code with the Finite Volume Method for the Diffusion and Discrete Ordinates Transport Equations. Application to Nuclear Safety Analyses.

September 2018

Author: Álvaro Bernal García

Supervisors: Dr. Gumersindo Jesús Verdú Martín
Dr. Rafael Miró Herrero

Abstract

The main objective of this thesis is the development of a Modal Method to solve two equations: the Neutron Diffusion Equation and the Discrete Ordinates Neutron Transport Equation. Moreover, this method uses the Finite Volume Method to discretize the spatial variables. The solution of these equations gives the neutron flux, which is related to the power produced in nuclear reactors; thus, the neutron flux is a paramount variable in Nuclear Safety Analyses. On the one hand, the use of Modal Methods is justified because one uses them to perform instability analyses in nuclear reactors. On the other hand, it is worth using the Finite Volume Method because one uses it to solve thermalhydraulic equations, which are strongly coupled with the energy generation in the nuclear fuel.

First, this thesis defines the equations mentioned above and the main methods to solve these equations. Furthermore, the thesis describes the major schemes and features of the Finite Volume Method. In addition, the author also introduces the major methods used in the Modal Method, which include the methods used to solve the eigenvalue problem, as well as those used to solve the time dependent Ordinary Differential Equations.

Next, the author develops several algorithms of the Finite Volume Method applied to the Steady State Neutron Diffusion Equation. In addition, the thesis includes an improvement of the multigroup formulation, which solves problems involving upscattering and fission terms in several energy groups. Moreover, the author optimizes the algorithms to do calculations with parallel computing.

The previous solution is used as initial condition to solve the time dependent Neutron Diffusion Equation. The author uses a Modal Method to do so, which transforms the Ordinary Differential Equations System into a smaller system that is solved by using the Exponential Matrix Method. Furthermore, the author developed a computationally efficient method to estimate the adjoint flux from the forward one, because the Modal Method uses the adjoint flux.

Additionally, the thesis also presents an algorithm to solve the eigenvalue problem of the Neutron Transport Equation. This algorithm uses the Discrete Ordinates formulation and the Finite Volume Method. In particular, the author uses two types of quadratures for the Discrete Ordinates and two interpolation schemes for the Finite Volume Method.

Finally, the author tested the developed methods in different types of nuclear reactors, including commercial ones. The author checks the accuracy of the values of the crucial variables in Nuclear Safety Analyses, which are the multiplication factor and the power distribution. Furthermore, the thesis includes a sensitivity analysis of several parameters, such as the mesh and numerical methods. In conclusion, excellent results are reported in both accuracy and computational cost.

Resumen

El principal objetivo de esta tesis es el desarrollo de un Método Modal para resolver dos ecuaciones: la Ecuación de la Difusión de Neutrones y la de las Ordenadas Discretas del Transporte de Neutrones. Además, este método está basado en el Método de Volúmenes Finitos para discretizar las variables espaciales. La solución de estas ecuaciones proporciona el flujo de neutrones, que está relacionado con la potencia que se produce en los reactores nucleares, por lo que es un factor fundamental para los Análisis de Seguridad Nuclear. Por una parte, la utilización del Método Modal está justificada para realizar análisis de inestabilidades en reactores. Por otra parte, el uso del Método de Volúmenes Finitos está justificado por la utilización de este método para resolver las ecuaciones termohidráulicas, que están fuertemente acopladas con la generación de energía en el combustible nuclear.

En primer lugar, esta tesis incluye la definición de estas ecuaciones y los principales métodos utilizados para resolverlas. Además, se introducen los principales esquemas y características del Método de Volúmenes Finitos. También se describen los principales métodos numéricos para el Método Modal, que incluye tanto la solución de problemas de autovalores como la solución de Ecuaciones Diferenciales Ordinarias dependientes del tiempo.

A continuación, se desarrollan varios algoritmos del Método de Volúmenes Finitos para el Estado Estacionario de la Ecuación de la Difusión de Neutrones. Se consigue desarrollar una formulación multigrupo, que permite resolver el problema de autovalores para cualquier número de grupos de energía, incluyendo términos de upscattering y de fisión en varios grupos de energía. Además, se desarrollan los algoritmos para realizar la computación en paralelo.

La solución anterior es la condición inicial para resolver la Ecuación de Difusión de Neutrones dependiente del tiempo. En esta tesis se utiliza un Método Modal, que transforma el Sistema de Ecuaciones Diferenciales Ordinarias en uno de mucho menor tamaño, que se resuelve con el Método de la Matriz Exponencial. Además, se ha desarrollado un método rápido para estimar el flujo adjunto a partir del directo, ya que se necesita en el Método Modal.

Por otra parte, se ha desarrollado un algoritmo que resuelve el problema de autovalores de la Ecuación del Transporte de Neutrones. Este algoritmo es para la formulación de Ordenadas Discretas y el Método de Volúmenes Finitos. En concreto, se han aplicado dos tipos de cuadraturas para las Ordenadas Discretas y dos esquemas de interpolación para el Método de Volúmenes Finitos.

Finalmente, se han aplicado estos métodos a diferentes tipos de reactores nucleares, incluyendo reactores comerciales. Se han evaluado los valores de la constante de multiplicación y de la potencia, ya que son las variables fundamentales en los Análisis de Seguridad Nuclear. Además, se ha realizado un análisis de sensibilidad de diferentes parámetros como la malla y métodos numéricos. En conclusión, se obtienen excelentes resultados, tanto en precisión como en coste computacional.

Resum

El principal objectiu d'esta tesi és el desenvolupament d'un Mètode Modal per a resoldre dos equacions: l'Equació de Difusió de Neutrons i la de les Ordenades Discretes del Transport de Neutrons. A més a més, este mètode està basat en el Mètode de Volums Finits per a discretitzar les variables espacials. La solució d'estes equacions proporcionen el flux de neutrons, que està relacionat amb la potència que es produïx en els reactors nuclears; per tant, el flux de neutrons és un factor fonamental en els Anàlisis de Seguretat Nuclear. Per una banda, la utilització del Mètode Modal està justificada per a realitzar anàlisis d'inestabilitats en reactors. Per altra banda, l'ús del Mètode de Volums Finits està justificat per l'ús d'este mètode per a resoldre les equacions termohidràuliques, que estan fortament acoblades amb la generació d'energia en el combustible nuclear.

En primer lloc, esta tesi inclou la definició d'estes equacions i els principals mètodes utilitzats per a resoldre-les. A més d'això, s'introduïxen els principals esquemes i característiques del Mètode de Volums Finits. Endemés, es descriuen els principals mètodes numèrics per al Mètode Modal, que inclou tant la solució del problema d'autovalors com la solució d'Equacions Diferencials Ordinàries dependents del temps.

A continuació, es desenvolupa diversos algoritmes del Mètode de Volums Finits per a l'Estat Estacionari de l'Equació de Difusió de Neutrons. Es conseguix desenvolupar una formulació multigrup, que permetre resoldre el problema d'autovalors per a qualsevol nombre de grups d'energia, incloent termes d' upscattering i de fissió en diversos grups d'energia. A més a més, es desenvolupen els algoritmes per a realitzar la computació en paral·lel.

La solució anterior és la condició inicial per a resoldre l'Equació de Difusió de Neutrons dependent del temps. En esta tesi s'utilitza un Mètode Modal, que transforma el Sistema d'Equacions Diferencials Ordinàries en un problema de menor tamany, que es resol amb el Mètode de la Matriu Exponencial. Endemés, s'ha desenvolupat un mètode ràpid per a estimar el flux adjunt a partir del directe, perquè es necessita en el Mètode Modal.

Per altra banda, s'ha desenvolupat un algoritme que resol el problema d'autovalors de l'Equació de Transport de Neutrons. Este algoritme és per a la formulació d'Ordenades Discretes i el Mètode de Volums Finits. En concret, s'han aplicat dos tipos de quadratures per a les Ordenades Discretes i dos esquemes d'interpolació per al Mètode de Volums Finits.

Finalment, s'han aplicat estos mètodes a diversos tipos de reactors nuclears, incloent reactors comercials. S'han avaluat els valor de la constat de multiplicació i de la potència, perquè són variables fonamentals en els Anàlisis de Seguretat Nuclear. Endemés, s'ha realitzat un anàlisi de sensibilitat de diversos paràmetres com la malla i mètodes numèrics. En conclusió, es conseguix obtenir excel·lents resultats, tant en precisió com en cost computacional.

Acknowledgments

First, I would like to thank the support and academic training of three persons. On the one hand, my supervisors, Rafael Miró and Gumersindo Verdú, who gave me an excellent training in Reactor Physics and numerical methods. On the other hand, José Román, because he trained me in parallel computing and solving eigenvalue problems. Without the knowledge in these areas, I would not have performed this thesis.

Second, I am grateful to the Spanish Ministry of Education, Culture and Sport, because of the grant University Professor Training Program. This grant funded the development of this thesis and two internships. Likewise, I would like to extend the acknowledgments to the supervisors of these two stays: Alain Hébert and Matthew Jessee, since they gave me the opportunity to work with them in École Polytechnique de Montréal and Oak Ridge National Laboratory.

Moreover, I also appreciated the academic training in Nuclear Engineering of all my professors of the Chemical and Nuclear Engineering Department, at the Universitat Politècnica de València.

On the other hand, I would like to show my gratitude to all my work partners because of the excellent work environment. Unfortunately, the list is so long that I cannot write everyone, but I would like to highlight Carles Mesado, Carlos Peña and Sergio Morató, since we had a fantastic time.

Finally, I would like to express my gratitude to all my friends of Valencia and Alicante, as they have always been close to me, both the good and bad moments. In addition, I appreciate all the love and support of my family. In particular, I would like to point out my parents and brother, because they gave me the best moments of my life and they have demonstrated me that they are always close to me when I need them.

Agradecimientos

En primer lugar, me gustaría agradecer el apoyo y formación de tres personas. Por una parte, mis directores de tesis, Rafael Miró y Gumersindo Verdú, que me dieron una excelente formación en Física de Reactores y métodos numéricos. Por otra parte, José Román, por su instrucción en la computación en paralelo y en el cálculo de autovalores. Sin el conocimiento de estos campos, no habría podido realizar esta tesis.

En segundo lugar, me gustaría agradecer al Ministerio de Educación, Cultura y Deporte la concesión del contrato predoctoral de Formación de Profesorado Universitario. Este contrato sirvió para financiar el desarrollo de esta tesis y para realizar dos estancias. Del mismo modo, me gustaría extender mis agradecimientos a los supervisores de mis estancias: Alain Hébert y Matthew Jessee, que me acogieron en la École Polytechnique de Montréal y Oak Ridge National Laboratory, respectivamente.

Además, me gustaría agradecer también la formación en Ingeniería Nuclear a todos los profesores del Departamento de Ingeniería Química y Nuclear de la Universitat Politècnica de València.

Por otra parte, quiero agradecer a todos mis compañeros el excelente ambiente de trabajo, que han hecho que esta experiencia haya sido tan agradable. Desgraciadamente, la lista es muy larga y no puedo escribir el nombre de todos, pero me gustaría hacer una mención especial a Carles Mesado, Carlos Peña y Sergio Morató, por los buenos momentos que hemos pasado.

Para finalizar, me gustaría agradecer el apoyo de mis amigos de Valencia y Alicante, que han estado a mi lado, tanto en los momentos buenos como malos. Finalmente, agradezco todo el cariño y apoyo de mi familia. En particular, querría resaltar a mis padres y mi hermano, pues me han dado los mejores momentos de mi vida y me han demostrado que siempre están cuando los necesito.

Contents

List of Symbols

Acronyms

ADF	Assembly Discontinuity Factor
ANM	Analytic Nodal Method
BiCG	Bi-Conjugate Gradient
BiCGSTAB	Bi-Conjugate Gradient stabilized
BWR	Boiling Water Reactor
CGS	Conjugate Gradients Squared
CMFD	Coarse Mesh Finite Difference Method
CPU	Central Processing Unit
DF	Discontinuity Factor
FDM	Finite Difference Method
FEM	Finite Element Method
FVM	Finite Volume Method
GCR	Generalized Conjugate Residual
GMRES	Generalized Minimal Residual
LWR	Light Water Reactor
MPI	Message Passing Interface
NCM	Nodal Collocation Method
NEM	Nodal Expansion Method
NM	Nodal Method
PETSc	Portable, Extensible Toolkit for Scientific Computation
PWR	Pressurized Water Reactor
SLEPc	Scalable Library for Eigenvalue Problem Computations
VVER	Water Water Energetic Reactor

Symbols

E	Energy variable
$\vec{r}$	Spatial variable
t	Time variable
v	Velocity of neutrons
$\vec{\Omega}$	Direction variable
θ	Polar angle
φ	Azimuthal angle
ϕ	Scalar neutron flux
ψ	Angular neutron flux
$\vec{J}$	Neutron current
$\vec{J}_{in}$	Incoming neutron current
$\vec{J}_{out}$	Outgoing neutron current
N	Neutron density
Σ_t	Total macroscopic cross section
Σ_a	Absorption macroscopic cross section
$\Sigma_s(E' \rightarrow E)$	Scattering macroscopic cross section from energy E' to energy E
$\nu\Sigma_f$	Nu-fission macroscopic cross section
β_k	Yield of delayed neutrons of the precursors group k
β	Total yield of delayed neutrons
χ	Fission spectrum of prompt neutrons
χ_k^{del}	Fission spectrum of delayed neutrons of the precursors group k
λ_k	Decay constant of the precursors group k
C_k	Concentration of precursors of the group k
K	Number of groups of precursors
G	Number of energy groups
v_g	Velocity of neutrons for the energy group g
ψ_g	Angular neutron flux for the energy group g
ϕ_g	Scalar neutron flux for the energy group g
$\vec{J}_g$	Neutron current for the energy group g
$\Sigma_{t,g}$	Total macroscopic cross section for the energy group g
$\Sigma_{a,g}$	Absorption macroscopic cross section for the energy group g
$\Sigma_{s,g' \rightarrow g}$	Scattering macroscopic cross section from energy g' to energy g
$\nu\Sigma_{f,g}$	Nu-fission macroscopic cross section for the energy group g
χ_g	Fission spectrum of prompt neutrons for the energy group g
$\chi_{g,k}^{del}$	Fission spectrum of delayed neutrons of the precursors group k and for the energy group g

α	Albedo
$\mathbf{k}$	Eigenvalue
Y_l^m	Spherical Harmonics of orders l and m
P_l	Legendre polynomial of order l
P_l^m	Associated Legendre polynomial of orders l and m
L	Order of the expansion with Legendre polynomials
N_c	Number of cells
N_f	Number of inner faces
N_b	Number of boundary faces
N_d	Number of directions
n_f	Number of faces of each cell
$\mathcal{C}$	Condition number
$\phi_{g,i}$	Cell averaged value of the scalar neutron flux for the energy group g and cell i
$J_{g,i,j}$	Face averaged value of the neutron current for the energy group g, cell i and face j, with outgoing direction from cell i
$\Sigma_{a,g}^i$	Absorption macroscopic cross section for the energy group g and cell i
$\Sigma_{s,g'\to g}^i$	Scattering macroscopic cross section from energy g' to energy g and cell i
$\nu\Sigma_{f,g}^i$	Nu-fission macroscopic cross section for the energy group g and cell i
χ_g^i	Fission spectrum of prompt neutrons for the energy group g and cell i
$\alpha_{g,j}$	Albedo for the energy group g and face j
$k_{i,j}$	Kernel used to calculate the neutron flux at face j, which is calculated by multiplying this kernel by the neutron flux of cell i
$k_{i,j}^{grad}$	Kernel used to calculate the gradient of the neutron flux at face j. The gradient is calculated by multiplying this kernel by the neutron flux of cell i
$p_t(x,y,z)$	Polynomial term t of the expansion of the neutron flux
$a_{g,i,t}$	Coefficient of the expansion of the neutron flux for the energy group g, cell i and polynomial term $p_t(x,y,z)$
$\overline{p}_t^{V_i}$	Cell averaged value of the polynomial term $p_t(x,y,z)$ in cell i
$\overline{p}_t^{S_{i,j}}$	Face averaged value of the polynomial term $p_t(x,y,z)$ in cell i and face j
$u_{i,j,x}$	X-Direction cosine of the normal of face j, in the outgoing direction of cell i
$u_{i,j,y}$	Y-Direction cosine of the normal of face j, in the outgoing direction of cell i

$u_{i,j,z}$	Z-Direction cosine of the normal of face j, in the outgoing direction of cell i
$\overline{\vec{\nabla} p_t}^{S_{i,j}}$	Face averaged value of the gradient of the polynomial term $p_t(x,y,z)$ in cell i and face j
v_g^i	Velocity of neutrons for the energy group g and cell i
$\chi_{g,k}^{del,i}$	Fission spectrum of delayed neutrons of the precursors group k and for the energy group g and cell i
C_k^i	Cell averaged value of the concentration of precursors of the group k in cell i
$r_{i,j}$	Distance from the centroid of cell i to the centroid of face j
r_{i_1,i_2}	Distance from the centroid of cell i_1 to the centroid of cell i_2
P_i	Power in the cell i
MP	Mean Power
PE_i	Power Error of P_i
EE_i	Eigenvalue Error of $\mathbf{k_i}$
MPE	Mean Power Error

Chapter 1

Introduction

1.1 Motivation and objectives

One of the most important variables in Nuclear Safety Analyses is the power generated inside nuclear reactor cores. This power comes from the energy released in each fission of the nuclear fuel, which is proportional to the neutron flux. Therefore, one can determine the spatial and time distribution of the power by calculating the distribution of the neutron flux.

The most accurate way of calculating the neutron population is by solving the Neutron Transport Equation. Although the explanation of this equation is not the goal of this section, the author would like to highlight some features of this equation to justify the development of this thesis. First, it is an integral-differential equation depending on several variables concerning the space, time, neutron energy and neutron direction. Second, the coefficients of this equation are strongly dependent on boron, control rods, burnup and thermal-hydraulic variables, such as the temperature and density. The full explanation of this equation is given in Chapter 2.

The solution of the Neutron Transport Equation in commercial nuclear reactor cores is not straightforward, basically because of two reasons. First, these cores are complex systems containing different components composed of different

materials and geometries. Second, the coefficients of the equation change over time, since the temperature and density distribution might change during the reactor operation. Consequently, one has to use numerical methods.

There is a large number of numerical methods to solve the Neutron Transport Equation, but computationally efficient methods are of great interest in Nuclear Safety Analyses. There is a faster way to calculate the neutron population in nuclear reactor cores by solving the Neutron Diffusion Equation. This equation is a simplification of the Neutron Transport Equation, and consequently its use is limited to certain conditions and is less accurate. Nevertheless, the Neutron Diffusion Equations is sufficiently accurate and efficient to be applied in Nuclear Safety Analyses. Therefore, it is worth developing and optimizing algorithms to solve both the Neutron Transport and Diffusion Equations.

Among the large number of numerical methods, the author of this thesis chose the Modal Method for the time discretization because of two reasons. First, this method can be used for modal and instability analyses of nuclear reactors, which are particularly interesting in Boiling Water Reactors. Second, this method might be fast for solving the time variables. As regards the spatial discretization, the Finite Volume Method was chosen due to three reasons. First, it can be easily applied to unstructured meshes, which can model any kind of geometry. Second, it is typically used in transport equations due to the conservation of the transported quantity. Third, it is the method most commonly used in thermal-hydraulics and the Neutron Transport and Diffusion Equations are strongly dependent on the thermal-hydraulic variables. Thereby, one could use the same method and meshes to solve this coupled problem.

All in all, the thesis objectives can be summarized as follows.

1. To develop a Finite Volume Method for the Steady State Neutron Diffusion Equation.
2. To develop a Modal Method for the Time dependent Neutron Diffusion Equation.
3. To develop a Finite Volume Method for the Steady State Neutron Transport Equation.
4. To optimize the algorithms for calculating large and complex problems in parallel computers with the state of the art methods.

1.2 Thesis outline

The thesis is organized in seven chapters. Next chapter introduces the state of the art concerning the main subjects of this thesis, which are classified in six sections. Section 2.1 explains the Neutron Diffusion Equation and highlights the major methods and codes used for solving it. Section 2.2 gives an explanation of the Neutron Transport Equation, summarizes the major methods applied to this equation and points out the most used codes to solve this equation. Section 2.3 gives a short introduction about geometry discretization. In particular, this section presents the typical geometry of commercial nuclear reactors and the different meshes to discretize it. Section 2.4 defines the Finite Volume Method and gives some background information of different schemes that are commonly used. Section 2.5 points out different methods for solving eigenvalue problems and linear systems. Moreover, this section summarizes the state of the art libraries containing these methods. The goal of this section is to introduce the main tools used for calculating the Steady State of both Neutron Diffusion and Transport Equations. Finally, Section 2.6 presents the major methods used for solving time dependent Ordinary Differential Equations. In addition, this section highlights several libraries containing these methods.

Chapter 3 develops the first objective of this thesis. This chapter is organized in five sections. In Section 3.1, the author applies the Finite Volume Method to the 2-energy group Neutron Diffusion Equation and defines its eigenvalue problem. Section 3.2 explains three methods for calculating the neutron currents. Section 3.3 extends the application of the Finite Volume Method to any multigroup formulation. Section 3.4 explains the algorithm for solving the eigenvalue problem. Section 3.5 explains the parallelization of the method, for fulfilling the fourth objective of this thesis.

Chapter 4 accomplishes the second objective of the thesis. It is divided in three sections. Section 4.1 explains the Modal Method used in this thesis for solving the time dependent Neutron Diffusion Equation. Section 4.2 describes an easy and fast method for estimating the adjoint flux from the forward one, because the adjoint flux is needed in the Modal Method. Finally, Section 4.3 presents a method for updating the modes, because these modes might change over time.

Chapter 5 achieves the third objective of the thesis. It is classified in three sections. Section 5.1 defines the Discrete Ordinates formulation of the Neutron Transport Equation with the Finite Volume Method. Section 5.2 summarizes several quadratures used for the Discrete Ordinates. Section 5.3 defines two in-

terpolation schemes for calculating the leakage terms of the Neutron Transport Equation with the Finite Volume Method.

Chapter 6 shows several benchmarks and analyses of the results for the different methods developed in this thesis. These benchmarks includes mock-up and commercial nuclear reactors. Finally, Chapter 7 not only summarizes the major conclusions, but also highlights the future work.

Chapter 2

State of the art

2.1 Neutron Diffusion Equation

Many authors have highlighted the importance of the Neutron Diffusion Equation for obtaining the neutron distribution in full-core calculations. This equation is accurate enough to provide a quantitative understanding of many physics features of nuclear reactors, as discussed in Stacey 2007. Stacey also states that the Neutron Diffusion Equation is the simplest and most widely used mathematical description for obtaining the neutron distribution in nuclear reactors, and consequently it is the workhorse computational method of Nuclear Reactor Physics. It is true that the Neutron Diffusion Equation was used for the design of most of the early reactors, as explained in Lamarsh and Baratta 2001. Now more sophisticated methods have been developed, but Lamarsh and Baratta also assert that the Neutron Diffusion Equation is still widely used to provide first estimates of reactor properties. Furthermore, the Neutron Diffusion Equation is widely used for core follow and monitoring commercial Light Water Reactors (LWRs).

One can obtain the Neutron Diffusion Equation from the Neutron Transport Equation, or from a Neutron Balance Equation, applied over a control domain, as discussed in Lamarsh and Baratta 2001, Cacuci 2010 or Hébert 2009. In this balance equation, one calculates the rate of change in number of neutrons

as the rate of production minus the rate of loss of neutrons. The rate of neutron production includes the rate of fission and scattering interactions and the decay of neutron precursors; the rate of neutron loss involves the rate of total interactions and the leakage of the control domain. Equation 2.1 shows the mathematical expression of this balance.

$$\begin{aligned}\frac{1}{v(E)}\frac{d\phi(E,\vec{r},t)}{dt} &= -\nabla\vec{J}(E,\vec{r},t) - \Sigma_t(E,\vec{r},t)\phi(E,\vec{r},t) + \\ &+ \int_0^\infty \Sigma_s(E'\rightarrow E,\vec{r},t)\phi(E',\vec{r},t)dE' + \\ &+ (1-\beta)\chi(E,\vec{r},t)\int_0^\infty \nu\Sigma_f(E',\vec{r},t)\phi(E',\vec{r},t)dE' + \\ &+ \sum_{k=1}^{K}\chi_k^{del}(E,\vec{r},t)\lambda_k C_k(\vec{r},t) \end{aligned} \tag{2.1}$$

In this equation, E is the energy variable, $\vec{r}$ is the spatial variable, t is the time variable, v is the velocity of neutrons, ϕ is the scalar neutron flux, $\vec{J}$ is the neutron current, Σ_t is the total macroscopic cross section, $\Sigma_s(E'\rightarrow E)$ is the scattering macroscopic cross section from the energy E' to E, β is the total yield of delayed neutrons, χ is the fission spectrum of prompt neutrons, $\nu\Sigma_f$ is the nu-fission macroscopic cross section, ν is the average number of neutrons produced in each fission, χ_k^{del} is the fission spectrum of the delayed neutrons of the precursors group k, λ_k is the decay constant of the precursors group k, C_k is the concentration of precursors of the group k, K is the total number of neutron precursors groups.

The scalar neutron flux is defined as $\phi(E,\vec{r},t) = v(E,\vec{r},t)N(E,\vec{r},t)$, where N is the neutron density, that is, the number of neutrons in a differential volume; thus, the units of N are cm^{-3} and those of ϕ are $cm^2\cdot s^{-1}$, since the units of v are $cm\cdot s^{-1}$. Therefore, one can use ϕ or N to define the neutron distribution; in Reactor Physics, one typically uses ϕ because one calculates the interaction rates of the neutrons as the product of ϕ and the macroscopic cross section corresponding to that interaction. The macroscopic cross section of certain type of interaction can be defined as the probability of that interaction to take place; its units are cm^{-1}. Further mathematical definitions of the macroscopic cross sections are beyond the scope of this thesis, yet one can look for them in the literature for Reactor Physics, for example in Stacey 2007, Hébert 2009, Lamarsh and Baratta 2001 or Cacuci 2010. As regards neutron interactions, the most important ones in Reactor Physics are

the total interaction, scattering interaction from E to E', fission interaction and absorption interaction. The total interaction can be calculated as the sum of the absorption and the scattering interaction, so one can calculate the total cross section as in Equation 2.2, in which Σ_a is the absorption macroscopic cross section. On the other hand, the units of C_k and λ_k are cm^{-3} and s^{-1} respectively. There are three non dimensional variables in Equation 2.1: β, χ and χ_k^{del}. The value of β might range from 0 to 1, but it is usually much lower than 1; χ and χ_k^{del} are probabilities of neutrons to be produced with certain energy.

$$\Sigma_t(E, \vec{r}, t) = \Sigma_a(E, \vec{r}, t) + \int_0^\infty \Sigma_s(E \to E', \vec{r}, t) dE' \tag{2.2}$$

It is worth mentioning two things related with C_k and λ_k. First, λ_k is the probability of decay per unit time of the precursor of group k. Second, $\lambda_k C_k$ is the rate of neutrons per volume produced by the decay of the precursors of group k. Moreover, the concentration of precursors, C_k, changes over time, which can be calculated with Equation 2.3, in which β_k is the yield of delayed neutrons of the precursors group k.

$$\frac{dC_k(\vec{r}, t)}{dt} = \beta_k \int_0^\infty \nu\Sigma_f(E', \vec{r}, t)\phi(E', \vec{r}, t)dE' - \lambda_k C_k(\vec{r}, t), \qquad k = 1, ..., K \tag{2.3}$$

The energy dependence of the previous variables complicates the solution of the Neutron Diffusion Equation, because the energy is a continuous variable and the cross sections depend continuously on the energy. For this reason, one typically uses a multigroup formulation, which discretizes the energy variable in a set of groups of energy. In this formulation, one habitually numbers the energy groups from the highest to the lowest energy. For example, for each energy group g, the energy ranges from E_g to E_{g-1}. The goal of this method is to obtain energy-averaged values in each group, which are expressed with the sub-index g. Equations 2.4-2.11 show these energy-averaged values, for certain function $f(E)$. However, one does not know $f(E)$, so one has to guess this function to obtain accurate multigroup cross sections. For example, the following function $f_g(E) = \phi(E)/\phi_g$ would be a great function for the energy group g, if $\phi(E)$ is a good guess of the real scalar neutron flux, because it would conserve the interaction rate, as shown in Equation 2.12 for the absorption interaction.

$$\phi_g(\vec{r},t) = \int_{E_g}^{E_{g-1}} \phi(E,\vec{r},t)dE \tag{2.4}$$

$$\vec{J}_g(\vec{r},t) = \int_{E_g}^{E_{g-1}} \vec{J}(E,\vec{r},t)dE \tag{2.5}$$

$$\frac{1}{v_g} = \int_{E_g}^{E_{g-1}} \frac{1}{v(E)} f(E)dE \tag{2.6}$$

$$\Sigma_{a,g}(\vec{r},t) = \int_{E_g}^{E_{g-1}} \Sigma_a(E,\vec{r},t)f(E)dE \tag{2.7}$$

$$\Sigma_{s,g\to g'}(\vec{r},t) = \int_{E_{g'}}^{E_{g'-1}} dE' \int_{E_g}^{E_{g-1}} \Sigma_s(E\to E',\vec{r},t)f(E)dE \tag{2.8}$$

$$\nu\Sigma_{f,g}(\vec{r},t) = \int_{E_g}^{E_{g-1}} \nu\Sigma_f(E,\vec{r},t)f(E)dE \tag{2.9}$$

$$\chi_g(\vec{r},t) = \int_{E_g}^{E_{g-1}} \chi(E,\vec{r},t)dE \tag{2.10}$$

$$\chi_{g,k}^{del}(\vec{r},t) = \int_{E_g}^{E_{g-1}} \chi_k^{del}(E,\vec{r},t)dE \tag{2.11}$$

$$\Sigma_{a,g}(\vec{r},t) = \frac{\int_{E_g}^{E_{g-1}} \Sigma_a(E,\vec{r},t)\phi(E,\vec{r},t)dE}{\phi_g(\vec{r},t)} \tag{2.12}$$

If ones applies the multigroup formulation of G energy groups to Equations 2.1-2.3, one obtains Equations 2.13 and 2.14, for each energy group g.

$$\begin{aligned}\frac{1}{v_g}\frac{d\phi_g(\vec{r},t)}{dt} &= -\nabla\vec{J}_g(\vec{r},t) - \left(\Sigma_{a,g}(\vec{r},t) + \sum_{\substack{g'=1\\g'\neq g}}^{G}\Sigma_{s,g\to g'}(\vec{r},t)\right)\phi_g(\vec{r},t) + \\ &+ \sum_{\substack{g'=1\\g'\neq g}}^{G}\Sigma_{s,g'\to g}(\vec{r},t)\phi_{g'}(\vec{r},t) + \\ &+(1-\beta)\chi_g(\vec{r},t)\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t)\phi_{g'}(\vec{r},t) + \\ &+ \sum_{k=1}^{K}\chi_{g,k}^{del}(\vec{r},t)\lambda_k C_k(\vec{r},t) \qquad (2.13)\end{aligned}$$

$$\frac{dC_k(\vec{r},t)}{dt} = \beta_k\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t)\phi_{g'}(\vec{r},t) - \lambda_k C_k(\vec{r},t), \qquad k=1,...,K \qquad (2.14)$$

In Neutron Diffusion Theory, one calculates the neutron current from the neutron flux. To do so, one uses Fick's Law. This law states there is a flow of neutrons from regions with higher neutron flux to regions with less neutron flux. Thus, one can calculate the neutron current as a proportional value of the gradient of the neutron flux, as shown in Equation 2.15. In this equation, the proportional constant D_g is called diffusion coefficient and has units of cm.

$$\vec{J}_g(\vec{r},t) = -D_g(\vec{r},t)\vec{\nabla}\phi_g(\vec{r},t) \qquad (2.15)$$

The Neutron Diffusion Equation provides valid results of the neutron flux under three assumptions, as discussed in Stacey 2007. First, the absorption is much less likely than the scattering. Second, the flux is assumed to be sufficiently slowly varying in space that it can be approximated by a linear spatial variation. Third, neutrons are scattered isotropically. The first condition is satisfied for most of the moderating and structural materials found in a nuclear reactor, but not for the fuel and control elements. The second condition is satisfied a few mean free paths away from the boundary of large (relative to the mean free path) homogeneous media with relatively uniform source distributions. The third condition is satisfied for scattering from heavy atomic mass nuclei. As nuclear reactors are highly heterogeneous, the Neutron Diffusion Equation might not provide an accurate solution of the neutron flux. However, one can

replace these highly heterogeneous materials by homogenized materials with averaged values of cross sections and diffusion coefficients, which reproduce the transport interaction rates, providing an accurate solution of the neutron flux.

The most accurate solution of the Neutron Diffusion Equation is the analytical one, which one can calculate for homogeneous reactors with simple geometries. Unfortunately, commercial nuclear reactors like Light Water Reactors (LWRs) are composed of different materials and geometries, as mentioned in the previous paragraph. Consequently, one has to discretize the geometry and use numerical methods to solve the Neutron Diffusion Equation.

The discretization of the geometry is explained in Section 2.3, but it is important to highlight the effects of this discretization on the Neutron Diffusion Equation. Basically, the geometry is discretized in different regions, in which one applies the Neutron Diffusion Equation, obtaining ϕ and $\vec{J}$ in each region. These variables might be discontinuous in the interface of two adjacent cells, which should not be. Therefore, one has to make sure that the continuity of ϕ and $\vec{J}$ is preserved in each interface. Equations 2.16 and 2.17 show these continuity conditions, for two adjacent regions V_1 and V_2 and the interface of them $\vec{r}_{V_1 \cap V_2}$.

$$\phi(\vec{r}_{V_1 \cap V_2} \in V_1) = \phi(\vec{r}_{V_1 \cap V_2} \in V_2) \tag{2.16}$$

$$\vec{J}(\vec{r}_{V_1 \cap V_2} \in V_1) = \vec{J}(\vec{r}_{V_1 \cap V_2} \in V_2) \tag{2.17}$$

Moreover, one needs boundary conditions to solve the Neutron Diffusion Equation. The boundary conditions most used are zero flux, reflective, zero incoming current and albedo (α), as discussed in Hébert 2009. Equations 2.18 and 2.19 show the zero flux and reflective conditions, in which $\vec{r}_b$ is the region of the boundary. With respect to the albedo condition, one can use two equations depending on the type of reactor. For Pressurized Water Reactors (PWRs) and Boiling Water Reactors (BWRs), one uses Equation 2.20; in this equation, $\vec{J}_{in}$ is the incoming neutron current and $\vec{J}_{out}$ is the outgoing neutron current, which are defined in the same equation. In addition, one can use Equation 2.20 with $\alpha = 0$ to define the boundary condition of zero incoming current. On the other hand, one uses Equation 2.21 for Water Water Energetic Reactors (VVERs).

$$\phi(\vec{r}_b) = 0 \tag{2.18}$$

$$\vec{\nabla}\phi(\vec{r}_b) = 0 \rightarrow \vec{J}(\vec{r}_b) = 0 \tag{2.19}$$

$$\alpha = \frac{\vec{J}_{in}(\vec{r}_b)}{\vec{J}_{out}(\vec{r}_b)} = \frac{\frac{1}{4}\phi(\vec{r}_b) - \frac{1}{2}\vec{J}(\vec{r}_b)}{\frac{1}{4}\phi(\vec{r}_b) + \frac{1}{2}\vec{J}(\vec{r}_b)} \rightarrow \vec{J}(\vec{r}_b) - \frac{1-\alpha}{2(1+\alpha)}\phi(\vec{r}_b) = 0 \tag{2.20}$$

$$\alpha = \frac{\vec{J}(\vec{r}_b)}{\phi(\vec{r}_b)} \rightarrow \vec{J}(\vec{r}_b) - \alpha\phi(\vec{r}_b) = 0 \tag{2.21}$$

With respect to the numerical methods, one should apply them for the spatial and time variables. First, one has to find the steady state solution, because this will be the initial condition of the transient calculation. This steady state is calculated by setting the time derivatives to 0. Nonetheless, this steady state is only accomplished for a specific condition of the geometry and materials (cross sections), which are not known a priori. As a consequence, one has to transform Equations 2.13 and 2.14 into an eigenvalue problem, as that of Equation 2.22, where $\mathbf{k}$ is the eigenvalue. If $\chi_g = \chi^{del}_{g,k}$, Equation 2.22 is simplified into Equation 2.23, because $\beta = \sum_{k=1}^{K}\beta_k$.

$$\begin{aligned} 0 \;=\; & -\nabla\vec{J}_g(\vec{r}) - \left(\Sigma_{a,g}(\vec{r}) + \sum_{\substack{g'=1 \\ g'\neq g}}^{G}\Sigma_{s,g\rightarrow g'}(\vec{r})\right)\phi_g(\vec{r}) + \\ & + \sum_{\substack{g'=1 \\ g'\neq g}}^{G}\Sigma_{s,g'\rightarrow g}(\vec{r})\phi_{g'}(\vec{r}) + \\ & + \frac{1}{\mathbf{k}}\left((1-\beta)\chi_g(\vec{r}) + \sum_{k=1}^{K}\chi^{del}_{g,k}(\vec{r})\beta_k\right)\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r})\phi_{g'}(\vec{r}) \end{aligned} \tag{2.22}$$

$$0 = -\nabla \vec{J}_g(\vec{r}) - \left(\Sigma_{a,g}(\vec{r}) + \sum_{\substack{g'=1 \\ g' \neq g}}^{G} \Sigma_{s,g \to g'}(\vec{r}) \right) \phi_g(\vec{r}) + \\ + \sum_{\substack{g'=1 \\ g' \neq g}}^{G} \Sigma_{s,g' \to g}(\vec{r}) \phi_{g'}(\vec{r}) + \frac{1}{\mathbf{k}} \chi_g(\vec{r}) \sum_{g'=1}^{G} \nu \Sigma_{f,g'}(\vec{r}) \phi_{g'}(\vec{r}) \quad (2.23)$$

The solution of the eigenvalue problem of Equation 2.23, using Fick's Law defined in Equation 2.15, gives the spatial distribution of the neutron flux. Actually, one could also include Fick's Law in Equation 2.23, obtaining Equation 2.24. However, these equations also contain spatial derivatives terms. Therefore, one has to apply numerical methods to these terms too. The methods most used for solving these equations are Nodal Methods (NMs), Finite Difference Methods (FDMs) and Finite Element Methods (FEMs), as explained in Hébert 2009.

$$0 = -\nabla \left(-D_g(\vec{r}) \vec{\nabla} \phi_g(\vec{r}) \right) - \left(\Sigma_{a,g}(\vec{r}) + \sum_{\substack{g'=1 \\ g' \neq g}}^{G} \Sigma_{s,g \to g'}(\vec{r}) \right) \phi_g(\vec{r}) + \\ + \sum_{\substack{g'=1 \\ g' \neq g}}^{G} \Sigma_{s,g' \to g}(\vec{r}) \phi_{g'}(\vec{r}) + \frac{1}{\mathbf{k}} \chi_g(\vec{r}) \sum_{g'=1}^{G} \nu \Sigma_{f,g'}(\vec{r}) \phi_{g'}(\vec{r}) \quad (2.24)$$

FDMs are easily applied, but they have two major drawbacks. First, FDMs might produce large matrices to obtain accurate results. Second, the application of these methods is limited to certain type of discretizations, particularly to structured meshes.

NMs use high order schemes to give accurate results in coarse meshes, by using for example Coarse Mesh Finite Difference (CMFD) methods. Examples of NMs are the Analytic Nodal Method (ANM), developed in Smith 1979, the Nodal Expansion Method (NEM), developed in Bennewitz, Finnemann, and Moldaschl 1975, and the Nodal Collocation Method (NCM), developed in Hébert 1987. On the one hand, ANM transforms the 3D Neutron Diffusion Equation into three 1D Neutron Diffusion Equations along each of the three directions, which one can solve analytically for certain approximations of the

leakages terms of these 1D equations. Likewise, NEM also transforms the 3D Neutron Diffusion Equation into three 1D equations, but one expands the neutron flux in each direction with 1D polynomial functions (typically quartic functions) to determine the leakage terms of the 1D equations. Although the analytic solution is more accurate than a polynomial expansion, NEM has two main advantages in comparison with ANM. First, it can be used in general multigroup formulations of the Neutron Diffusion Equation, whereas ANM is only applied to the 2 energy group formulation. Second, NEM can be applied to hexagonal geometries, while ANM is limited to Cartesian geometries. On the other hand, NCM expands $\phi(x, y, z)$ with product of Legendre Polynomials in each direction, that is, $\phi(x, y, z) = \sum_{k_1} \sum_{k_2} \sum_{k_3} P_{k_1}(x) P_{k_2}(y) P_{k_3}(z)$; then, the method obtains different equations by multiplying the Neutron Diffusion Equation by each $P_{k_1}(x) P_{k_2}(y) P_{k_3}(z)$. The main advantage of NCM is that it gives excellent results in coarse meshes. However, the main drawback is that it can be only applied in Cartesian meshes. One can find the application of the NCM for the Neutron Diffusion Equation in Verdú et al. 1994, Ginestar et al. 1998 and Miró et al. 2002.

With respect to FEMs, one can use them in any kind of discretization, but they might produce large matrices and solve the weak formulation of the Neutron Diffusion Equation. A large number of FEMs have been applied to the Neutron Diffusion Equation, for example those shown in Hébert 1993, Hébert 2008 and Vidal-Ferrandiz et al. 2014.

There is a large number of codes for solving the Neutron Diffusion Equation by means of different methods, based on FDMs, NMs and FEMs. Some examples are: SIMULATE-3 (DiGiovine et al. 1995), PARCS (Downar et al. 2006), TRIVAC (Hébert and Sekki 2010), VALKIN (Verdú et al. 1994, Miró et al. 2002), DIFF-3D (Derstine 2011), NESTLE (Turinsky et al. 1994), COBAYA (Aragonés and Ahnert 1986, Aragonés, Ahnert, and García-Herranz 2007), SKETCH-N (Zimin 2002) and DYN3D (Grundmann et al. 2005).

2.2 Neutron Transport Equation

Neutron transport is the process in which neutrons propagate through the atoms in a physical system. This includes the streaming of neutrons from one collision site to the next, the loss of neutrons by scattering and absorption interactions, and the production of neutrons by scattering and fission interactions. The Neutron Transport Equation is an accurate mathematical formulation for describing this neutron process and obtaining the neutron population in closed

domains. The derivation of this equation is based on the principle of neutron conservation, that is, a balance equation.

One should use seven independent variables to characterize a general 3D neutron transport process, wich are: three variables for the space domain ($\vec{r} = x\vec{i} + y\vec{j} + z\vec{k}$), two variables for the direction domain ($\vec{\Omega} = \vec{\Omega}(\theta, \varphi)$), one variable for the energy domain (E) and one variable for the time domain (t). The direction variable is defined by three direction cosines, as one can see in Figure 2.1 and Equation 2.25. Nonetheless, $\vec{\Omega}$ only has two independent variables, because of Equation 2.26. In addition, one can define these two variables with the polar (θ) and azimuthal (φ) angles shown in Figure 2.1. These seven variables enable one to specify the neutron density $N(\vec{\Omega}, E, \vec{r}, t)$ at a certain space position in the system $\vec{r}$, traveling in a certain direction $\vec{\Omega}$, with an energy E, and at a time t. It is worth using the neutron flux instead of N, as justified in Section 2.1. The neutron flux considering the direction variables is defined in Equation 2.27, it is called angular neutron flux and has the same units as the scalar neutron flux.

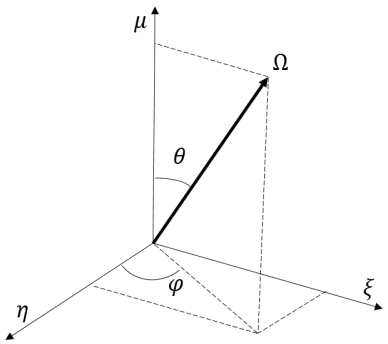


Figure 2.1: Direction variable

$$\vec{\Omega} = \mu\vec{i} + \eta\vec{j} + \xi\vec{k} \tag{2.25}$$

$$\mu^2 + \eta^2 + \xi^2 = 1 \tag{2.26}$$

$$\psi(\vec{\Omega}, E, \vec{r}, t) = v(E) \cdot N(\vec{\Omega}, E, \vec{r}, t) \tag{2.27}$$

The derivation of the Neutron Transport Equation is beyond the scope of this thesis, but one can look for this derivation in Cacuci 2010 or Hébert 2009. Equation 2.28 shows the Neutron Transport Equation. This equation has similar terms to those of Equation 2.1. Nevertheless, in Equation 2.28, ψ is the angular neutron flux, $\Sigma_s(\vec{\Omega}' \rightarrow \vec{\Omega}, E' \rightarrow E)$ is the double differential scattering cross section from direction $\vec{\Omega}'$ to direction $\vec{\Omega}$ and from energy E' to energy E. In addition, if one compares Equations 2.1 and 2.28, one concludes that the scalar neutron flux and the current can be calculated as in Equations 2.29 and 2.30. One can also realize that the neutrons produced by fission events and decay of precursors are divided by 4π, because these events are considered to produce neutrons isotropically. Likewise the previous section, the concentration of precursors, C_k, changes over time and can be calculated with Equation 2.31.

$$\begin{aligned}
&\frac{1}{v(E)}\frac{d\psi(\vec{\Omega},E,\vec{r},t)}{dt} = -\nabla\vec{\Omega}\psi(\vec{\Omega},E,\vec{r},t) - \Sigma_t(E,\vec{r},t)\psi(\vec{\Omega},E,\vec{r},t) + \\
+ \quad &\int_0^\infty \int_{4\pi} \Sigma_s(\vec{\Omega}' \rightarrow \vec{\Omega}, E' \rightarrow E, \vec{r}, t)\psi(\vec{\Omega}', E', \vec{r}, t)d\vec{\Omega}'dE' + \\
+ \quad &(1-\beta)\frac{\chi(E,\vec{r},t)}{4\pi}\int_0^\infty \nu\Sigma_f(E',\vec{r},t)\int_{4\pi}\psi(\vec{\Omega}',E',\vec{r},t)d\vec{\Omega}'dE' + \\
+ \quad &\frac{1}{4\pi}\sum_{k=1}^{K}\chi_k^{del}(E,\vec{r},t)\lambda_k C_k(\vec{r},t)
\end{aligned} \tag{2.28}$$

$$\phi(E',\vec{r},t) = \int_{4\pi}\psi(\vec{\Omega}',E',\vec{r},t)d\vec{\Omega}' \tag{2.29}$$

$$\vec{J}(E',\vec{r},t) = \int_{4\pi}\vec{\Omega}'\psi(\vec{\Omega}',E',\vec{r},t)d\vec{\Omega}' \tag{2.30}$$

$$\frac{dC_k(\vec{r},t)}{dt} = \beta_k\int_0^\infty\int_{4\pi}\nu\Sigma_f(E',\vec{r},t)\psi(\vec{\Omega}',E',\vec{r},t)d\vec{\Omega}'dE' - \lambda_k C_k(\vec{r},t), \qquad k = 1,...,K \tag{2.31}$$

One can also use the multigroup approach in the previous equations to deal with the energy variables. As explained in Section 2.1, the multigroup approach obtains energy-averaged values in each group, which are expressed with the sub-index g, by integrating each variable with a weighting function $f(E)$. This function should be a good guess of the energy distribution of neutron flux to conserve the interaction rates. Actually, this function might also take into

account the angular distribution for cross sections depending on directional variables, which makes more difficult to find out the suitable functions. If one considers that these functions $f(E)$ are known, one can apply the multi-group approach to Equations 2.28 and 2.31, which gives Equations 2.32 and 2.33. There are only two different terms in these equations with respect to the multrigroup Neutron Diffusion Equation: ψ_g, which is the angular neutron flux, and $\Sigma_{s,g'\to g}(\vec{\Omega}' \to \vec{\Omega}, \vec{r}, t)$, which is the scattering cross section from the energy group g' to g and from the direction $\vec{\Omega}'$ to $\vec{\Omega}$. These terms, ψ_g and $\Sigma_{s,g'\to g}(\vec{\Omega}' \to \vec{\Omega}, \vec{r}, t)$, are defined in Equations 2.34 and 2.35.

$$\begin{aligned}\frac{1}{v_g}\frac{d\psi_g(\vec{\Omega},\vec{r},t)}{dt} &= -\nabla\vec{\Omega}\psi_g(\vec{\Omega},\vec{r},t) - \Sigma_{t,g}(\vec{r},t)\psi_g(\vec{\Omega},\vec{r},t) + \\ &+ \sum_{g'=1}^{G}\int_{4\pi}\Sigma_{s,g'\to g}(\vec{\Omega}'\to\vec{\Omega},\vec{r},t)\psi_{g'}(\vec{\Omega}',\vec{r},t)d\vec{\Omega}' + \\ &+ (1-\beta)\frac{\chi_g(\vec{r},t)}{4\pi}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t)\int_{4\pi}\psi_{g'}(\vec{\Omega}',\vec{r},t)d\vec{\Omega}' + \\ &+ \frac{1}{4\pi}\sum_{k=1}^{K}\chi_{g,k}^{del}(\vec{r},t)\lambda_k C_k(\vec{r},t) \end{aligned} \tag{2.32}$$

$$\frac{dC_k(\vec{r},t)}{dt} = \beta_k\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t)\int_{4\pi}\psi_{g'}(\vec{\Omega}',\vec{r},t)d\vec{\Omega}' - \lambda_k C_k(\vec{r},t), \qquad k = 1,...,K \tag{2.33}$$

$$\psi_g(\vec{\Omega},\vec{r},t) = \int_{E_g}^{E_{g-1}}\psi(\vec{\Omega},E,\vec{r},t)dE \tag{2.34}$$

$$\Sigma_{s,g'\to g}(\vec{\Omega}'\to\vec{\Omega},\vec{r},t) = \int_{E_{g'}}^{E_{g'-1}}dE'\int_{E_g}^{E_{g-1}}\Sigma_s(\vec{\Omega}'\to\vec{\Omega},E'\to E,\vec{r},t)f(E')dE \tag{2.35}$$

Currently, the analytical solution of Equations 2.32 and 2.33 in any geometry composed of different materials is not possible. Therefore, one has to discretize the geometry and apply numerical methods, as stated in Section 2.1. In the same way, one has to ensure that ψ is continuous in the whole domain. Consequently, for any adjacent regions V_1 and V_2, whose interface is $\vec{r}_{V_1\cap V_2}$, Equation 2.36 must be accomplished.

$$\psi(\vec{r}_{V_1 \cap V_2} \in V_1) = \psi(\vec{r}_{V_1 \cap V_2} \in V_2) \tag{2.36}$$

As regards the boundary conditions, one has to define them only for the incoming neutrons. The boundary conditions most used are: vacuum, specular reflection, albedo and white boundary, which are defined in Equations 2.37-2.40 respectively. In these equations, $\vec{\Omega}_{in}$ and $\vec{\Omega}_{out}$ are incoming and outgoing directions with respect to the boundary $\vec{r}_b$; $\vec{n}_{\vec{r}_b}$ is the normal direction to boundary $\vec{r}_b$. The white boundary condition means the following: all neutrons leaving the system through the boundary are isotropically emitted back into the domain. The code developed in this thesis does not include the white boundary condition.

$$\psi(\vec{\Omega}_{in}, \vec{r}_b) = 0 \tag{2.37}$$

$$\psi(\vec{\Omega}_{in}, \vec{r}_b) = \psi(\vec{\Omega}_{out}, \vec{r}_b) \tag{2.38}$$

$$\psi(\vec{\Omega}_{in}, \vec{r}_b) = \alpha\, \psi(\vec{\Omega}_{out}, \vec{r}_b) \tag{2.39}$$

$$\psi(\vec{\Omega}_{in}, \vec{r}_b) = \frac{\int_{\vec{\Omega}\cdot\vec{n}_{\vec{r}_b}} d\vec{\Omega} |\vec{\Omega} \cdot \vec{n}_{\vec{r}_b}| \psi(\vec{\Omega}, \vec{r}_b)}{\int_{\vec{\Omega}\cdot\vec{n}_{\vec{r}_b}} d\vec{\Omega} |\vec{\Omega} \cdot \vec{n}_{\vec{r}_b}|} \tag{2.40}$$

Likewise the steady state of the Neutron Diffusion Equation, one obtains the steady state of the Neutron Transport Equation by transforming Equations 2.32 and 2.33 into an eigenvalue problem, which is shown in Equation 2.41. In addition, one can simplify this equation if $\chi_g = \chi^{del}_{g,k}$, obtaining Equation 2.42.

$$\begin{aligned} 0 &= -\nabla\vec{\Omega}\psi_g(\vec{\Omega}, \vec{r}) - \Sigma_{t,g}(\vec{r})\psi_g(\vec{\Omega}, \vec{r}) + \\ &+ \sum_{g'=1}^{G} \int_{4\pi} \Sigma_{s,g'\to g}(\vec{\Omega}' \to \vec{\Omega}, \vec{r})\psi_{g'}(\vec{\Omega}', \vec{r}) d\vec{\Omega}' + \\ &+ \frac{1}{4\pi\,\mathbf{k}} \left((1-\beta)\chi_g(\vec{r}) + \sum_{k=1}^{K} \chi^{del}_{g,k}(\vec{r})\beta_k \right) \sum_{g'=1}^{G} \nu\Sigma_{f,g'}(\vec{r}) \int_{4\pi} \psi_{g'}(\vec{\Omega}', \vec{r}) d\vec{\Omega}' \end{aligned} \tag{2.41}$$

$$\begin{aligned} 0 &= -\nabla\vec{\Omega}\psi_g(\vec{\Omega},\vec{r}) - \Sigma_{t,g}(\vec{r})\psi_g(\vec{\Omega},\vec{r}) + \\ &+ \sum_{g'=1}^{G}\int_{4\pi}\Sigma_{s,g'\to g}(\vec{\Omega}'\to\vec{\Omega},\vec{r})\psi_{g'}(\vec{\Omega}',\vec{r})d\vec{\Omega}' + \\ &+ \frac{\chi_g(\vec{r})}{4\pi\,\mathbf{k}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r})\int_{4\pi}\psi_{g'}(\vec{\Omega}',\vec{r})d\vec{\Omega}' \end{aligned} \tag{2.42}$$

Equations 2.41 or 2.42 contain not only spatial derivatives terms, but also integral terms depending on the direction variable. Thus, one has to use numerical methods to discretize the direction variables, besides the numerical methods for the spatial variables. There are two methods for dealing with the direction variable: Spherical Harmonics (P_n) and Discrete Ordinates (S_n) methods.

The Spherical Harmonics method is based on the expansion of ψ_g with spherical harmonics, but truncating the expansion up to term n, as shown in Equation 2.43. In this equation, $Y_l^m(\vec{\Omega})$ are the spherical harmonics functions and $\psi_{g,l}^m(\vec{r})$ are the moments of ψ_g, which are calculated as shown in Equation 2.44. In this equation, $\overline{Y}_l^m(\vec{\Omega})$ is the complex conjugate of $Y_l^m(\vec{\Omega})$. However, one cannot calculate $\psi_{g,l}^m(\vec{r})$, because one does not know the angular distribution of ψ_g. Instead, the Spherical Harmonics method consists in substituting Equation 2.43 into Equation 2.45, so one can determine $\psi_{g,l}^m(\vec{r})$. The reader can find further details of this method in the literature, for example in Henry 1975.

$$\psi_g(\vec{\Omega},\vec{r}) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l}\psi_{g,l}^m(\vec{r})Y_l^m(\vec{\Omega}) \approx \sum_{l=0}^{n}\sum_{m=-l}^{l}\psi_{g,l}^m(\vec{r})Y_l^m(\vec{\Omega}) \tag{2.43}$$

$$\psi_{g,l}^m(\vec{r}) = \int_{4\pi}\overline{Y}_l^m(\vec{\Omega})\psi_g(\vec{\Omega},\vec{r})d\vec{\Omega} \tag{2.44}$$

$$
\begin{aligned}
0 \;=\; & \int_{4\pi} \overline{Y}_l^m(\vec{\Omega}) \Bigg(- \nabla\vec{\Omega}\psi_g(\vec{\Omega},\vec{r}) - \Sigma_{t,g}(\vec{r})\psi_g(\vec{\Omega},\vec{r}) + \\
& + \sum_{g'=1}^{G} \int_{4\pi} \Sigma_{s,g'\to g}(\vec{\Omega}' \to \vec{\Omega},\vec{r})\psi_{g'}(\vec{\Omega}',\vec{r})d\vec{\Omega}' + \\
& + \frac{\chi_g(\vec{r})}{4\pi\,\mathbf{k}} \sum_{g'=1}^{G} \nu\Sigma_{f,g'}(\vec{r}) \int_{4\pi} \psi_{g'}(\vec{\Omega}',\vec{r})d\vec{\Omega}') \Bigg) d\vec{\Omega} \\
& , l = 0, \cdots, n; \;\; m = -l, \cdots, l
\end{aligned} \tag{2.45}
$$

On the other hand, the Discrete Ordinates method is based on a discretization of $\vec{\Omega}$ in a set of discrete directions $\vec{\Omega}_n$, which are also called quadrature sets. One has to choose these $\vec{\Omega}_n$ to obtain the maximum accuracy in the integrals containing $\vec{\Omega}$. For an arbitrary function $f(\vec{\Omega})$, the goal of this method is to obtain the weights w_n and the directions $\vec{\Omega}_n$ to solve with the highest accuracy the integration of Equation 2.46. A number of quadrature sets has been developed for calculating this integration. The ones most used are: level-symmetric, Legendre-Chebyshev and product quadrature. The explanation of how one obtains these quadrature sets is beyond the scope of this document, but the author will describe its main features and show some examples. In the level-symmetric quadrature, one selects point on the unit sphere in such a way to preserve the symmetry of the eight octants with respect to $\pi/2$ rotations and conserve the odd and even moments of μ, η and ξ. Figure 2.2 shows an example of the level-symmetric quadrature. The Legendre-Chebyshev quadrature aims to conserve moments to a maximum order without the constraints of the symmetry condition. This quadrature has a similar triangle-shaped direction layout on the unit sphere, such as that of level-symmetric. Although the lack of symmetry, this quadrature is the best choice for mathematically conserving higher moments. As regards the product quadrature, it is based on the separation of variables: $f(\vec{\Omega}) = f_\mu(\mu)f_\varphi(\varphi)$. Then, the integration of f is performed as in Equation 2.47.

$$
\int_{4\pi} f(\vec{\Omega})d\vec{\Omega} \approx \sum_{n=1}^{N_d} w_n f(\vec{\Omega}_n) \tag{2.46}
$$

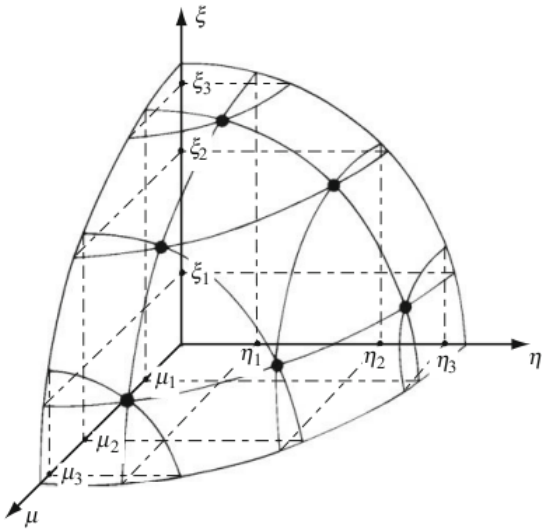


Figure 2.2: S_6 level-symmetric quadrature

$$\begin{aligned}\int_{4\pi} f(\vec{\Omega})d\vec{\Omega} &= \int_{-1}^{1} f_\mu(\mu)d\mu \int_0^{2\pi} f_\varphi(\varphi)d\varphi \approx \\ &\approx \sum_{n_\mu=1}^{N_\mu} w_{n_\mu} f_\mu(\mu_{n_\mu}) \sum_{n_\varphi=1}^{N_\varphi} w_{n_\varphi} f_\varphi(\varphi_{n_\varphi}) = \\ &= \sum_{n_\mu=1}^{N_\mu} \sum_{n_\varphi=1}^{N_\varphi} w_{n_\mu} w_{n_\varphi} f_\mu(\mu_{n_\mu}) f_\varphi(\varphi_{n_\varphi}) = \sum_{n=1}^{N_d} w_n f(\mu_n, \varphi_n)\end{aligned} \tag{2.47}$$

One can directly apply the integration of Equation 2.46 to Equation 2.42, but it is better to expand the scattering with spherical harmonics, as explained in Hébert 2009. This involves expanding ψ of the scattering terms, as in Equation 2.43, and expanding $\Sigma_{s,g'\to g}(\vec{\Omega}' \to \vec{\Omega})$ with Legendre polynomials (P_l), as shown in Equation 2.48, in which $\Sigma_{s,g'\to g,l}$ is defined in Equation 2.49 and L is the order of the expansion. These two expansions might provide a long and difficult expression of the scattering terms. However, one could simplify the integration of the scattering term, obtaining the expression of Equation

2.50, as explained in Appendix A of Yi 2009. In this equation, $\phi_{g',l}$, $\phi^k_{C,g',l}$ and $\phi^k_{S,g',l}$ are defined in Equations 2.51-2.53. In addition, in these last equations P_l is the Legendre polynomial of order l and P^k_l is the associated Legendre polynomial of orders l and k.

$$\Sigma_{s,g'\to g}(\vec{\Omega}'\to\vec{\Omega}) = \Sigma_{s,g'\to g}(\vec{\Omega}'\cdot\vec{\Omega}) = \Sigma_{s,g'\to g}(\mu_0) \approx \sum_{l=0}^{L}\frac{2l+1}{2}\Sigma_{s,g'\to g,l}P_l(\mu_0) \tag{2.48}$$

$$\Sigma_{s,g'\to g,l} = \int_{-1}^{1} P_l(\mu_0)\Sigma_{s,g'\to g}(\mu_0)d\mu_0 \tag{2.49}$$

$$\int_{4\pi}\Sigma_{s,g'\to g}(\vec{\Omega}'\to\vec{\Omega},\vec{r})\psi_{g'}(\vec{\Omega}',\vec{r})d\vec{\Omega}' = \sum_{l=0}^{L}(2l+1)\Sigma_{s,g'\to g,l}\Bigg\{P_l(\mu)\phi_{g',l} + \\ +2\sum_{k=1}^{l}\frac{(l-k)!}{(l+k)!}P^k_l(\mu)\cdot\left[\phi^k_{C,g',l}\cos(k\varphi)+\phi^k_{S,g',l}\sin(k\varphi)\right]\Bigg\} \tag{2.50}$$

$$\phi_{g',l} = \frac{1}{4\pi}\int_{-1}^{1}\int_{0}^{2\pi} P_l(\mu')\psi_{g'}(\mu',\varphi')d\mu'd\varphi' \tag{2.51}$$

$$\phi^k_{C,g',l} = \frac{1}{4\pi}\int_{-1}^{1}\int_{0}^{2\pi} P^k_l(\mu')\cos(k\varphi')\psi_{g'}(\mu',\varphi')d\mu'd\varphi' \tag{2.52}$$

$$\phi^k_{S,g',l} = \frac{1}{4\pi}\int_{-1}^{1}\int_{0}^{2\pi} P^k_l(\mu')\sin(k\varphi')\psi_{g'}(\mu',\varphi')d\mu'd\varphi' \tag{2.53}$$

If one substitutes Equations 2.50 and 2.51 in Equation 2.42, one obtains Equation 2.54. Finally, one obtains the Discrete Ordinates formulation of this equation by setting $\vec{\Omega} = \vec{\Omega}_n = \vec{\Omega}_n(\mu_n,\varphi_n)$, obtaining Equation 2.55. In this equation $\phi_{g',l}$, $\phi^k_{C,g',l}$ and $\phi^k_{S,g',l}$ are integrated with Equation 2.46, as shown in Equations 2.56-2.58.

$$
\begin{aligned}
0 &= -\nabla\vec{\Omega}\psi_g(\vec{\Omega},\vec{r}) - \Sigma_{t,g}(\vec{r})\psi_g(\vec{\Omega},\vec{r}) + \\
&+ \sum_{g'=1}^{G}\sum_{l=0}^{L}(2l+1)\Sigma_{s,g'\to g,l}\Bigg\{P_l(\mu)\phi_{g',l} + \\
&+ 2\sum_{k=1}^{l}\frac{(l-k)!}{(l+k)!}P_l^k(\mu)\cdot\left[\phi^k_{C,g',l}\cos(k\varphi) + \phi^k_{S,g',l}\sin(k\varphi)\right]\Bigg\} + \\
&+ \frac{\chi_g(\vec{r})}{\mathbf{k}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r})\phi_{g',0}
\end{aligned}
\tag{2.54}
$$

$$
\begin{aligned}
0 &= -\nabla\vec{\Omega}_n\psi_g(\vec{\Omega}_n,\vec{r}) - \Sigma_{t,g}(\vec{r})\psi_g(\vec{\Omega}_n,\vec{r}) + \\
&+ \sum_{g'=1}^{G}\sum_{l=0}^{L}(2l+1)\Sigma_{s,g'\to g,l}\Bigg\{P_l(\mu_n)\phi_{g',l} + \\
&+ 2\sum_{k=1}^{l}\frac{(l-k)!}{(l+k)!}P_l^k(\mu_n)\cdot\left[\phi^k_{C,g',l}\cos(k\varphi_n) + \phi^k_{S,g',l}\sin(k\varphi_n)\right]\Bigg\} + \\
&+ \frac{\chi_g(\vec{r})}{\mathbf{k}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r})\phi_{g',0}
\end{aligned}
\tag{2.55}
$$

$$
\phi_{g',l} \approx \sum_{m=1}^{N_d} w_m P_l(\mu_m)\psi_{g'}(\mu_m,\varphi_m) \tag{2.56}
$$

$$
\phi^k_{C,g',l} \approx \sum_{m=1}^{N_d} w_m P_l^k(\mu_m)\cos(k\varphi_m)\psi_{g'}(\mu_m,\varphi_m) \tag{2.57}
$$

$$
\phi^k_{S,g',l} \approx \sum_{m=1}^{N_d} w_m P_l^k(\mu_m)\sin(k\varphi_m)\psi_{g'}(\mu_m,\varphi_m) \tag{2.58}
$$

Although the Discrete Ordinates method might be the simplest method, this method suffers from a limitation called ray effect in 2D and 3D geometries

(Lathrop 1968). The ray effect is an oscillation of the flux distribution about its exact value that can be observed in cases with very little scattering and localized sources. This is caused by the discretization of the angular variable over the directions of the quadrature, producing a numerical solution which is represented as a sum of delta distributions in angle, though the exact solution is a continuous distribution in angle.

Furthermore, there are other methods to deal with the Neutron Transport Equation, such as the Method of Characteristics and the Collision Probability, but they solve other formulations of Equation 2.42. On the one hand, the Method Of Characteristics solves the characteristic form of the transport equation by following the straight neutron paths of the neutron as it moves across the complete domain, as discussed in Askew 1972. The explanation of the characteristic formulation is beyond the scope of this work, but it corresponds to an integration of $\nabla\vec{\Omega}\psi$ over the characteristic, which is a straight line direction $\vec{\Omega}$ corresponding to the particle trajectory. On the other hand, the Collision Probability method is obtained from the spatial discretization of the integral formulation of the Neutron Transport Equation assuming isotropic particle sources. One obtains this integral formulation by integrating the angular flux along its characteristic, for a given value of the source density. The interested reader might find further details in the literature, for example in Hébert 2009.

All the methods mentioned in this section are deterministic methods, that is, they discretize the independent variables of the Neutron Transport Equation in order to obtain the neutron distribution. By contrast, there is one method which does not need to discretize this equation, the Monte Carlo method. This method obtains the neutron distribution by simulating the different events that can occur in the domain, which are the nuclear interactions and reactions. In this method, these events are simulated in a sequential form by taking into account the probabilities of occurrence and using random numbers, thus, it is a stochastic method. Therefore, the solution of the problem is equivalent to a sampling of the different events. The major capability of the Monte Carlo method is the solution of complex problems in terms of energy and geometry. It can solve continuous energy problems and different types of geometry due to the fact that it does not deal with the integral form of the Transport Equation. Nevertheless, this method implies errors because of the statistical resolution, which can be reduced by increasing the number of events simulated, thereby, increasing the computational time.

Finally there is a large number of codes for solving the Neutron Transport Equation by means of the deterministic methods described in this section. Examples of these are: CASMO (Edenius et al. 1995), NEWT (DeHart and Jessee 2005), Polaris (Jessee et al. 2014), Denovo (Evans et al. 2010), DRAGON (Marleau, Hébert, and Roy 2011), TORT (Rhoades and Simpson 1997), APOLLO2 (Sanchez et al. 1988), CRONOS2 (Lautard, Loubiere, and Fedon-Magnaud 1990), MPACT (Kochunas et al. 2013), PARTISN (Alcouffe et al. 2005), TITAN (Yi 2009), DANTSYS (Alcouffe et al. 1995), VARIANT (Palmiotti, Lewis, and Carrico 1995), DeCART (Cho et al. 2003), ATILLA (Wareing, McGhee, and Morel 1996), etc. There are also several Monte Carlo codes, such as: MCNP (Brown 2003), KENO (Goluoglu et al. 2011), SERPENT (Leppänen 2013), etc.

2.3 Spatial Discretization

The spatial discretization of some domain consists in transforming the domain into a set of elements with simple geometry and composed of only one material. This set of elements is called a mesh, which can be classified in two types: structured and unstructured meshes.

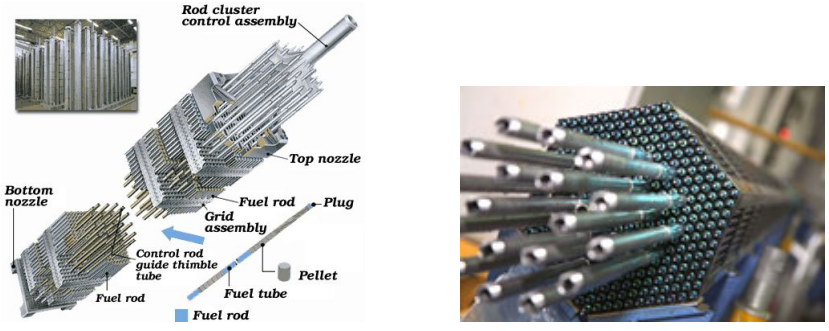
Structured meshes are those whose elements have a regular connectivity, so one can store them in 2D or 3D dimensional arrays. The main advantage of the structured meshes is that the neighborhood relations are defined by storage arrangement, and consequently they required less pre-processing operations. The major drawback of structured meshes is that they cannot model accurately realistic reactor geometry. Examples of 3D structured meshes are: Cartesian, cylindrical and spherical meshes. Each of these meshes divides the domain in each of their 3 independent variables: (x, y, z) for Cartesian meshes, (r, φ, z) for cylindrical meshes and (r, φ, θ) for spherical meshes. In addition, some hexagonal meshes could have a regular connectivity, so one could consider these particular meshes as structured meshes too.

By contrast, unstructured meshes are those whose elements do not have a regular connectivity. These meshes can have any and several kind of elements, like triangles, quadrangles or any polygon in 2D; tetrahedra, pyramids, hexahedra or any polyhedron in 3D. The main advantage of these meshes is that they can model any kind of geometry. However, the major downside is that they require more pre-processing operations than the structured meshes. In unstructured meshes, elements are numbered sequentially, as are faces, nodes, and other geometric quantities. This means that there is no direct way to link

various entities together based on their indices alone. Thus local connectivity has to be defined explicitly starting with determining the geometric quantities for a particular element, as explained in Moukalled, Mangani, and Darwish 2015.

The disctretization of the geometry with structured meshes is straightforward, but one should use grid generators to discretize efficiently the domain with unstructured meshes. There is a large number of free and open-source or commercial mesh generators, such as Gmsh, enGrid, Netgen, Discretizer, snappyHexMesh, ICEM-CFSD, CUBIT, etc. In this work, the author used Gmsh (Geuzaine and Remacle 2009), which is a free 3D finite element grid generator with a built-in Computer-aided design engine and post-processor. The author chose this grid generator due to four reasons. First, it is fast and user-friendly. Second, it can generate tetrahedral and hexahedral meshes. Third, it is free and open-source. Fourth, the developers keep it up to date.

As regards the geometry of nuclear reactor cores, these cores are composed of a number of fuel assemblies arranged in a certain configuration. The geometry of the fuel assemblies and the configuration depends on the type of reactors. Figure 2.3 shows some fuel assemblies for PWR and VVER. As a commercial nuclear reactor core might have hundreds of fuel assemblies, one habitually homogenizes each fuel assembly to perform full reactor core calculations. Consequently, one obtains homogenized parallelepipeds for PWR or BWR and homogenized hexagonal prisms for VVER, as shown in Figure 2.4. So, one can conclude that one can apply structured meshes for modeling LWR, but it is not straightforward for VVER. Finally, the calculations without homogenization might require unstructured meshes to deal with the details of the geometry.



PWR VVER

Figure 2.3: Fuel assemblies [www.world-nuclear.org]

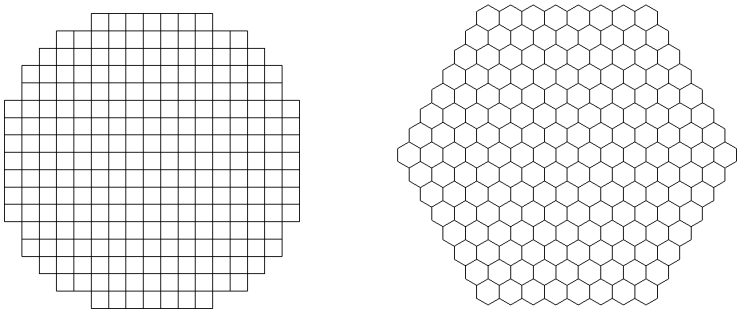


Figure 2.4: Configuration of fuel assemblies within typical cores

When one performs the homogenization of each assembly, one has to define cross sections, and also diffusion coefficients for the Neutron Diffusion Equation, for the homogenized assembly. These homogenized cross sections should be calculated to conserve in each assembly not only the interaction rates, but also the leakage rates.

Next, the homogenization problem and the main methods are presented, although the goal of this section is not to fully explain the homogenization methods. As regards the conservation of the interaction rates, one can define the conservation of certain interaction rate of type x, energy group g, in the assembly i as shown in Equation 2.59. In this equation, $\phi_g(\vec{r})$ is the neutron flux distribution, for the energy group g, of the assembly without homogenization, whereas $\hat{\phi}_g(\vec{r})$ is the neutron flux distribution, for the energy group g, of the homogenized assembly. Thus, one could calculate the homogenized cross section for this assembly ($\Sigma^i_{x,g}$) with Equation 2.60, if one knew $\phi_g(\vec{r})$ and $\hat{\phi}_g(\vec{r})$.

$$\Sigma^i_{x,g} \int_{V_i} \hat{\phi}_g(\vec{r}) dV = \int_{V_i} \Sigma_{x,g}(\vec{r}) \phi_g(\vec{r}) dV \tag{2.59}$$

$$\Sigma^i_{x,g} = \frac{\int_{V_i} \Sigma_{x,g}(\vec{r}) \phi(\vec{r}) dV}{\int_{V_i} \hat{\phi}(\vec{r}) dV} \tag{2.60}$$

On the other hand, one can define the conservation of the leakage rates for the Neutron Diffusion Equation as in Equation 2.61. In this equation, D^i_g is the

diffusion coefficient for the homogenized assembly i and energy group g, and S_j is the face j of the assembly. Similarly, if $\vec{J}_g(\vec{r})$ is known, one can calculate D_g^i as in Equation 2.62. However, in this equation one can appreciate that D_g^i might have different values depending on the face j, but one needs cell averaged values of D_g^i for the homogenized assembly. This issue introduces the first problem of the homogenization process: one cannot guarantee the conservation of the leakage rates, if one uses a cell averaged diffusion coefficient without adding additional equations. The second problem is that $\hat{\phi}_g(\vec{r})$ is not known, and clearly the spatial distribution might be different from the spatial distribution of $\phi_g(\vec{r})$.

$$D_g^i \int_{S_j} \vec{\nabla}\hat{\phi}(\vec{r})dS = \int_{S_j} \vec{J}_g(\vec{r})dS \tag{2.61}$$

$$D_g^i = \frac{\int_{S_j} \vec{J}_g(\vec{r})dS}{\int_{S_j} \vec{\nabla}\hat{\phi}(\vec{r})dS} \tag{2.62}$$

There are two main class of methods for calculating these homogenized cross sections and diffusion coefficients, while conserving the interaction and leakage rates: Discontinuity Factors (DFs) and SPH factors. The Discontinuity Factors were introduced in the Equivalence Theory (Wagner and Koebke 1983, Smith 1986). The Equivalence Theory is based on three points. First, the conservation of the neutron flux in each assembly ($\int_{V_i} \hat{\phi}_g(\vec{r})dV = \int_{V_i} \phi_g(\vec{r})dV$) to obtain the homogenized cross sections in each assembly. Second, the conservation of the leakage rate in the whole assembly to calculate a homogenized diffusion coefficient for each assembly. Third, use of DFs in each face of the assembly to take into account the conservation of the leakage rates at each face. The DFs proposed in Smith 1986 add extra equations in the faces by considering that $\hat{\phi}_g(\vec{r})$ might be discontinuous at the interfaces of different assemblies. Equation 2.63 defines these DFs for the energy group g, region V_1 and the interface $\vec{r}_{V_1 \cap V_2}$. Once these DFs are known, one uses them to redefine the flux continuity of the homogenized problem, as shown in Equation 2.64. This equation guarantees the continuity of the neutron flux for the assembly without homogenization. Although this DFs are widely used, other authors have used other definitions, such as Current Discontinuity Factors (Sanchez 2009).

$$DF_g(\vec{r}_{V_1 \cap V_2} \in V_1) = \frac{\phi_g(\vec{r}_{V_1 \cap V_2} \in V_1)}{\hat{\phi}_g(\vec{r}_{V_1 \cap V_2} \in V_1)} \tag{2.63}$$

$$DF_g(\vec{r}_{V_1 \cap V_2} \in V_1) \cdot \hat{\phi}_g(\vec{r}_{V_1 \cap V_2} \in V_1) = DF_g(\vec{r}_{V_1 \cap V_2} \in V_2) \cdot \hat{\phi}_g(\vec{r}_{V_1 \cap V_2} \in V_2) \tag{2.64}$$

The SPH factors were introduced in the superhomogenization method (Kavenoky 1980, Hébert and Kavenoky 1981). These factors are calculated to conserve the interaction and leakage rates in the homogenized assembly. The method consists in calculating the SPH factor $\mu_{g,i}$ for each energy group g and assembly i, as shown in Equation 2.65. By means of this SPH factor, one calculates the homogenized cross sections as in Equation 2.66. Since $\hat{\phi}(\vec{r})$ depends on the homogenized cross section, one has to determine this factor by iterative calculations. Further details about this calculation can be found in Hébert and Kavenoky 1981 and Hébert 2009.

$$\mu_{g,i} = \frac{\int_{V_i} \phi(\vec{r}) dV}{\int_{V_i} \hat{\phi}(\vec{r}) dV} \tag{2.65}$$

$$\Sigma^i_{x,g} = \frac{\int_{V_i} \Sigma_{x,g}(\vec{r}) \phi(\vec{r}) dV}{\int_{V_i} \hat{\phi}(\vec{r}) dV} = \mu_{g,i} \frac{\int_{V_i} \Sigma_{x,g}(\vec{r}) \phi(\vec{r}) dV}{\int_{V_i} \phi(\vec{r}) dV} \tag{2.66}$$

2.4 Finite Volume Method

The Finite Volume Method (FVM) is a numerical method that transforms the partial differential equations representing conservation laws over differential volumes into discrete algebraic equations over finite cells or elements, as discussed in Moukalled, Mangani, and Darwish 2015. Basically, the FVM transforms the partial differential equations into algebraic equations by integrating them over each discrete element (V_i), obtaining cell averaged values of the unknowns and face averaged values of the divergence terms, as stated in Moukalled, Mangani, and Darwish 2015 or Hoffmann and Chiang 2000.

Next, the author will give an example of the FVM for the partial differential equation of Equation 2.67, in which $f(\vec{r})$ is an arbitrary function and a is a constant coefficient. Equation 2.68 shows the application of the FVM to the previous partial differential equation. One can see in Equation 2.68 the use of the Divergence theorem for transforming the cell integrals (V_i) into face

integrals (S). In addition, the example considers that the face S of the cell V_i is composed of n_f faces $S_{i,j}$. Finally, one can see the use of cell averaged values f_i, defined in Equation 2.69, and face averaged values $f_{i,j}$, defined in Equation 2.70.

$$0 = \nabla f(\vec{r}) + af(\vec{r}) \tag{2.67}$$

$$\begin{aligned} 0 &= \int_{V_i} \nabla f(\vec{r})d\vec{r} + \int_{V_i} af(\vec{r})d\vec{r} = \oint_S f(\vec{r})d\vec{r} + \int_{V_i} af(\vec{r})d\vec{r} = \\ &= \sum_{j=1}^{n_f} \int_{S_{i,j}} f(\vec{r})d\vec{r} + \int_{V_i} af(\vec{r})d\vec{r} = \sum_{j=1}^{n_f} f_{i,j}S_{i,j} + af_iV_i \end{aligned} \tag{2.68}$$

$$f_i = \frac{\int_{V_i} f(\vec{r})d\vec{r}}{V_i} \tag{2.69}$$

$$f_{i,j} = \frac{\int_{S_{i,j}} f(\vec{r})d\vec{r}}{S_{i,j}} \tag{2.70}$$

Once the geometry is discretized in N_c cells, one can apply Equation 2.68 to each cell obtaining an algebraic system, which one has to solve to determine the cell averaged values f_i. The downside of the method is that the face averaged values $f_{i,j}$ are not known, so one has to guess these values from the cell averaged ones. However, FVM has two main strengths, as discussed in Moukalled, Mangani, and Darwish 2015 and in Hoffmann and Chiang 2000. First, it is strictly conservative, since the flux entering a given cell is identical to that leaving the adjacent cell. Second, it can be formulated in the physical space on unstructured polygonal meshes.

There is a number of interpolation schemes to determine the face averaged values, as discussed in Moukalled, Mangani, and Darwish 2015. The schemes most known are: central difference (or linear interpolation), upwind, second order upwind, QUICK (Quadratic Upstream Interpolation for Convective Kinematics), FROMM, least squares, moving least squares, High Resolution schemes and Gudonov-type schemes.

Figures 2.5-2.9 display the first interpolation schemes, from central difference to FROMM. In these figures, i_1, i_2 and i_3 are the cells; f_i is the cell averaged value for cell i and f_j is the face averaged value for face j, which is the interpolated

value; $\vec{u}$ is the direction of the flux. These figures show clearly the interpolation for all schemes. Nevertheless, the author would like to point out that QUICK obtains f_j by considering a quadratic function of f in which $f_i = f(i)$. As regards the accuracy, upwind is first order accurate, QUICK is third order accurate, and the rest are second order accurate. Although upwind is the least accurate, it is the most stable. The numerical stability is of great interest in the numerical solution, because it is related with the production of oscillations in the numerical solution, which are a non-physical behavior of the real solution. Another important issue is the numerical diffusion due to the methods used for discretizing the equations. These methods might model the diffusion terms of the equations with a different behavior of the real system.

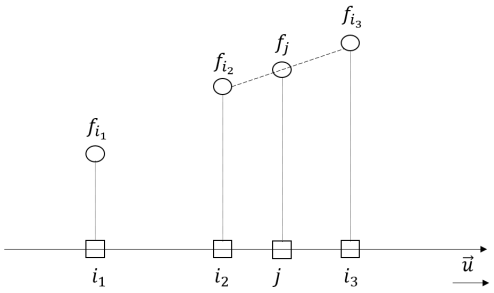


Figure 2.5: Central difference scheme

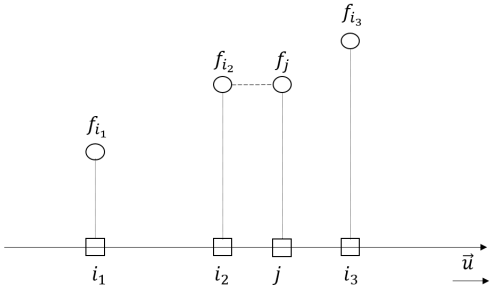


Figure 2.6: Upwind scheme

Other schemes are those based on least squares or moving least squares. Least squares schemes consist in expanding f as a sum of n polynomial terms $p_k(x, y, z)$, whose coefficients a_k are not known, as shown in Equation 2.71. Then, one calculates these coefficients by solving the least squares problem to minimize

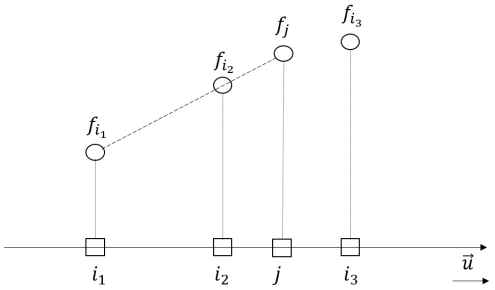


Figure 2.7: Second order upwind scheme

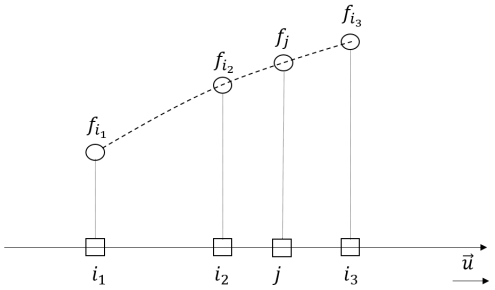


Figure 2.8: QUICK scheme

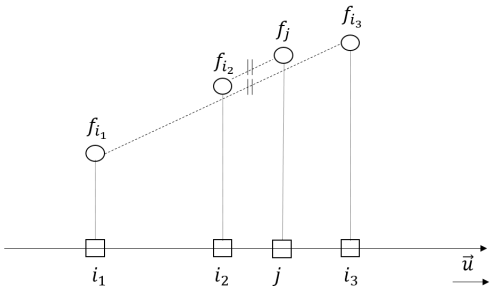


Figure 2.9: FROMM scheme

the error of Equation 2.72. In this equation, f_{i_l} is the cell averaged value of f in cell i_l, m is the number of cells considered for the interpolation, and $(x_{i_l}, y_{i_l}, z_{i_l})$ are the coordinates of the centroid of cell i_l. Likewise, moving least squares schemes also expand the function in the same way, but one minimizes the error of Equation 2.73. The only difference is that the error is weighted by a function $W(x, y, z)$, with the purpose of obtaining smooth interpolations. One can find examples of this method in Cueto-Felgueroso et al. 2007.

$$f(x, y, z) \approx \sum_{k=1}^{n} a_k p_k(x, y, z) \tag{2.71}$$

$$error = \sum_{l=1}^{m} \left(f(x_{i_l}, y_{i_l}, z_{i_l}) - f_{i_l}\right)^2 = \sum_{l=1}^{m} \left(\sum_{k=1}^{n} a_k p_k(x_{i_l}, y_{i_l}, z_{i_l}) - f_{i_l}\right)^2 \tag{2.72}$$

$$error = \sum_{l=1}^{m} \left(f(x_{i_l}, y_{i_l}, z_{i_l}) - f_{i_l}\right)^2 W(x, y, z) \tag{2.73}$$

On the other hand, High Resolution schemes were developed to obtain second or higher order accuracy without oscillations. There is a number of these schemes, but the most known and used are: MUSCL and WENO. MUSCL (Monotonic Upwind Scheme for Conservation Laws) was first defined in Van Leer 1997. Basically, MUSCL combines first and second order schemes in a non-linear way to obtain a scheme which is stable second order accurate. For example, one might combine upwind and central difference schemes of Figures 2.6 and 2.5 as in Equation 2.74, with a weight δ depending on the ratio of the slopes $s^D_{i2,j}$ and $s^U_{i2,j}$ defined in Equations 2.75 and 2.76. Likewise, one may combine upwind and second order upwind schemes as in Equation 2.77. There could be a large number of combinations, but one can define a general formulation of MUSCL as in Equation 2.78, in which h can be any function of $f^U_{i_2,j}$ and $f^D_{i_2,j}$, which are defined in Equations 2.79 and 2.80. In these last equations, SL is a slope limiter function. There are a wide variety of slope limiters, such as: CHARM (Zhou 1995), HCUS (Waterson and Deconinck 1995), HQUICK (Waterson and Deconinck 1995), Koren (Koren 1993), minmod (Roe 1986), superbee (Roe 1986), ospre(Waterson and Deconinck 1995), smart (Gaskell and Lau 1988), Sweby (Sweby 1984), van Leer (Van Leer 1997), etc.

$$f_j = \delta\left(\frac{s^U_{i2,j}}{s^D_{i2,j}}\right) \cdot f_{i_2} + \left[1 - \delta\left(\frac{s^U_{i2,j}}{s^D_{i2,j}}\right)\right] \cdot \left[f_{i_2} + \frac{r_j - r_{i_2}}{r_{i_3} - r_{i_2}}(f_{i_3} - f_{i_2})\right] \tag{2.74}$$

$$s^D_{i2,j} = \frac{f_{i_3} - f_{i_2}}{r_{i_3} - r_{i_2}} \tag{2.75}$$

$$s^U_{i2,j} = \frac{f_{i_2} - f_{i_1}}{r_{i_2} - r_{i_1}} \tag{2.76}$$

$$f_j = \delta\left(\frac{s^U_{i2,j}}{s^D_{i2,j}}\right) \cdot f_{i_2} + \left[1 - \delta\left(\frac{s^U_{i2,j}}{s^D_{i2,j}}\right)\right] \cdot \left[f_{i_2} + \frac{r_j - r_{i_2}}{r_{i_2} - r_{i_1}}(f_{i_2} - f_{i_1})\right] \tag{2.77}$$

$$f_j = h\left(f^U_{i_2,j}, f^D_{i_2,j}\right) \tag{2.78}$$

$$f^U_{i_2,j} = f_{i_2} + SL\left(\frac{s^U_{i2,j}}{s^D_{i2,j}}\right) \cdot s^U_{i2,j} \cdot (r_j - r_{i_2}) \tag{2.79}$$

$$f^D_{i_2,j} = f_{i_2} + SL\left(\frac{s^U_{i2,j}}{s^D_{i2,j}}\right) \cdot s^D_{i2,j} \cdot (r_j - r_{i_2}) \tag{2.80}$$

On the other hand, the general formulation of WENO (Weighted Essentially Non Oscillatory) was developed in Shu 1998. The method consists in calculating f_j as a weighted sum of different values f^s_j as shown in Equation 2.81, in which the weights w_s depend on the cell averaged values f_i. First, these values f^s_j are interpolated values of f, in face j, from certain set s containing n_s cell averaged values f_i. To calculate f^s_j, one expands f as a sum of n_s polynomials $p_{s,k}$ with coefficients $a_{s,k}$, as shown in Equation 2.82. One can calculate these coefficients $a_{s,k}$ by substituting the coordinates and values of cells i_k, so one obtains f^s_j as a weighted sum of f_i, as one can see in Equation 2.82. The author does not include in this section the calculation of the weights w_s because of the extent of the explanation, but the interested reader can obtain the complete definition in Shu 1998.

$$f_j = \sum_{s=0}^{n-1} w_s f^s_j \tag{2.81}$$

$$f^s_j = \sum_{k=1}^{n_s} a_{s,k} p_{s,k}(r_j) = \sum_{k=1}^{n_s} b_{s,k} f_{i_k} \tag{2.82}$$

In addition, Godunov-type methods are an important class of numerical methods for hyperbolic conservation laws, since these equations contain time derivatives terms. These methods use the exact solution of the Riemann problem and do not produce oscillations around discontinuities (Berkenbosch, Kaasschieter, and Thije Boonkkamp 1994). Basically, Godunov-type methods assume a piecewise constant numerical solution in each cell and the time evolution of the solution is obtained by solving an initial value Riemann problem in each computational cell (Macian-Juan and Mahaffy 1998).

Finally, an excellent review of the different schemes is performed in Macian-Juan and Mahaffy 1998. This work stated that simple second order methods have a good behavior in smooth regions, yet these methods suffer from oscillatory behavior around discontinuities. Moreover, the work highlights the use of high order methods with flux limiters to increase the accuracy of the solution and avoid the oscillations near the discontinuities. An important drawback of these methods is that these limiter functions might depend on the flux, so one has to perform an iterative calculation to solve the algebraic system.

2.5 Calculation of Eigenvalue Problems

Eigenvalue problems coming from discretized systems of equations are expressed in matrix form as in Equation 2.83, in which A and B are square matrices, x is the eigenvector and λ is the eigenvalue. If matrix B is non-singular, one can define the standard eigenvalue problem as in Equation 2.84, in which $\mathcal{A} = B^{-1}A$. One might obtain the eigenvalues of 2.84 by finding the roots of the characteristic polynomial or by diagonalizing $\mathcal{A}$. However, these operations are costly for large matrices. Fortunately, these matrices are sparse and one might need only few eigenvalues. In this case, there are other kind of methods solving this kind of eigenvalue problems in a more efficient way, which are the iterative methods.

$$Ax = \lambda Bx \tag{2.83}$$

$$\mathcal{A}x = \lambda x \tag{2.84}$$

The iterative methods for solving eigenvalue problems can be classified in: Single Vector Iteration, Subspace Iteration, Krylov subspace, Davidson methods, Newton Methods and Block Newton Methods. Single Vector Iteration methods

were the first ones employed in the sparse eigenvalue scenario. These methods come from the observation that a vector that is repeatedly multiplied by matrix $\mathcal{A}$ usually tends to align in the direction of the eigenvector associated to its dominant eigenvalue. The ones most known are: Power Iteration, Inverse Iteration and Rayleigh Quotient Iteration. Nevertheless, these methods might be the least competitive for calculating multiple eigenvalues. More competitive methods for calculating multiple eigenvalues are Subspace Iteration and Krylov subspace methods, which are classed in the group of projection methods. A great overview of projection methods is performed in Saad 1983, particularly for large sparse eigenvalue problems. As regards Subspace Iteration methods, they can be seen as a generalization of the Power Iteration Method in the sense that they iterate simultaneously on m initial vectors, instead of just one. In spite of this enhancement, Subspace Iteration methods are generally inferior to Krylov subspace methods. Examples of Krylov subspace methods are: Arnoldi, Lanczos and Krylov-Schur methods. Krylov subspace methods are widely used to compute the eigenvalues in the extremes of the spectrum of standard eigenvalue problems, because they are the most competitive. Nonetheless, Davidson methods may present better performance computing interior eigenvalues close to a given target. The author will give a brief overview of each method in the following paragraphs, except the Davidson methods, because the important eigenvalues in Reactor Physics are those of the extremes of the spectrum. Finally, Newton Methods can be very fast and suitable, particularly for nonlinear problems or generalized eigenvalue problems. A special case of Newton Methods are the Block Newton Methods, which have the same features as the Newton Methods, but they are used for calculating several eigenvalues in an efficient way.

The Power Iteration method consists in premultiplying a vector x_0 by matrix $\mathcal{A}$ repeatedly, that is, generating the sequence of vectors $\mathcal{A}^k x_0$. If $\mathcal{A}$ has a unique eigenvalue of largest modulus λ_1, and x_0 is not deficient in the direction of the corresponding eigenvector x_1, the method converges to a multiple of x_1. The rate of convergence is linear and depends on the factor $|\lambda_1/\lambda_2|$, being λ_2 the next largest eigenvalue in magnitude. Therefore, convergence might be slow if these two eigenvalues are close in magnitude and will not be achieved at all in the case that there is no unique dominant eigenvalue. One can find some algorithms of this method in Hernandez et al. 2005a.

The Inverse Iteration method is similar to the Power Iteration method, but one iterates with the matrix $\mathcal{A}^{-1}$ instead of the original matrix $\mathcal{A}$. The method is often combined with shifts of origin so vectors are multiplied by $(\mathcal{A} - \sigma I)^{-1}$, obtaining the eigenvalue of $\mathcal{A}$ that is closest to scalar σ and the corresponding

eigenvector. The main advantage of this method is a faster convergence due to a better separation of the eigenvalues, although it is still linear. The disadvantage is that it has to deal with the inverse, but it is not necessary to compute the inverse explicitly. Instead, a linear system of equations is solved at each iteration. This issue will be discussed few paragraphs after. Some algorithms of this method are given in Hernandez et al. 2005a.

The Rayleigh Quotient Iteration uses the same approach as that of the Inverse Iteration method, but it changes σ during the iteration process, being σ the Rayleigh quotient of the last approximate eigenvector (y), as shown in Equation 2.85. This method has two major drawbacks. First, it might converge to an eigenvalue which is not the closest to the initial σ. Second, its higher cost because it requires a factorization at every iteration. One could see some algorithm of this method in Hernandez et al. 2005a.

$$\sigma = \frac{y^* \mathcal{A} y}{y^* y} \tag{2.85}$$

Subspace Iteration methods compute approximations of the eigenvalues and eigenvectors by projecting the problem onto the subspace spanned by the columns of $\mathcal{A}^k X_0$, where X_0 is the initial set of vectors and k is the current iteration number. The reader can find more information in Hernandez et al. 2005b.

Arnoldi methods compute an orthonormal basis of the Krylov subspace of order m associated with matrix $\mathcal{A}$ and initial vector x_0. This Krylov subspace is given in Equation 2.86. Projection methods for eigenvalue problems are intended for computing a partial eigensolution, that is, given a square matrix $\mathcal{A}$ of order N, the objective is to compute a small number of eigenpairs, λ_i , x_i , $i = 1, \cdots, m$, with $m \ll N$. The Arnoldi method computes not only this orthonormal basis (V_m), but also the projected matrix H, the upper Hessenberg matrix, at the same time and in an efficient and numerically stable way.

$$\mathcal{K}_m(\mathcal{A}, x_0) = span\left\{x_0, \mathcal{A}x_0, \mathcal{A}^2 x_0, \cdots, \mathcal{A}^{m-1} x_0\right\} \tag{2.86}$$

This projection method calculates the eigenvalue problem $Hy_i = \theta_i y_i$, of order m, instead of $\mathcal{A}x_i = \lambda_i x_i$, of order N. Taking into account that ($H = V_m^T \mathcal{A} V_m$) and ($V_m^T V_m = I_m$), one concludes that the pair (λ_i,$V_m y_i$) can be taken as an approximation of the eigenpair (λ_i,x_i) of matrix $\mathcal{A}$. This method will converge very fast, if the initial vector x_0 is rich in the direction of the wanted eigen-

vectors, which is usually not the case. So, many iterations may be required, which implies a growth in storage requirements and computational time. A solution for this problem is to stop after some iterations and restart the method, by using a new initial vector computed from the recently obtained spectral approximations.

Different approaches can be used for the restart: explicit and implicit. Explicit algorithms calculate the initial vector as a linear combination of the computed eigenvectors, but it is difficult to choose the appropriate parameters. Implicit algorithms combine the Arnoldi process with the implicitly shifted QR algorithm, in which an m-step Arnoldi factorization is compacted into an (m - d)-step Arnoldi factorization, which retains the relevant eigeninformation of the large factorization. The implementation of the implicit restart in a numerically stable way is difficult, but it could be solved by using a Krylov-Schur decomposition, which is the one used in Krylov-Schur methods.

Lanczos methods are related to the Arnoldi methods, since Lanczos can be seen as a particular case of Arnoldi when the matrix is symmetric. However, Lanczos methods obtains a tridiagonal matrix, instead of the upper Hessenberg matrix (H), by a simple three-term recurrence formula. One can find further details of this formula in Hernandez et al. 2006.

Krylov-Schur methods are Arnoldi methods that use a Krylov-Schur decomposition instead of Krylov decompositions. This means that one obtains the real Schur form of $\mathcal{A}$, instead of the upper Hessenberg matrix; this real Schur form is an upper triangular matrix. The method is developed in Stewart 2002 and one can find several algorithms in Hernandez et al. 2007.

Newton methods are not specific methods for calculating eigenvalue problems, but they can be very effective, particularly if one knows a close estimation of the eigenvectors, or for non-linear eigenvalue problems (Anselone and Rall 1968), or for generalized eigenvalue problems. For the generalized eigenvalue problem of Equation 2.83, one can define the function $F = 0$, which is used in Newton methods, as in Equation 2.87. In this equation, f_{norm} is a normalization function, which is not unique. Then, one performs an iterative calculation for calculating the eigenpairs (x_{n+1}, λ_{n+1}) by using Equation 2.88 and solving the linear system of Equation 2.89, where J is the Jacobian of F. The main drawback of Newton methods is the calculation of the Jacobian and the solution of the linear system of Equation 2.89. Nonetheless, one can overcome this drawback by using Jacobian Free Newton Krylov methods (JFNK). Further details about JFNK methods can be found in Knoll and Keyes 2004.

$$0 = F(x,\lambda) = \begin{pmatrix} Ax - \lambda Bx \\ f_{norm}(x,\lambda) \end{pmatrix} = \begin{pmatrix} Ax - \lambda Bx \\ x^T x - 1 \end{pmatrix} \tag{2.87}$$

$$\begin{pmatrix} x_{n+1} \\ \lambda_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ \lambda_n \end{pmatrix} + \begin{pmatrix} \delta x_n \\ \delta \lambda_n \end{pmatrix} \tag{2.88}$$

$$J(x_n,\lambda_n) \begin{pmatrix} \delta x_n \\ \delta \lambda_n \end{pmatrix} = -F(x_n,\lambda_n) \tag{2.89}$$

Regarding the eigenvalue problems of the Neutron Diffusion and Transport Equations, several eigenvalue problems can be defined, but the eigenvalue problems most commonly used are: the λ-eigenvalue problem for commercial reactors and the α-eigenvalue problem for sub-critical systems, like accelerator driven sub-critical systems. Several authors used different projection methods for calculating multiple eigenvalues of these problems.

On the one hand, several authors applied Subspace Iteration methods for calculating the λ-eigenvalue problem of the Neutron Diffusion Equation discretized with different methods (Döring, Kalkkuhl, and Schröder 1993, Verdú et al. 1994, and Modak and Jain 1996). Another work applied Subspace Iteration methods to the Neutron Diffusion Equation for the α-eigenvalue problem (Singh et al. 2009). Moreover, other works applied Subspace Iteration methods to the α-eigenvalue problem of the Neutron Transport Equation (Gupta and Modak 2011, and Kophazi and Lathouwers 2012).

On the other hand, there are a number of works applying the Implicit Restarted Arnoldi Method to the mentioned eigenvalue problems: λ-eigenvalue problem of the Neutron Diffusion Equation (Verdú et al. 1999), α-eigenvalue problem of the Neutron Diffusion Equation (Verdu et al. 2010), λ-eigenvalue problem of the Neutron Transport Equation (Warsa et al. 2004) and α-eigenvalue problem of the Neutron Transport Equation (Lathouwers 2003, and Kophazi and Lathouwers 2012). Actually, one can find a comparison of Subspace Iteration and Implicit Restated Arnoldi methods, for the α-eigenvalue problem of the Neutron Transport Equation, in Kophazi and Lathouwers 2012. This work concluded that the Implicit Restated Arnoldi method was superior in terms of computational time.

In the last years, a number of works has been published, which use Krylov-Schur methods to calculate multiple eigenvalues of the λ-eigenvalue problem, of the Neutron Diffusion Equation (Bernal et al. 2017a, Vidal-Ferrandiz et al.

2014, Bernal et al. 2018, Theler 2013, and Carreño et al. 2017). Actually, one can find a great analysis of the application of Krylov-Schur methods for the calculation of different kinds of eigenvalue problems of the Neutron Diffusion Equation in Carreño et al. 2017.

As regards Newton methods, one can find several works applying JFNK methods for the calculation of the first eigenvalue for the Neutron Diffusion and Transport Equations (Gill and Azmy 2011, and Gill et al. 2011), but one might find only one work calculating multiple eigenvalues (Mahadevan and Ragusa 2008). On the other hand, there are several works using Block Newton Methods for calculating multiple eigenvectors (González-Pintor, Ginestar, and Verdú 2011, and González-Pintor, Ginestar, and Verdú 2013).

Another important issue in the calculation of eigenvalues is the solution of linear systems. This is particularly useful for calculating the product of the inverse of a matrix and a vector. The calculation of the inverse of a matrix is very costly, so it is better to solve a linear system as shown in Equation 2.90, where b is a vector. For small matrices $\mathcal{A}$, one can use Gaussian elimination, Cramer's rule or LU decomposition. These methods are called direct methods and they solve accurately the linear system, but they are costly for large matrices. For larges matrices, it is worth using iterative methods, but their use is limited depending on the condition number of the matrix of the system. The use of iterative or direct solvers depends on the condition number and size of the matrices. For well-conditioned and large matrices, iterative solvers are faster and require less memory resources than direct solvers. By contrast, for matrices with bad condition numbers, iterative solvers might not converge, so one should use direct solvers.

$$x = \mathcal{A}^{-1}b \rightarrow \mathcal{A}x = b \tag{2.90}$$

In linear systems, the condition number of a matrix can be seen as a bound on how inaccurate the solution x will be after applying a numerical method. This inaccuracy does not take into account the effects of round-off errors. In addition, the condition number is a property of the matrix, not the numerical method used to solve the corresponding system. One can calculate the condition number as the ratio of the largest singular value to the smallest one, as shown in Equation 2.91.

$$\mathcal{C} = \frac{\sigma_{largest}(\mathcal{A})}{\sigma_{smallest}(\mathcal{A})} \tag{2.91}$$

There is a wide variety of iterative methods for solving linear systems, but the state of the art are those combining Krylov subspace methods and a preconditioner. Krylov subspace methods might be faster than direct methods, but the rate of convergence of these methods for a particular linear system is strongly dependent on their spectrum. A preconditioner is a matrix multiplying a linear system, which alters the spectrum of the linear system, and consequently accelerates the convergence of the iterative method. The explanation of all the methods is beyond the scope of this thesis, so the author will name the iterative methods and preconditioners most used. The main methods are: Bi-Conjugate Gradient (BiCG) (Fletcher 1976), BiCGSTAB (Bi-Conjugate Gradient stabilized) (Vorst 1992), Conjugate Gradients Squared (CGS) (Sonneveld 1989), Generalized Minimal Residual (GMRES) (Saad and Schultz 1986), Generalized Conjugate Residual (GCR) (Eisenstat, Elman, and Schultz 1983), etc. The major preconditioners are: Jacobi, Incomplete LU, SOR, Additive Schwarz, etc.

One can use all the methods mentioned above for solving eigenvalue problems and linear systems, because there is a number of libraries containing different algorithms of these methods. The state of the art for solving eigenvalue problems and linear systems are SLEPc and PETSc respectively.

SLEPc, the Scalable Library for Eigenvalue Problem Computations (Hernandez, Roman, and Vidal 2005; Hernandez, Roman, and Vidal 2003; Roman et al. 2017), is a software library for the solution of large, sparse eigenproblems on parallel computers. It provides projection methods or other methods with similar properties, such as Krylov-Schur or Jacobi-Davidson. SLEPc is built on top of PETSc (Portable, Extensible Toolkit for Scientific Computation) (Balay et al. 2017) and extends it with all the functionality necessary for the solution of eigenvalue problems, which includes matrix operations and solution of linear systems. PETSc provides several algorithms for different iterative methods, but there are other libraries providing better algorithms for direct solvers, such as MUMPS (MUltifrontal Massively Parallel sparse direct Solver) (Amestoy, Duff, and L'excellent 2000).

Other important libraries for solving linear algebra problems are: BLAS (Lawson et al. 1979), LAPACK (Anderson et al. 1999), ScaLAPACK (Blackford et al. 1997), ARPACK (Lehoucq, Sorensen, and Yang 1998), etc. In addition, Matlab (Shampine and Reichelt 1997) is a software containing a number of algorithms for solving different kind of mathematical problems, and particularly, linear systems and eigenvalue problems.

The BLAS (Basic Linear Algebra Subprograms) are routines that provide standard building blocks for performing basic vector and matrix operations. Because the BLAS are efficient, portable, and widely available, they are commonly used in the development of high quality linear algebra software, LAPACK for example.

LAPACK (Linear Algebra PACKage) provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers. Dense and banded matrices are handled, but not general sparse matrices. In all areas, similar functionality is provided for real and complex matrices, in both single and double precision.

The ScaLAPACK (or Scalable LAPACK) library includes a subset of LAPACK routines redesigned for distributed memory Multiple Instruction Multiple Data (MIMD) parallel computers. It is currently written in a Single-Program-Multiple-Data style using explicit message passing for interprocessor communication. It assumes matrices are laid out in a two-dimensional block cyclic decomposition.

ARPACK (ARnoldi PACKage) is a numerical software library written in FORTRAN for solving large scale eigenvalue problems. Actually, this library was designed to compute a few eigenvalues and corresponding eigenvectors of large sparse matrices, by using the Implicitly Restarted Arnoldi Method or, in the case of symmetric matrices, the corresponding variant of the Lanczos algorithm.

2.6 Time dependent Ordinary Differential Equations

The spatially discretized time dependent Ordinary Differential Equation in the Neutron Transport and Diffusion Equations are first order Ordinary Differential Equations. One can define first order Ordinary Differential Equations as in Equation 2.92, in which $y(t)$ and $f(t, y)$ could be either functions or vectors. Since $f(t, y(t))$ could be any function, the analytical solution of Equation 2.92 is not straightforward, and therefore one has to apply numerical methods for solving this equation.

Basically, numerical methods discretize the differential term for certain time interval $[t_{i-1}, t_i]$, where one knows $y(t_{i-1})$ and wants to calculate $y(t_i)$. A

simple approach is to discretize the differential terms as in Equation 2.93. Then, one substitutes Equation 2.93 in Equation 2.92, so one obtains Equation 2.94. This method is the Euler method. Moreover, one should take into account two important issues of this approach. First, which time value t should be used in $f(t, y(t))$. If ones uses t_{i-1}, the method is called Forward Difference method, whereas if ones uses t_i, the method is called Backward Difference method. Second, the accuracy of the approach of the differential term.

$$\frac{dy(t)}{dt} = f(t, y(t)) \tag{2.92}$$

$$\frac{dy(t)}{dt} \approx \frac{y(t_i) - y(t_{i-1})}{t_i - t_{i-1}} = \frac{y_i - y_{i-1}}{t_i - t_{i-1}} \tag{2.93}$$

$$y_i = y_{i-1} + (t_i - t_{i-1}) \cdot f(t, y(t)) \tag{2.94}$$

According to the time value t used in $f(t, y(t))$, one can classify the numerical methods for first order Ordinary Differential Equations as explicit or implicit methods. Explicit methods use t_{i-1}, whereas implicit methods use t_i. Implicit methods are most stable than explicit methods, but the solution of the implicit methods is more difficult than that of the explicit ones.

One could also discretize Equation 2.92 in $[t_{i-1}, t_i]$ as in Equation 2.95. In this equation, one might use different approaches for integrating numerically $f(t, y(t))$. If one considers that f is a constant value in $[t_{i-1}, t_i]$, Equation 2.95 will be the same as Equation 2.94. However, one could use higher order methods to integrate $f(t, y(t))$. On the one hand, one could use the values of y calculated in previous steps to perform the integration. On the other hand, one could use more collocation points to perform the numerical integration. Depending on this criterion, the numerical methods are grouped in linear multistep methods or multistage methods, as discussed in Butcher 2008.

$$\int_{t_{i-1}}^{t_i} \frac{dy(t)}{dt} dt = \int_{t_{i-1}}^{t_i} f(t, y(t)) dt \rightarrow y_i = y_{i-1} + \int_{t_{i-1}}^{t_i} f(t, y(t)) dt \tag{2.95}$$

Linear multistep methods use the values of y and f calculated in the previous time intervals to obtain an accurate calculation of the integration. Thus, these methods transform Equation 2.95 into Equation 2.96, for k previous steps. In this equation, a_s and b_s are coefficients, which are calculated depending on the

linear multistep method used. As one cannot know the values of y_n for $n > i$, one has to use the information of the previous steps to increase the accuracy of the solution.

$$y_i = \sum_{s=1}^{k} a_s y_{i-s} + (t_i - t_{i-1}) \cdot \sum_{s=0}^{k} b_s f(t_{i-s}, y_{i-s}) \tag{2.96}$$

Multistage methods do not use the values of previous time intervals, but they increase the number of collocation points in $[t_{i-1}, t_i]$ to solve the integration. These methods are also known as Runge-Kutta methods. The collocation points $\bar{y}_r$ are calculated as in Equation 2.97, where $a_{r,i}$ and $b_{r,j}$ are coefficients and k is the number of collocation points. Then, one substitutes the collocation points in Equation 2.98, in which c_r are coefficients. All these coefficients depend on the kind of multistage method.

$$\bar{y}_r = f\left(a_{r,1} t_{i-1} + a_{r,2} t_i, b_{r,0} y_{i-1} + \sum_{j=1}^{r-1} b_{r,j} \bar{y}_j\right) \qquad , r = 1, ..., k \tag{2.97}$$

$$y_i = y_{i-1} + (t_i - t_{i-1}) \cdot \sum_{r=1}^{k} c_r \bar{y}_r \tag{2.98}$$

In addition, one might use a combination of linear multistep and multistage methods. These kind of methods are called general linear methods, as stated in Butcher 2008.

Another kind of method is the exponential method. This method consist in integrating analytically Equation 2.92. One can easily perform this integration, if one defines the Ordinary Differential Equation as in Equation 2.99, where y is a vector and $\mathcal{A}$ is a constant matrix. In this case, Equation 2.100 provides the analytical integration for any value of time, where y_0 is the initial condition, that is, $y_0 = y(t_0)$. In addition, one can calculate y for any time t_i with a recurrence formula, as shown in Equation 2.101.

$$\frac{dy(t)}{dt} = \mathcal{A} y(t) \tag{2.99}$$

$$y(t) = e^{\mathcal{A} \cdot (t - t_0)} y_0 \tag{2.100}$$

$$y_i = y(t_i) = e^{\mathcal{A}\cdot(t_i - t_0)} y_0 = e^{\mathcal{A}\cdot(t_i - t_{i-1})} e^{\mathcal{A}\cdot(t_{i-1} - t_0)} y_0 = e^{\mathcal{A}\cdot(t_i - t_{i-1})} y_{i-1} \quad (2.101)$$

Although the exponential method provides an analytical solution of the integration of Equation 2.92, this solution is not the analytical one, because $\mathcal{A}$ might change over time and might depend on $y(t)$. However, for a short time interval $[t_{i-1}, t_i]$, $\mathcal{A}(t)$ would not have a strong variation. Consequently, one could use the approach of Equation 2.102 and perform the analytical integration as in Equation 2.101. Thus, one can perform the analytical integration in several short time intervals as in Equation 2.103 and updating matrix $\mathcal{A}_i$ for the next intervals with Equation 2.102.

$$\mathcal{A}(t) \approx \mathcal{A}(t_i) = \mathcal{A}_i \quad (2.102)$$

$$y_i = e^{\mathcal{A}_i\cdot(t_i - t_{i-1})} y_{i-1} \qquad , i = 1, ..., n \quad (2.103)$$

There are a number of libraries and software containing different algorithms for solving first order Ordinary Differential Equations. Some of them are: PETSc (Balay et al. 2017), Trilinos (Heroux et al. 2003), SUNDIALS (Hindmarsh et al. 2005), Matlab (Shampine and Reichelt 1997), etc. The author also includes in this section SLEPc (Hernandez, Roman, and Vidal 2005), as this library includes a solver for calculating the exponential matrix, which is needed in the exponential method.

As regards the time dependent Neutron Diffusion and Transport Equations, one should apply implicit methods, because these are the only ones stable (Stacey 2007, Hébert 2009). In particular, the Backward Difference method is the most commonly used. Even with implicit methods, one should use short time steps for obtaining accurate results. However, one could use high order implicit methods to obtain accurate results with reasonable time steps, such as the high order Backward method developed in Ginestar et al. 1998. The main drawback of the implicit methods is that one has to solve large linear systems in each time step. Although one might accelerate the calculation of these linear systems, as done in Ginestar et al. 1998, one could use other methods to avoid the calculation of a large number of linear systems. For this reason, the Point Kinetics method was developed. This method assumes that the spatial distribution of the neutron flux does not change over time, but calculates the time variation of an amplitude multiplying this spatial distribution. Although this method is very simple, it is only valid for homogeneous changes of cross

sections, which do not happen in commercial nuclear reactors (Stacey 2007, Hébert 2009). An improvement of the Point Kinetics method is the Quasi-Static method (Ott 1966). This method considers that the solution is composed of a time amplitude which changes fast over time and a shape function, or shape vector, changing slowly over time. Thus, one uses two time scales for calculating the solution: a short-time scale and a long-time scale. For the short-time scale one uses the Point Kinetics method, whereas for the long-time scale one updates the shape function (or vector) with an implicit method. Several authors demonstrated the feasibility and accuracy of the Quasi-Static method (Verdú et al. 1995, Goluoglu and Dodds 2001, and Dulla, Mund, and Ravetto 2008).

Chapter 3

Steady State of the Neutron Diffusion Equation with the Finite Volume Method

3.1 Two-energy group Neutron Diffusion Equation

There are several approaches of the Neutron Diffusion Equation depending on the energy discretization. The most commonly used in commercial nuclear reactors is the two energy group discretization, without upscattering terms and with production of neutron from fissions only in the first energy group. If ones applies this approach to Equation 2.23, one obtains Equations 3.1 and 3.2 for each energy group.

$$\begin{aligned} 0 &= -\nabla \vec{J}_1(\vec{r}) - \left(\Sigma_{a,1}(\vec{r}) + \Sigma_{s,1\to 2}(\vec{r})\right)\phi_1(\vec{r}) + \\ &\quad + \frac{1}{\mathbf{k}}\left(\nu\Sigma_{f,1}(\vec{r})\phi_1(\vec{r}) + \nu\Sigma_{f,2}(\vec{r})\phi_2(\vec{r})\right) \end{aligned} \tag{3.1}$$

$$0 = -\nabla \vec{J}_2(\vec{r}) - \Sigma_{a,2}(\vec{r})\phi_2(\vec{r}) + \Sigma_{s,1\to 2}(\vec{r})\phi_1(\vec{r}) \tag{3.2}$$

One should apply numerical methods to the previous equations to obtain algebraic terms instead of the differential ones. Firstly, the geometry is discretized by using a mesh generator, in particular Gmsh (Geuzaine and Remacle 2009), as mentioned in Section 2.3. The discretized geometry will be composed of cells such as tetrahedra or hexahedra for 3D or triangles or quadrangles for 2D, containing only one homogenized material. Secondly, one applies the FVM to Equations 3.1 and 3.2, producing Equations 3.3 and 3.4 for each cell i. In these equations, n_f is the number of faces for each cell, $J_{g,i,j}$ is the face averaged value of $\vec{J}_g$ at face j of cell i, $\Sigma^i_{s,1\to2}$ is the value of $\Sigma_{s,1\to2}$ in cell i, $\nu\Sigma^i_{f,g}$ is the value of $\nu\Sigma_{f,g}$ in cell i and $\phi_{g,i}$ is the cell averaged value of ϕ_g in cell i. In addition, for 3D, S_j is the area of the face j of cell i and V_i is the volume of cell i; for 2D, S_j is the length of the face j of cell i and V_i is the area of cell i.

$$\sum_{j=1}^{n_f} S_j J_{1,i,j} + V_i \left(\Sigma^i_{a,1} + \Sigma^i_{s,1\to2}\right)\phi_{1,i} = \frac{1}{\mathbf{k}} V_i \left(\nu\Sigma^i_{f,1}\phi_{1,i} + \nu\Sigma^i_{f,2}\phi_{2,i}\right) \tag{3.3}$$

$$\sum_{j=1}^{n_f} S_j J_{2,i,j} + V_i\Sigma^i_{a,2}\phi_{2,i} - V_i\Sigma^i_{s,1\to2}\phi_{1,i} = 0 \tag{3.4}$$

Equations 3.3 and 3.4 do not contain derivatives, but $J_{g,i,j}$ is not known. These values are obtained by applying the FVM to Equation 2.15 producing Equation 3.5. In this equation, D^i_g is the value of D_g in cell i and $\vec{\nabla}\phi_{g,i,j}$ is the face averaged value of the gradient of the neutron flux at face j for cell i. Equation 3.5 does not contain derivatives, because $\vec{\nabla}\phi_{g,i,j}$ is a face averaged value.

$$J_{g,i,j} = -D^i_g \vec{\nabla}\phi_{g,i,j} \tag{3.5}$$

Since the geometry is discretized and might not be homogeneous, additional interfaces equations are required, as discussed in Section 2.1. These equations are the neutron current continuity and the flux continuity for each energy group, which are defined in Equations 2.17 and 2.64 respectively. Equation 2.64 defines the flux continuity by using the Discontinuity Factors (DFs). If one calculates the face averaged values of these equations, one obtains Equations 3.6 and 3.7, for face j, which is adjacent to cells i and l. It is important to point out that both currents of Equation 3.6 are defined with outgoing direction from their respective cells, so Equation 3.6 must contain the negative sign to accomplish the continuity of Equation 2.17. Furthermore, in Equation 3.7, $DF_{g,i,j}$ is the Discontinuity Factor for the face j of cell i for the energy group g. Moreover, one typically uses Assembly Discontinuity Factors (ADFs)

instead of DFs, as stated in Smith 1986. Therefore, one can formulate the continuity of Equation 3.7 as in Equation 3.8.

$$J_{g,i,j} = -J_{g,l,j} \rightarrow -D_g^i \vec{\nabla}\phi_{g,i,j} = D_g^l \vec{\nabla}\phi_{g,l,j} \tag{3.6}$$

$$DF_{g,i,j}\phi_{g,i,j} = DF_{g,l,j}\phi_{g,l,j} \tag{3.7}$$

$$ADF_{g,i,j}\phi_{g,i,j} = ADF_{g,l,j}\phi_{g,l,j} \tag{3.8}$$

Likewise the previous equations, one should obtain face averaged values of the boundary conditions defined in Section 2.1, so one transforms Equations 2.18-2.21 into Equations 3.9-3.12. In Equations 3.11 and 3.12, $\alpha_{g,j}$ is the albedo for the energy group g and face j. One can define these four types of boundary conditions in a general formulation as in Equation 3.13, in which $b_{j,1}$ and $b_{j,2}$ depends on the boundary condition, as shown in Table 3.1.

$$\phi_{g,i,j} = 0 \tag{3.9}$$

$$\vec{\nabla}\phi_{g,i,j} = 0 \rightarrow J_{g,i,j} = 0 \tag{3.10}$$

$$J_{g,i,j} - \frac{1-\alpha_{g,j}}{2(1+\alpha_{g,j})}\phi_{g,i,j} = 0 \tag{3.11}$$

$$J_{g,i,j} - \alpha_{g,j}\phi_{g,i,j} = 0 \tag{3.12}$$

$$b_{j,1}J_{g,i,j} + b_{j,2}\phi_{g,i,j} = 0 \tag{3.13}$$

Finally, let us consider the number of equations and the number of unknowns to solve the problem defined by N_c number of cells, N_b number of boundary faces and N_f number of inner faces. As regards the equations, for each energy group, one has N_c cells equations corresponding to Equation 3.3 or 3.4 in each cell, $2N_f$ continuity equations corresponding to Equations 3.6 and 3.8 at each inner face, N_b boundary conditions corresponding to Equation 3.13 at each boundary face. These equations provide a total number of equations of $N_c + 2N_f + N_b$

Table 3.1: General definition of boundary conditions for the Neutron Diffusion Equation

Boundary condition	$b_{j,1}$	$b_{j,2}$
Zero flux	0	1
Reflective	1	0
Albedo for LWRs	1	$-\frac{1-\alpha_{g,j}}{2(1+\alpha_{g,j})}$
Albedo for VVERs	1	$-\alpha_{g,j}$

for each energy group. With respect to the unknowns for each energy group, one has N_c values corresponding to $\phi_{g,i}$; $4N_f$ values corresponding to $J_{g,i,j}$, $J_{g,l,j}$, $\phi_{g,i,j}$,$\phi_{g,l,j}$; and $2N_b$ boundary values corresponding to $\phi_{g,i,j}$ and $J_{g,i,j}$. This gives a total number of unknowns of $N_c + 4N_f + 2N_b$ for each energy group. Therefore, one realizes that the number of unknowns is greater than the number of equations. Consequently, one has to calculate some of the unknowns from the others. What one does to solve this problem is to calculate the face averaged values $\phi_{g,i,j}$ and $J_{g,i,j}$ or $\vec{\nabla}\phi_{g,i,j}$ from the cell averaged values $\phi_{g,i}$.

3.2 Calculation of the face averaged values of fluxes and currents

In this section, the author has developed three methods for calculating the face averaged values from the cell averaged values. The first one is based on the Moving Least Squares method. The second one is a modification of the first one, based on a polynomial expansion of the neutron flux. The third one uses the same polynomial expansion, but reduces the number of unknowns and improves the condition number of the system matrices. The next subsections define each of these methods.

3.2.1 Moving Least Squares method

The method is based on the Moving Least Squares method used in a finite volume framework called *Arb*. *Arb*, as explained in Harvie 2012, solves multi-physics transport problems, in which the user defines the transport and boundary conditions by means of pseudo-mathematical expressions. Then, *Arb* transforms these expressions in algorithms written in Fortran code.

The method calculates the face averaged values as a weighted sum of the cell averaged values of the cells surrounding the face j. On the one hand, the

method calculates the same face averaged value for the two adjacent cells to face j, i and l, to guarantee the continuity: $\phi_{g,i,j} = \phi_{g,l,j}$ and $\vec{\nabla}\phi_{g,i,j} = \vec{\nabla}\phi_{g,l,j}$. As this method only calculates one face averaged value in face j for the flux and the gradient of the flux, the author will name these values $\phi_{g,j}$ and $\vec{\nabla}\phi_{g,j}$. On the other hand, the weights are calculated by means of the Moving Least Squares method explained in Cueto-Felgueroso et al. 2007. If one uses this method, one calculates $\phi_{g,j}$ and $\vec{\nabla}\phi_{g,j}$ as in Equations 3.14 and 3.15. In these equations, n is the number of cells surrounding the face j, and $k_{i,j}$ and $k_{i,j}^{grad}$ are the weights for calculating $\phi_{g,j}$ and $\vec{\nabla}\phi_{g,j}$, which are named *kernels* in Harvie 2012.

$$\phi_{g,j} = \sum_{i=1}^{n} k_{i,j}\phi_{g,i} \tag{3.14}$$

$$\vec{\nabla}\phi_{g,j} = \sum_{i=1}^{n} k_{i,j}^{grad}\phi_{g,i} \tag{3.15}$$

Therefore, this method reduces the number of unknowns for each energy group to N_c, because the unknowns are reduced to the cell averaged values of the neutron flux. Moreover, if one considers that $ADF_{g,i,j} = ADF_{g,l,j}$ and $D_g^i = D_g^l$ for the inner face j, with adjacent cells i and l, this method guarantees the continuity of Equations 3.6 and 3.8. Consequently, the number of equations for each group is reduced to $N_c + N_b$, which are the Neutron Diffusion Equations and boundary conditions. Since now there are more equations than unknowns, this method adds unknowns at the boundaries to get the same number. These extra unknowns are called *virtual fluxes* and are equivalent to $\phi_{g,i}$, but they are defined in virtual cells which are adjacent to the boundary faces. Figure 3.1 shows an example of these virtual fluxes for a 2D domain composed of 9 quadrangles, with 12 boundary faces. These virtual fluxes are taken into account in Equations 3.14 and 3.15.

As regards the assumption $ADF_{g,i,j} = ADF_{g,l,j}$, this might be true for some reactors. However, the supposition that $D_g^i = D_g^l$ may be only valid if cells i and l are composed of the same material. Since in this method the values $\vec{\nabla}\phi_{g,i,j}$ and $\vec{\nabla}\phi_{g,l,j}$ are the same, Equation 3.6 will not be accomplished if $D_g^i \neq D_g^l$. To solve that, the author of this thesis proposed to build a new diffusion coefficient for this face j: D_g^j. This value is the one used for calculating $J_{g,i,j}$ and $J_{g,l,j}$ as shown in Equation 3.16. Although there is not a clear rule

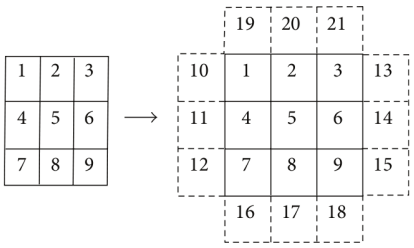


Figure 3.1: Virtual fluxes

for defining D_g^j, the author proposed in this section four types of D_g^j, defined in Equations 3.17-3.20, which are named: Cell i, Cell l, Homogenized and Linear.

$$J_{g,i,j} = -J_{g,l,j} = D_g^j \vec{\nabla}\phi_{g,j} = D_g^j \sum_{i=1}^{n} k_{i,j}^{grad} \phi_{g,i} \tag{3.16}$$

$$D_g^j = D_g^i \tag{3.17}$$

$$D_g^j = D_g^l \tag{3.18}$$

$$D_g^j = D_g^i \frac{|k_{i,j}^{grad}|}{|k_{i,j}^{grad}| + |k_{l,j}^{grad}|} + D_g^l \frac{|k_{l,j}^{grad}|}{|k_{i,j}^{grad}| + |k_{l,j}^{grad}|} \tag{3.19}$$

$$D_g^j = \frac{D_g^i D_g^l \left(|k_{i,j}^{grad}| + |k_{l,j}^{grad}| \right)}{D_g^i |k_{l,j}^{grad}| + D_g^l |k_{i,j}^{grad}|} \tag{3.20}$$

Finally, if one substitutes Equation 3.16 in Equations 3.3, 3.4 and 3.13, one obtains Equations 3.21-3.23. In Equations 3.21 and 3.22, $u_{i,j} = 1$ if cell i is the first adjacent cell to face j, or $u_{i,j} = -1$ if cell i is the second adjacent cell to face j.

$$\sum_{j=1}^{n_f} S_j u_{i,j} D_1^j \sum_{l=1}^{n} k_{l,j}^{grad} \phi_{1,l} + V_i \left(\Sigma_{a,1}^i + \Sigma_{s,1\to 2}^i \right) \phi_{1,i} = \frac{1}{\mathbf{k}} V_i \left(\nu\Sigma_{f,1}^i \phi_{1,i} + \nu\Sigma_{f,2}^i \phi_{2,i} \right) \tag{3.21}$$

$$\sum_{j=1}^{n_f} S_j u_{i,j} D_2^j \sum_{l=1}^{n} k_{l,j}^{grad} \phi_{2,l} + V_i \Sigma_{a,2}^i \phi_{2,i} - V_i \Sigma_{s,1\to 2}^i \phi_{1,i} = 0 \tag{3.22}$$

$$b_{j,1} D_g^i \sum_{l=1}^{n} k_{l,j}^{grad} \phi_{g,l} + b_{j,2} \sum_{l=1}^{n} k_{l,j} \phi_{g,l} = 0 \tag{3.23}$$

One can build the eigenvalue problem in matrix form with Equations 3.21-3.23, as shown in Equation 3.24. The eigenvector of this problem is composed of vectors Φ_g for each energy group g, which is defined in Equation 3.25. In this last equation, $\phi_{g,i}$ for $N_c + 1 \leq i \leq N_c + N_b$ are the virtual fluxes.

$$\begin{pmatrix} L_{1,1} & 0 \\ L_{2,1} & L_{2,2} \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \end{pmatrix} = \frac{1}{\mathbf{k}} \begin{pmatrix} M_{1,1} & M_{1,2} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \end{pmatrix} \tag{3.24}$$

$$\Phi_g = \begin{pmatrix} \phi_{g,1} \\ \vdots \\ \phi_{g,N_c} \\ \phi_{g,N_c+1} \\ \vdots \\ \phi_{g,N_c+N_b} \end{pmatrix} \tag{3.25}$$

The matrix $L_{1,1}$ contains the terms corresponding to $\sum_{j=1}^{n_f} S_j u_{i,j} D_1^j \sum_{l=1}^{n} k_{l,j}^{grad}$ and $V_i \left(\Sigma_{a,1}^i + \Sigma_{s,1\to 2}^i \right)$ of Equation 3.21, and $b_{j,1} D_1^i \sum_{l=1}^{n} k_{l,j}^{grad} + b_{j,2} \sum_{l=1}^{n} k_{l,j}$ of Equation 3.23.

The matrix $L_{2,2}$ contains the terms corresponding to $\sum_{j=1}^{n_f} S_j u_{i,j} D_2^j \sum_{l=1}^{n} k_{l,j}^{grad}$ and $V_i \Sigma_{a,2}^i$ of Equation 3.22, and $b_{j,1} D_2^i \sum_{l=1}^{n} k_{l,j}^{grad} + b_{j,2} \sum_{l=1}^{n} k_{l,j}$ of Equation 3.23.

The matrix $L_{2,1}$ contains the terms corresponding to $-V_i \Sigma_{s,1\to 2}^i$ of Equation 3.22.

The matrix $M_{1,1}$ contains the terms corresponding to $V_i \nu \Sigma_{f,1}^i$ of Equation 3.21.

The matrix $M_{1,2}$ contains the terms corresponding to $V_i \nu \Sigma_{f,2}^i$ of Equation 3.21.

This method was applied in the Neutron Diffusion Equation and published in Bernal et al. 2014.

3.2.2 Inter-cells polynomial expansion method

The previous method has two main drawbacks. First, one cannot include the continuity condition of Equation 3.8. Second, one has to use fine meshes to obtain accurate results. For this reason, the author developed this method.

The method is similar to the Least Squares method, since it expands the neutron flux with a polynomial expansion and tries to find out the coefficients of the expansion from the geometry information. However, there are two main differences between this method and the previous one. First, instead of evaluating the polynomials at the centroid of the cells, it uses the cell and face averaged values of the polynomials. Second, this method solves a linear system instead of solving a least square problem.

This method proposes a polynomial expansion of the neutron flux for each energy group and cell. First, the number of terms of this expansion must equal the number of equations for each energy group and cell. Figure 3.2 shows a geometry discretized into two cells, where the Neutron Diffusion Equations, boundary conditions, neutron flux continuity and current continuity are applied. In Figure 3.2, one can appreciate that the number of equations for each energy group and cell is the number of faces plus one $(n_f + 1)$, so this will be the number of terms of this expansion, which is exhibited in Equation 3.26. In this expansion, each polynomial term $p_t(x, y, z)$ is assumed to be known, and is defined in Equation 3.26 as a simple 3D monomial $x^{\alpha_t} y^{\beta_t} z^{\gamma_t}$. In contrast, the coefficients of the expansion ($a_{g,i,t}$) are unknown and will be determined by solving the eigenvalue problem. Since there are infinite polynomial combinations, one has to perform a sensitivity analysis to determine the best polynomial set.

$$\phi_{g,i}(x, y, z) = \sum_{t=1}^{n_f+1} a_{g,i,t} p_t(x, y, z) = \sum_{t=1}^{n_f+1} a_{g,i,t} x^{\alpha_t} y^{\beta_t} z^{\gamma_t} \tag{3.26}$$

Since the polynomial terms are known and depend exclusively on the geometry of cells and faces and number of faces, one can easily calculate the cell and face averaged values of the neutron flux, as shown in Equations 3.27 and 3.28.

$$\phi_{g,i} = \frac{1}{V_i} \int_{V_i} \phi_{g,i}(x, y, z) dV = \sum_{t=1}^{n_f+1} a_{g,i,t} \frac{1}{V_i} \int_{V_i} p_t(x, y, z) dV = \sum_{t=1}^{n_f+1} a_{g,i,t} \overline{p}_t^{V_i} \tag{3.27}$$

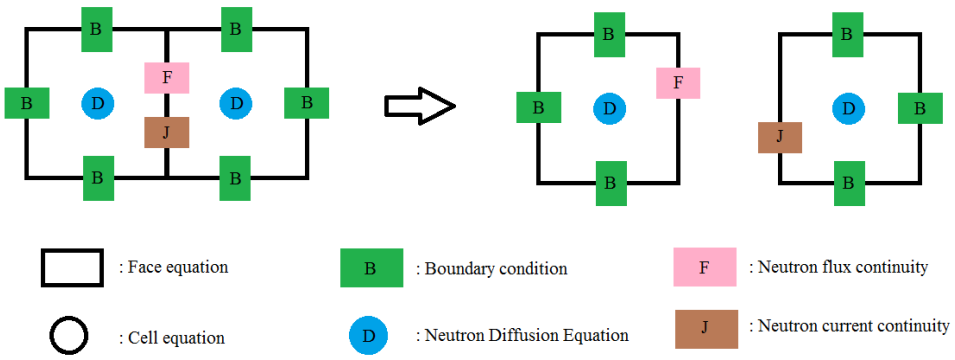


Figure 3.2: Equations applied to a discretized geometry

$$\phi_{g,i,j} = \frac{1}{S_j}\int_{S_j}\phi_{g,i}(x,y,z)dS = \sum_{t=1}^{n_f+1} a_{g,i,t}\frac{1}{S_j}\int_{S_j} p_t(x,y,z)dS = \sum_{t=1}^{n_f+1} a_{g,i,t}\overline{p}_t^{S_{i,j}} \tag{3.28}$$

Furthermore, one can calculate analytically the gradient of the neutron flux with Equations 3.29-3.32. Then, one obtains the face averaged value of the gradient of the neutron flux with Equation 3.33, where $u_{i,j,x}$, $u_{i,j,y}$ and $u_{i,j,z}$ are the direction cosines of the normal of face S_j in the outgoing direction of cell i. Therefore, one can calculate $J_{g,i,j}$ by means of Fick's Law (Equation 3.5) and this last equation, as shown in Equation 3.34.

$$\begin{aligned}\vec{\nabla}\phi_{g,i}(x,y,z) &= \sum_{t=1}^{n_f+1} a_{g,i,t}\vec{\nabla}p_t(x,y,z) \\ &= \sum_{t=1}^{n_f+1} a_{g,i,t}\left(\frac{dp_t(x,y,z)}{dx}\vec{i} + \frac{dp_t(x,y,z)}{dy}\vec{j} + \frac{dp_t(x,y,z)}{dz}\vec{k}\right)\end{aligned} \tag{3.29}$$

$$\frac{dp_t(x,y,z)}{dx} = \alpha_t x^{\alpha_t - 1} y^{\beta_t} z^{\gamma_t} \tag{3.30}$$

$$\frac{dp_t(x,y,z)}{dy} = \beta_t x^{\alpha_t} y^{\beta_t - 1} z^{\gamma_t} \tag{3.31}$$

$$\frac{dp_t(x,y,z)}{dz} = \gamma_t x^{\alpha_t} y^{\beta_t} z^{\gamma_t - 1} \tag{3.32}$$

$$\begin{aligned}
\vec{\nabla}\phi_{g,i,j} &= \frac{1}{S_j}\int_{S_j} \vec{\nabla}\phi_{g,i}(x,y,z)d\vec{S} = \\
&= \frac{1}{S_j}\int_{S_j} \vec{\nabla}\phi_{g,i}(x,y,z)\cdot\left(u_{i,j,x}\vec{i} + u_{i,j,y}\vec{j} + u_{i,j,z}\vec{k}\right)dS \\
&= \sum_{t=1}^{n_f+1} a_{g,i,t}\left(u_{i,j,x}\frac{1}{S_j}\int_{S_j}\frac{dp_t(x,y,z)}{dx}dS + u_{i,j,y}\frac{1}{S_j}\int_{S_j}\frac{dp_t(x,y,z)}{dy}dS\right. \\
&\quad \left. + u_{i,j,z}\frac{1}{S_j}\int_{S_j}\frac{dp_t(x,y,z)}{dz}dS\right) = \sum_{t=1}^{n_f+1} a_{g,i,t}\overline{\vec{\nabla}p_t}^{S_{i,j}}
\end{aligned} \tag{3.33}$$

$$J_{g,i,j} = -D_g^i \sum_{t=1}^{n_f+1} a_{g,i,t}\overline{\vec{\nabla}p_t}^{S_{i,j}} \tag{3.34}$$

Regarding the cell averaged values of the polynomials ($\overline{p}_t^{V_i}$), this method calculates them analytically. First, the method transforms the polynomials $x^{\alpha_t}y^{\beta_t}z^{\gamma_t}$, which are given in Cartesian coordinates, into a sum of polynomials with triangular or tetrahedral coordinates. Second, the method integrates the polynomials given in triangular or tetrahedral coordinates. This transformation of coordinates is shown in Equation 3.35. If the element is a triangle, ζ_i are the triangular coordinates, x_i and y_i are coordinates of the nodes, and $n = 3$, which is the number of nodes. If the element is a tetrahedron, ζ_i are the tetrahedral coordinates, x_i, y_i and z_i are coordinates of the nodes, and $n = 4$. If one substitutes Equation 3.35 in $x^{\alpha_t}y^{\beta_t}$ and applies the multinomial theorem, one obtains $x^{\alpha_t}y^{\beta_t}$ as a function of the triangular coordinates, as shown in Equation 3.36. Likewise, for tetrahedra, one obtains $x^{\alpha_t}y^{\beta_t}z^{\gamma_t}$ as a function of the tetrahedral coordinates in Equation 3.37. The advantage of using these transformations is that one can integrate the monomials $\zeta_1^i\zeta_2^j\zeta_3^k$ and $\zeta_1^i\zeta_2^j\zeta_3^k\zeta_4^l$ with Equations 3.38 and 3.39 respectively. In these equations, S is the area of the triangle and V is the volume of the tetrahedron.

$$x = \sum_{i=1}^{n} x_i \zeta_i \qquad y = \sum_{i=1}^{n} y_i \zeta_i \qquad z = \sum_{i=1}^{n} z_i \zeta_i \tag{3.35}$$

$$\begin{aligned} x^{\alpha_t} y^{\beta_t} &= \left(\sum_{i=1}^{3} x_i \zeta_i \right)^{\alpha_t} \left(\sum_{j=1}^{3} y_j \zeta_j \right)^{\beta_t} = \\ &= \sum_{i_1=0}^{\alpha_t} \sum_{i_2=0}^{i_1} \sum_{j_1=0}^{\beta_t} \sum_{j_2=0}^{j_1} \frac{\alpha_t! \cdot x_1^{\alpha_t - i_1} \cdot x_2^{i_1 - i_2} \cdot x_3^{i_2}}{(\alpha_t - i_1)! \cdot (i_1 - i_2)! \cdot i_2!} \cdot \\ &\cdot \frac{\beta_t! \cdot y_1^{\beta_t - j_1} \cdot y_2^{j_1 - j_2} \cdot y_3^{j_2}}{(\beta_t - j_1)! \cdot (j_1 - j_2)! \cdot j_2!} \cdot \zeta_1^{\alpha_t + \beta_t - i_1 - j_1} \cdot \zeta_2^{i_1 + j_1 - i_2 - j_2} \cdot \zeta_3^{i_2 + j_2} \end{aligned} \tag{3.36}$$

$$\begin{aligned} x^{\alpha_t} y^{\beta_t} z^{\gamma_t} &= \left(\sum_{i=1}^{4} x_i \zeta_i \right)^{\alpha_t} \left(\sum_{j=1}^{4} y_j \zeta_j \right)^{\beta_t} \left(\sum_{k=1}^{4} z_k \zeta_k \right)^{\gamma_t} = \\ &= \sum_{i_1=0}^{\alpha_t} \sum_{i_2=0}^{i_1} \sum_{i_3=0}^{i_2} \sum_{j_1=0}^{\beta_t} \sum_{j_2=0}^{j_1} \sum_{j_3=0}^{j_2} \sum_{k_1=0}^{\gamma_t} \sum_{k_2=0}^{k_1} \sum_{k_3=0}^{k_2} \frac{\alpha_t! \cdot x_1^{\alpha_t - i_1} \cdot x_2^{i_1 - i_2} \cdot x_3^{i_2 - i_3} \cdot x_4^{i_3}}{(\alpha_t - i_1)! \cdot (i_1 - i_2)! \cdot (i_2 - i_3)! \cdot i_3!} \cdot \\ &\cdot \frac{\beta_t! \cdot y_1^{\beta_t - j_1} \cdot y_2^{j_1 - j_2} \cdot y_3^{j_2 - j_3} \cdot y_4^{j_3}}{(\beta_t - j_1)! \cdot (j_1 - j_2)! \cdot (j_2 - j_3)! \cdot j_3!} \cdot \frac{\gamma_t! \cdot z_1^{\gamma_t - k_1} \cdot z_2^{k_1 - k_2} \cdot z_3^{k_2 - k_3} \cdot z_4^{k_3}}{(\gamma_t - k_1)! \cdot (k_1 - k_2)! \cdot (k_2 - k_3)! \cdot k_3!} \cdot \\ &\cdot \zeta_1^{\alpha_t + \beta_t + \gamma_t - i_1 - j_1 - k_1} \cdot \zeta_2^{i_1 + j_1 + k_1 - i_2 - j_2 - k_2} \cdot \zeta_3^{i_2 + j_2 + k_2 - i_3 - j_3 - k_3} \cdot \zeta_4^{i_3 + j_3 + k_3} \end{aligned} \tag{3.37}$$

$$\frac{1}{S} \int_S \zeta_1^i \zeta_2^j \zeta_3^k dS = \frac{i! \cdot j! \cdot k!}{(i + j + k + 2)!} \cdot 2 \tag{3.38}$$

$$\frac{1}{V} \int_V \zeta_1^i \zeta_2^j \zeta_3^k \zeta_4^l dV = \frac{i! \cdot j! \cdot k! \cdot l!}{(i + j + k + l + 3)!} \cdot 6 \tag{3.39}$$

With respect to quadrangles, first, one can divide each quadrangle into two triangles and calculate the cell averaged value in each triangle: $\overline{p}_t^{V_{t_1}}$ and $\overline{p}_t^{V_{t_2}}$. Second, one adds these values as in Equation 3.40, where V_{t_1} and V_{t_2} are the areas of each triangle and V_i is the area of the quadrangle. Regarding the hexahedra, first one can divide each hexahedron into five tetrahedra. If one considers the number of the nodes of the hexahedron shown in Figure 3.3, one can obtain five tetrahedra with the following nodes: the first tetrahedron is composed of nodes 1, 2, 3 and 6; the second one is composed of nodes 1, 5,

8 and 6; the third one is composed of nodes 1, 3, 8 and 6; the fourth one is composed of nodes 7, 3, 8 and 6; the fifth one is composed of nodes 1, 3, 8 and 4. Likewise the quadrangles, one can calculate the cell averaged values in each tetrahedron and add the values as in Equation 3.41. In this equation, V_i is the volume of the hexahedron, V_{t_k} is the volume of each tetrahedron and $\overline{p}_t^{V_{t_k}}$ is the cell averaged value in each tetrahedron.

$$\overline{p}_t^{V_i} = \frac{1}{V_i} \sum_{k=1}^{2} \overline{p}_t^{V_{t_k}} V_{t_k} \tag{3.40}$$

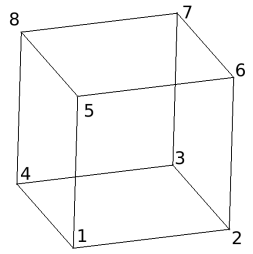


Figure 3.3: Hexahedron

$$\overline{p}_t^{V_i} = \frac{1}{V_i} \sum_{k=1}^{5} \overline{p}_t^{V_{t_k}} V_{t_k} \tag{3.41}$$

Regarding the face averaged values, one can use similar equations as the previous ones, so one can use the same algorithms. However, one has to use local axes for the faces. For tetrahedra, the faces are triangles and one can define any coordinate in the local axes. These local axes have constant values for any local Z-coordinate. In addition, one can transform any coordinate into the local axes as shown in Equations 3.42-3.44. In these equations, x_i is the X-coordinate of the node i of the triangle, y_i is the Y-coordinate of the node i of the triangle, z_i is the Z-coordinate of the node i of the triangle, x_i^{local} is the local X-coordinate of the node i of the triangle, y_i^{local} is the local Y-coordinate of the node i of the triangle, z^{local} is the local Z-coordinate of the triangle and $\langle x, x^{local} \rangle$ is the dot product between the X-axis and the local X-axis.

$$x_i^{local} = \langle x, x^{local} \rangle x_i + \langle y, x^{local} \rangle y_i + \langle z, x^{local} \rangle z_i \tag{3.42}$$

$$y_i^{local} = \left\langle x, y^{local} \right\rangle x_i + \left\langle y, y^{local} \right\rangle y_i + \left\langle z, y^{local} \right\rangle z_i \tag{3.43}$$

$$z^{local} = \left\langle x, z^{local} \right\rangle x_3 + \left\langle y, z^{local} \right\rangle y_3 + \left\langle z, z^{local} \right\rangle z_3 \tag{3.44}$$

For the local axes of the triangles (faces of the tetrahedra), one can define triangular coordinates as shown in Equations 3.45 and 3.46. If one uses these last equations, one can develop the monomial x as shown in Equation 3.47, so x is expressed as a weighted sum of ζ_i. The final expression of Equation 3.47 is similar to that obtained with tetrahedral coordinates, but with $\overline{x}_i$ and ζ_4 defined in Equation 3.48. One can use the same approach for y and z, by using $\overline{y_i}$ and $\overline{z_i}$ defined in Equations 3.49 and 3.50.

$$x^{local} = \sum_{i=1}^{3} x_i^{local} \zeta_i \tag{3.45}$$

$$y^{local} = \sum_{i=1}^{3} y_i^{local} \zeta_i \tag{3.46}$$

$$\begin{aligned}
x &= \left\langle x^{local}, x \right\rangle x^{local} + \left\langle y^{local}, x \right\rangle y^{local} + \left\langle z^{local}, x \right\rangle z^{local} = \\
&= \left\langle x^{local}, x \right\rangle \sum_{i=1}^{3} x_i^{local} \zeta_i + \left\langle y^{local}, x \right\rangle \sum_{i=1}^{3} y_i^{local} \zeta_i + \left\langle z^{local}, x \right\rangle z^{local} = \\
&= \sum_{i=1}^{3} \left(\left\langle x^{local}, x \right\rangle x_i^{local} + \left\langle y^{local}, x \right\rangle y_i^{local} \right) \zeta_i + \left\langle z^{local}, x \right\rangle z^{local} \zeta_4 = \\
&= \sum_{i=1}^{4} \overline{x}_i \zeta_i
\end{aligned} \tag{3.47}$$

$$\begin{aligned}
\overline{x}_i &= \left\langle x^{local}, x \right\rangle x_i^{local} + \left\langle y^{local}, x \right\rangle y_i^{local} \qquad , 1 \leq i \leq 3 \\
\overline{x}_4 &= \left\langle z^{local}, x \right\rangle \left(\left\langle x, z^{local} \right\rangle x_3 + \left\langle y, z^{local} \right\rangle y_3 + \left\langle z, z^{local} \right\rangle z_3 \right) \\
\zeta_4 &= 1
\end{aligned} \tag{3.48}$$

$$\begin{aligned}
\overline{y}_i &= \left\langle x^{local}, y \right\rangle x_i^{local} + \left\langle y^{local}, y \right\rangle y_i^{local} \qquad , 1 \leq i \leq 3 \\
\overline{y}_4 &= \left\langle z^{local}, y \right\rangle \left(\left\langle x, z^{local} \right\rangle x_3 + \left\langle y, z^{local} \right\rangle y_3 + \left\langle z, z^{local} \right\rangle z_3 \right) \\
\zeta_4 &= 1
\end{aligned} \tag{3.49}$$

$$\begin{aligned}
\overline{z}_i &= \left\langle x^{local}, z \right\rangle x_i^{local} + \left\langle y^{local}, z \right\rangle y_i^{local} \qquad , 1 \leq i \leq 3 \\
\overline{z}_4 &= \left\langle z^{local}, z \right\rangle \left(\left\langle x, z^{local} \right\rangle x_3 + \left\langle y, z^{local} \right\rangle y_3 + \left\langle z, z^{local} \right\rangle z_3 \right) \\
\zeta_4 &= 1
\end{aligned} \tag{3.50}$$

Then, one can develop $x^{\alpha_t} y^{\beta_t} z^{\gamma_t}$ as in Equation 3.51, which is similar to the development used for the cell averaged values. Finally, one can obtain the face averaged values by integrating the terms $\zeta_1^i \zeta_2^j \zeta_3^k$ with Equation 3.38 for the triangle.

$$\begin{aligned}
x^{\alpha_t} y^{\beta_t} z^{\gamma_t} &= \left(\sum_{i=1}^{4} \overline{x} \zeta_i \right)^{\alpha_t} \left(\sum_{j=1}^{4} \overline{y}_j \zeta_j \right)^{\beta_t} \left(\sum_{k=1}^{4} \overline{z}_k \zeta_k \right)^{\gamma_t} = \\
&= \sum_{i_1=0}^{\alpha_t} \sum_{i_2=0}^{i_1} \sum_{i_3=0}^{i_2} \sum_{j_1=0}^{\beta_t} \sum_{j_2=0}^{j_1} \sum_{j_3=0}^{j_2} \sum_{k_1=0}^{\gamma_t} \sum_{k_2=0}^{k_1} \sum_{k_3=0}^{k_2} \frac{\alpha_t! \cdot \overline{x}_1^{\alpha_t - i_1} \cdot \overline{x}_2^{i_1 - i_2} \cdot \overline{x}_3^{i_2 - i_3} \cdot \overline{x}_4^{i_3}}{(\alpha_t - i_1)! \cdot (i_1 - i_2)! \cdot (i_2 - i_3)! \cdot i_3!} \cdot \\
&\cdot \frac{\beta_t! \cdot \overline{y}_1^{\beta_t - j_1} \cdot \overline{y}_2^{j_1 - j_2} \cdot \overline{y}_3^{j_2 - j_3} \cdot \overline{y}_4^{j_3}}{(\beta_t - j_1)! \cdot (j_1 - j_2)! \cdot (j_2 - j_3)! \cdot j_3!} \cdot \frac{\gamma_t! \cdot \overline{z}_1^{\gamma_t - k_1} \cdot \overline{z}_2^{k_1 - k_2} \cdot \overline{z}_3^{k_2 - k_3} \cdot \overline{z}_4^{k_3}}{(\gamma_t - k_1)! \cdot (k_1 - k_2)! \cdot (k_2 - k_3)! \cdot k_3!} \cdot \\
&\cdot \zeta_1^{\alpha_t + \beta_t + \gamma_t - i_1 - j_1 - k_1} \cdot \zeta_2^{i_1 + j_1 + k_1 - i_2 - j_2 - k_2} \cdot \zeta_3^{i_2 + j_2 + k_2 - i_3 - j_3 - k_3}
\end{aligned} \tag{3.51}$$

Likewise, one can use the same approach for the face averaged values of the triangles. In this case, the faces are lines and one can define any coordinate in the local axes. These local axes have constant values for any local Y-coordinate. In addition, one can transform any coordinate into the local axes as shown in Equations 3.52 and 3.53.

$$x_i^{local} = \left\langle x, x^{local} \right\rangle x_i + \left\langle y, x^{local} \right\rangle y_i \tag{3.52}$$

$$y^{local} = \left\langle x, y^{local} \right\rangle x_2 + \left\langle y, y^{local} \right\rangle y_2 \tag{3.53}$$

For the local axes of the lines (faces of the triangles), one can define the local coordinates as shown in Equation 3.54. If one uses this last equation, one can

develop the monomial x as shown in Equation 3.55, so x is expressed as a weighted sum of ζ_i. The final expression of Equation 3.55 is similar to that obtained with triangular coordinates, but with $\overline{x}_i$ and ζ_3 defined in Equation 3.56. One can use the same approach for y, by using $\overline{y_i}$ defined in Equation 3.57.

$$x^{local} = \sum_{i=1}^{2} x_i^{local} \zeta_i \tag{3.54}$$

$$\begin{aligned} x &= \left\langle x^{local}, x \right\rangle x^{local} + \left\langle y^{local}, x \right\rangle y^{local} = \\ &= \left\langle x^{local}, x \right\rangle \sum_{i=1}^{2} x_i^{local} \zeta_i + \left\langle y^{local}, x \right\rangle y^{local} = \\ &= \sum_{i=1}^{2} \left\langle x^{local}, x \right\rangle x_i^{local} \zeta_i + \left\langle y^{local}, x \right\rangle y^{local} \zeta_3 = \sum_{i=1}^{3} \overline{x}_i \zeta_i \end{aligned} \tag{3.55}$$

$$\begin{aligned} \overline{x}_i &= \left\langle x^{local}, x \right\rangle x_i^{local} \qquad , 1 \leq i \leq 2 \\ \overline{x}_3 &= \left\langle y^{local}, x \right\rangle \left(\left\langle x, y^{local} \right\rangle x_2 + \left\langle y, y^{local} \right\rangle y_2 \right) \\ \zeta_3 &= 1 \end{aligned} \tag{3.56}$$

$$\begin{aligned} \overline{y}_i &= \left\langle x^{local}, y \right\rangle x_i^{local} \qquad , 1 \leq i \leq 2 \\ \overline{y}_3 &= \left\langle y^{local}, y \right\rangle \left\langle x, y^{local} \right\rangle x_2 + \left\langle y, y^{local} \right\rangle y_2 \\ \zeta_3 &= 1 \end{aligned} \tag{3.57}$$

Then, one can develop $x^{\alpha_t} y^{\beta_t}$ as in Equation 3.58, which is similar to the development used for the cell averaged values. Finally, one can obtain the face averaged values by integrating the terms $\zeta_1^i \zeta_2^j$ with Equation 3.59. In this equation, $\mathcal{L}$ is the length of the line (face of the triangle).

$$x^{\alpha_t} y^{\beta_t} = \left(\sum_{i=1}^{3} \overline{x}_i \zeta_i \right)^{\alpha_t} \left(\sum_{j=1}^{3} \overline{y}_j \zeta_j \right)^{\beta_t} =$$
$$= \sum_{i_1=0}^{\alpha_t} \sum_{i_2=0}^{i_1} \sum_{j_1=0}^{\beta_t} \sum_{j_2=0}^{j_1} \frac{\alpha_t! \cdot \overline{x}_1^{\alpha_t - i_1} \cdot \overline{x}_2^{i_1 - i_2} \cdot \overline{x}_3^{i_2}}{(\alpha_t - i_1)! \cdot (i_1 - i_2)! \cdot i_2!} \cdot$$
$$\cdot \frac{\beta_t! \cdot \overline{y}^{\beta_t - j_1} \cdot \overline{y}_2^{j_1 - j_2} \cdot \overline{y}_3^{j_2}}{(\beta_t - j_1)! \cdot (j_1 - j_2)! \cdot j_2!} \cdot \zeta_1^{\alpha_t + \beta_t - i_1 - j_1} \cdot \zeta_2^{i_1 + j_1 - i_2 - j_2} \tag{3.58}$$

$$\frac{1}{\mathcal{L}} \int_{\mathcal{L}} \zeta_1^i \zeta_2^j dL = \frac{i! \cdot j!}{(i + j + 1)!} \tag{3.59}$$

As regards the polynomial expansion, there are infinitely many possible polynomial sets, so the author restricted the sets to monomials $x^{\alpha_t} y^{\beta_t} z^{\gamma_t}$ of order 2, that is, $\alpha_t + \beta_t + \gamma_t \leq 2$. On the one hand, there are ten 3D monomials of order 2: 1, x, y, z, x^2, y^2, z^2, xy, xz and yz. On the other hand, there are six 2D monomials of order 2, by considering only x and y: 1, x, y, x^2, y^2 and xy. Depending of the number of faces of the elements (n_f) and the number of monomials (n_m), one might obtain different number of ($n_f + 1$)-combinations of these monomials; this number is named n_c. In fact, one can calculate the number of possible combinations by means of the binomial coefficient, as shown in Table 3.2. The author tested all these combinations for different geometries, and concluded that there are only few combinations providing accurate results. For triangles, there are two sets: [1, x, y, x^2] and [1, x, y, y^2]. For quadrangles, there are three sets: [1, x, y, x^2, y^2], [1, x, y, x^2, xy] and [1, x, y, y^2, xy]. For tetrahedra, there are three sets: [1, x, y, z, x^2], [1, x, y, z, y^2] and [1, x, y, z, z^2]. For hexahedra, there are three sets: [1, x, y, z, x^2, y^2, z^2], [1, x, y, z, x^2, z^2, xy] and [1, x, y, z, y^2, z^2, xy].

Table 3.2: Number of ($n_f + 1$)-combinations of the monomials

Element	Dimension	n_m	n_f	$n_c = \binom{n_m}{n_f + 1}$
Triangle	2D	6	3	15
Quadrangle	2D	6	4	6
Tetrahedron	3D	10	4	252
Hexahedron	3D	10	6	120

Once the averaged values are calculated, one can substitute these values in the equations. If one substitutes $\phi_{g,i}$ and $J_{g,i,j}$ of Equations 3.27 and 3.34 in the

Neutron Diffusion Equations of Equations 3.3 and 3.4, one obtains Equations 3.60 and 3.61.

$$\sum_{t=1}^{n_f+1} a_{1,i,t} \left(-D_1^i \sum_{j=1}^{n_f} S_j \overline{\vec{\nabla} p_t}^{S_{i,j}} + V_i \left(\Sigma_{a,1}^i + \Sigma_{s,1\to 2}^i \right) \overline{p}_t^{V_i} \right) = = \frac{1}{\mathbf{k}} \left(\sum_{t=1}^{n_f+1} a_{1,i,t} V_i \nu \Sigma_{f,1}^i \overline{p}_t^{V_i} + \sum_{t=1}^{n_f+1} a_{2,i,t} V_i \nu \Sigma_{f,2}^i \overline{p}_t^{V_i} \right) \quad (3.60)$$

$$\sum_{t=1}^{n_f+1} a_{2,i,t} \left(-D_2^i \sum_{j=1}^{n_f} S_j \overline{\vec{\nabla} p_t}^{S_{i,j}} + V_i \Sigma_{a,2}^i \overline{p}_t^{V_i} \right) - \sum_{t=1}^{n_f+1} a_{1,i,t} V_i \Sigma_{s,1\to 2}^i \overline{p}_t^{V_i} = 0 \quad (3.61)$$

For the current continuity equations, one has to substitute $J_{g,i,j}$ of Equation 3.34 in Equation 3.6, so one obtains Equation 3.62. For the neutron flux continuity, one has to substitute $\phi_{g,i,j}$ of Equation 3.28 in Equation 3.8, so one obtains Equation 3.63.

$$\sum_{t=1}^{n_f+1} a_{g,i,t} D_g^i \overline{\vec{\nabla} p_t}^{S_{i,j}} + \sum_{t=1}^{n_f+1} a_{g,l,t} D_g^l \overline{\vec{\nabla} p_t}^{S_{l,j}} = 0 \quad (3.62)$$

$$\sum_{t=1}^{n_f+1} a_{g,i,t} ADF_{g,i,j} \overline{p}_t^{S_{i,j}} - \sum_{t=1}^{n_f+1} a_{g,l,t} ADF_{g,l,j} \overline{p}_t^{S_{l,j}} = 0 \quad (3.63)$$

As regards the boundary conditions equations, one has to substitute $\phi_{g,i,j}$ and $J_{g,i,j}$ of Equations 3.28 and 3.34 in Equation 3.13, so one obtains Equation 3.64.

$$\sum_{t=1}^{n_f+1} a_{g,i,t} \left(-b_{j,1} D_g^i \overline{\vec{\nabla} p_t}^{S_{i,j}} + b_{j,2} \overline{p}_t^{S_{i,j}} \right) = 0 \quad (3.64)$$

Finally, one can build the eigenvalue problem defined in Section 3.2.1 (Equation 3.24), if one considers the following equations for each energy group: N_c Equations 3.60 or 3.61, N_f Equations 3.62, N_f Equations 3.63, and N_b Equations 3.64. Nonetheless, in this eigenvalue problem, the vectors Φ_g of the eigenvector are those of Equation 3.65.

$$\Phi_g = \begin{pmatrix} a_{g,1,1} \\ \vdots \\ a_{g,1,n_f+1} \\ a_{g,2,1} \\ \vdots \\ a_{g,2,n_f+1} \\ \vdots \\ a_{g,N_c,1} \\ \vdots \\ a_{g,N_c,n_f+1} \end{pmatrix} \tag{3.65}$$

The matrix $L_{1,1}$ contains the terms corresponding to $-D_1^i \sum_{j=1}^{n_f} S_j \overrightarrow{\nabla p_t}^{S_{i,j}} + V_i \left(\Sigma_{a,1}^i + \Sigma_{s,1\to 2}^i\right) \overline{p}_t^{V_i}$ of Equation 3.60, $D_1^i \overrightarrow{\nabla p_t}^{S_{i,j}}$ and $D_1^l \overrightarrow{\nabla p_t}^{S_{l,j}}$ of Equation 3.62, $ADF_{1,i,j} \overline{p}_t^{S_{i,j}}$ and $-ADF_{1,l,j} \overline{p}_t^{S_{l,j}}$ of Equation 3.63, and $-b_{j,1} D_1^i \overrightarrow{\nabla p_t}^{S_{i,j}} + b_{j,2} \overline{p}_t^{S_{i,j}}$ of Equation 3.64.

The matrix $L_{2,2}$ contains the terms corresponding to $-D_2^i \sum_{j=1}^{n_f} S_j \overrightarrow{\nabla p_t}^{S_{i,j}} + V_i \Sigma_{a,2}^i \overline{p}_t^{V_i}$ of Equation 3.61, $D_2^i \overrightarrow{\nabla p_t}^{S_{i,j}}$ and $D_2^l \overrightarrow{\nabla p_t}^{S_{l,j}}$ of Equation 3.62, $ADF_{2,i,j} \overline{p}_t^{S_{i,j}}$ and $-ADF_{2,l,j} \overline{p}_t^{S_{l,j}}$ of Equation 3.63, and $-b_{j,1} D_2^i \overrightarrow{\nabla p_t}^{S_{i,j}} + b_{j,2} \overline{p}_t^{S_{i,j}}$ of Equation 3.64.

The matrix $L_{2,1}$ contains the terms corresponding to $-V_i \Sigma_{s,1\to 2}^i \overline{p}_t^{V_i}$ of Equation 3.61.

The matrix $M_{1,1}$ contains the terms corresponding to $V_i \nu \Sigma_{f,1}^i \overline{p}_t^{V_i}$ of Equation 3.60.

The matrix $M_{1,2}$ contains the terms corresponding to $V_i \nu \Sigma_{f,2}^i \overline{p}_t^{V_i}$ of Equation 3.60.

This method was applied in the Neutron Diffusion Equation and published in Bernal et al. 2016b.

3.2.3 Improved inter-cells polynomial expansion method

The major downside of the previous method is that the number of equations might be extremely large, and consequently the size of the matrices of the eigenvalue problem. In fact, the number of equations for each energy group with the previous method is the number of cells times the number of faces of each cell plus one ($N_c \cdot (n_f + 1)$). Thus, reactors with large number of nodes, such as BWRs, will be modeled with a large number of equations, and consequently it will require a high computational time.

In this section, the author develops an improvement of the previous method. With the aim of accelerating the calculation, the number of equations will be reduced to $N_c + N_f$, which will reduce up to 60%, and the unknowns will be $\phi_{g,i}$ for each cell i and $J_{g,j}$ for each inner face j. The idea is to implicitly define the boundary conditions and the current continuity, and therefore the only equations will be the Neutron Diffusion Equations in each cell and the flux continuity in each inner face.

To define the current continuity implicitly, only one unknown current per each inner face j will be considered. For each inner face j, whose adjacent cells are i and l, the direction of its unknown current ($J_{g,j}$) will be from cell i to cell l, thus $J_{g,i,j} = J_{g,j}$, and $J_{g,l,j}$ will be substituted by $-J_{g,j}$, and consequently the current condition will be accomplished: $J_{g,j} = J_{g,i,j} = -J_{g,l,j}$.

With the aim of changing the unknowns from $a_{g,i,t}$ to $\phi_{g,i}$ and $J_{g,j}$, Equation 3.66 is considered from the polynomial expansion for each cell i. In this equation, $F_{g,i,j}$ is defined in Equation 3.67 and $f_{g,i,j}$ is defined in Equation 3.68, where $u_{i,j} = 1$ for cell i and $u_{i,j} = -1$ for cell l. In Equation 3.66, the first row is the calculation of $\phi_{g,i}$; the rest of rows are two different calculations depending on the face j. If the face is a boundary, the row calculates the boundary condition. If the face is an inner face, the row calculates $J_{g,j}/D_g^i$, taking into account the appropriate sign with $u_{i,j}$.

$$\begin{bmatrix} \overline{p}_1^{V_i} & \cdots & \overline{p}_{n_f+1}^{V_i} \\ f_{i,1,1} & \cdots & f_{i,1,n_f+1} \\ \vdots & & \vdots \\ f_{i,n_f,1} & \cdots & f_{i,n_f,n_f+1} \end{bmatrix} \begin{bmatrix} a_{g,i,1} \\ \vdots \\ a_{g,i,n_f+1} \end{bmatrix} = \begin{bmatrix} \phi_{g,i} \\ F_{g,i,1} \\ \vdots \\ F_{g,i,n_f} \end{bmatrix} \tag{3.66}$$

$$F_{g,i,j} = \begin{cases} 0 & \text{if face } j \text{ is a boundary face} \\ -u_{i,j}\vec{\nabla}\phi_{g,i,j} = \frac{J_{g,j}}{D_g^i} & \text{if face } j \text{ is an inner face} \end{cases} \tag{3.67}$$

$$f_{i,j,t} = \begin{cases} -b_{j,1} D_g^i \overline{\vec{\nabla} p_t}^{S_{i,j}} + b_{j,2} \overline{p}_t^{S_{i,j}} & \text{if face } j \text{ is a boundary face} \\ -u_{i,j} \overline{\vec{\nabla} p_t}^{S_{i,j}} & \text{if face } j \text{ is an inner face} \end{cases} \tag{3.68}$$

The matrix of Equation 3.66 will be called I_i and its inverse I_i^{-1}. If one inverts I_i, one obtains $a_{g,i,t}$ as a weighted sum of $\phi_{g,i}$ and $F_{g,i,k}$, as shown in Equation 3.69. In this equation, $I_i^{-1}(t,k)$ is the value of matrix I_i^{-1} for row t and column k. Although the calculation of the inverse of a matrix is not recommended, in this case it is appropriate, because the largest matrix size will be $n_f + 1$.

$$a_{g,i,t} = I_i^{-1}(t,1)\phi_{g,i} + \sum_{k=1}^{n_f} I_i^{-1}(t,1+k) F_{g,i,k} \tag{3.69}$$

Now, one can use this last equation to calculate $\phi_{g,i,j}$ by substituting Equation 3.69 in Equation 3.28, which gives Equation 3.70, where $X_{i,j,k}$ is defined in Equation 3.71. One can see that each $X_{i,j,k}$ represents the contribution of each unknown to $\phi_{g,i,j}$. It is better to multiply Equation 3.70 by D_g^i as expressed in Equation 3.72, because $D_g^i F_{g,i,j}$ is related to $J_{g,j}$, as Equation 3.73 shows.

$$\phi_{g,i,j} = X_{i,j,1}\phi_{g,i} + \sum_{t=1}^{n_f} X_{i,j,t+1} F_{g,i,t} \tag{3.70}$$

$$X_{i,j,k} = \sum_{t=1}^{n_f+1} \overline{p}_t^{S_{i,j}} I_i^{-1}(t,k) \tag{3.71}$$

$$D_g^i \phi_{g,i,j} = X_{i,j,1} D_g^i \phi_{g,i} + \sum_{t=1}^{n_f} X_{i,j,t+1} D_g^i F_{g,i,t} \tag{3.72}$$

$$D_g^i F_{g,i,j} = \begin{cases} 0 & \text{if face } j \text{ is a boundary face} \\ J_{g,j} & \text{if face } j \text{ is an inner face} \end{cases} \tag{3.73}$$

Moreover, if there is any boundary face with a boundary condition that is not reflective, one has to calculate $J_{g,i,j}$ for this face. To do so, one has to substitute $a_{g,i,t}$ of Equation 3.69 in Equation 3.34, so one obtains Equation

3.74. In this equation, $R_{i,j,k}$ is defined in Equation 3.75. One can see that each $R_{i,j,k}$ represents the contribution of each unknown to $\vec{\nabla}\phi_{g,i,j}$.

$$J_{g,i,j} = -D_g^i \vec{\nabla}\phi_{g,i,j} = R_{i,j,1} D_g^i \phi_{g,i} + \sum_{t=1}^{n_f} R_{i,j,t+1} D_g^i F_{g,i,t} \tag{3.74}$$

$$R_{i,j,k} = -\sum_{t=1}^{n_f+1} \overline{\vec{\nabla} p_t}^{S_{i,j}} I_i^{-1}(t,k) \tag{3.75}$$

As said before, the eigenvalue problem of this method is composed of two type of equations for each energy group. First, N_c Neutron Diffusion Equations corresponding to N_c cells, as those of Equations 3.76 and 3.77. In these equations, $\mathcal{F}_{in}$ represents the inner faces, whereas $\mathcal{F}_{bnr}$ represents the boundary faces with any boundary condition that is not reflective. One has obtained Equations 3.76 and 3.77 by substituting Equation 3.74 and $J_{g,i,j} = u_{i,j} J_{g,j}$ in the Neutron Diffusion Equations of Equations 3.3 and 3.4.

$$\sum_{\substack{j=1 \\ j\in\mathcal{F}_{bnr}}}^{n_f} S_j \left(R_{i,j,1} D_1^i \phi_{1,i} + \sum_{t=1}^{n_f} R_{i,j,t+1} D_1^i F_{1,i,t} \right) + \sum_{\substack{j=1 \\ j\in\mathcal{F}_{in}}}^{n_f} S_j u_{i,j} J_{1,j} + $$
$$+ V_i \left(\Sigma_{a,1}^i + \Sigma_{s,1\to 2}^i \right) \phi_{1,i} = \frac{1}{\mathbf{k}} V_i \left(\nu\Sigma_{f,1}^i \phi_{1,i} + \nu\Sigma_{f,2}^i \phi_{2,i} \right) \tag{3.76}$$

$$\sum_{\substack{j=1 \\ j\in\mathcal{F}_{bnr}}}^{n_f} S_j \left(R_{i,j,1} D_2^i \phi_{2,i} + \sum_{t=1}^{n_f} R_{i,j,t+1} D_2^i F_{2,i,t} \right) + \sum_{\substack{j=1 \\ j\in\mathcal{F}_{in}}}^{n_f} S_j u_{i,j} J_{2,j} +$$
$$+ V_i \Sigma_{a,2}^i \phi_{2,i} - V_i \Sigma_{s,1\to 2}^i \phi_{1,i} = 0 \tag{3.77}$$

The second type of equation of the eigenvalue problem is the neutron flux continuity at the inner faces. Therefore, there are N_f Equations 3.78. One has obtained this equation by substituting Equation 3.72 in Equation 3.8.

$$
\begin{aligned}
0 &= ADF_{g,i,j}\phi_{g,i,j} - ADF_{g,l,j}\phi_{g,l,j} = \frac{ADF_{g,i,j}}{D_g^i} D_g^i \phi_{g,i,j} - \frac{ADF_{g,l,j}}{D_g^l} D_g^l \phi_{g,l,j} = \\
&= \frac{ADF_{g,i,j}}{D_g^i} \left(X_{i,j,1} D_g^i \phi_{g,i} + \sum_{t=1}^{n_f} X_{i,j,t+1} D_g^i F_{g,i,t} \right) - \\
&- \frac{ADF_{g,l,j}}{D_g^l} \left(X_{l,j,1} D_g^l \phi_{g,l} + \sum_{t=1}^{n_f} X_{l,j,t+1} D_g^l F_{g,l,t} \right) \qquad (3.78)
\end{aligned}
$$

Finally, one can build the eigenvalue problem defined in Section 3.2.1 (Equation 3.24), if one considers the following equations for each energy group: N_c Equations 3.76 or 3.77 and N_f Equations 3.78. However, in this eigenvalue problem, the vectors Φ_g of the eigenvector are those of Equation 3.79.

$$
\Phi_g = \begin{pmatrix} \phi_{g,1} \\ \vdots \\ \phi_{g,N_c} \\ J_{g,1} \\ \vdots \\ J_{g,N_f} \end{pmatrix} \qquad (3.79)
$$

The matrix $L_{1,1}$ contains the terms corresponding to $S_j R_{i,j,k} D_1^i$, $S_j u_{i,j}$ and $V_i \left(\Sigma_{a,1}^i + \Sigma_{s,1\to 2}^i \right)$ of Equation 3.76, and the terms $ADF_{1,i,j} X_{i,j,1}$, $\frac{ADF_{1,i,j}}{D_1^i} X_{i,j,t+1}$, $-ADF_{1,l,j} X_{l,j,1}$ and $-\frac{ADF_{1,l,j}}{D_1^l} X_{l,j,t+1}$ of Equation 3.78.

The matrix $L_{2,2}$ contains the terms corresponding to $S_j R_{i,j,k} D_2^i$, $S_j u_{i,j}$ and $V_i \Sigma_{a,2}^i$ of Equation 3.77. The matrix also contains the terms $ADF_{2,i,j} X_{i,j,1}$, $\frac{ADF_{2,i,j}}{D_2^i} X_{i,j,t+1}$, $-ADF_{2,l,j} X_{l,j,1}$ and $-\frac{ADF_{2,l,j}}{D_2^l} X_{l,j,t+1}$ of Equation 3.78.

The matrix $L_{2,1}$ contains the terms corresponding to $-V_i \Sigma_{s,1\to 2}^i$ of Equation 3.77.

The matrix $M_{1,1}$ contains the terms corresponding to $V_i \nu \Sigma_{f,1}^i$ of Equation 3.76.

The matrix $M_{1,2}$ contains the terms corresponding to $V_i \nu \Sigma_{f,2}^i$ of Equation 3.76.

This method was applied in the Neutron Diffusion Equation and published in Bernal et al. 2016a.

In addition, one can subdivide matrices $L_{g,g'}$ and $M_{g,g'}$ as in Equations 3.80-3.82. The important issue of this subdivision is that matrix LA_g is a diagonal matrix and its dimension is N_c.

$$L_{g,g} = \begin{pmatrix} LA_g & LB_g \\ LC_g & LD_g \end{pmatrix}_{(N_c+N_f)\times(N_c+N_f)} \tag{3.80}$$

$$L_{g,g'} = \begin{pmatrix} L^{cells}_{g,g'} & 0_{N_c\times N_f} \\ 0_{N_f\times N_c} & 0_{N_f\times N_f} \end{pmatrix}, \qquad g \neq g' \tag{3.81}$$

$$M_{g,g'} = \begin{pmatrix} M^{cells}_{g,g'} & 0_{N_c\times N_f} \\ 0_{N_f\times N_c} & 0_{N_f\times N_f} \end{pmatrix} \tag{3.82}$$

3.3 Multigroup formulation

If one applies the FVM to the Neutron Diffusion Equation for any energy group, one obtains Equation 3.83.

$$\sum_{j=1}^{n_f} S_j J_{g,i,j} + V_i \left(\Sigma^i_{a,g} + \sum_{\substack{g'=1 \\ g'\neq g}}^{G} \Sigma^i_{s,g\to g'} \right) \phi_{g,i} - \\ -V_i \sum_{\substack{g'=1 \\ g'\neq g}}^{G} \Sigma^i_{s,g'\to g}\phi_{g',i} = \frac{1}{\mathbf{k}} \chi^i_g V_i \sum_{g'=1}^{G} \nu\Sigma^i_{f,g'}\phi_{g',i} \tag{3.83}$$

One can extend any of the methods of Section 3.2 to the multigroup formulation of the Neutron Diffusion Equation. Equation 3.84 shows this formulation for the method of Section 3.2.1, Equation 3.85 shows this formulation for the method of Section 3.2.2 and Equation 3.86 shows this formulation for the method of Section 3.2.3.

$$\sum_{j=1}^{n_f} S_j u_{i,j} D_g^j \sum_{l=1}^{n} k_{l,j}^{grad} \phi_{g,l} + V_i \left(\Sigma_{a,g}^i + \sum_{\substack{g'=1 \\ g' \neq g}}^{G} \Sigma_{s,g \to g'}^i \right) \phi_{g,i} -$$

$$-V_i \sum_{\substack{g'=1 \\ g' \neq g}}^{G} \Sigma_{s,g' \to g}^i \phi_{g',i} = \frac{1}{\mathbf{k}} \chi_g^i \sum_{g'=1}^{G} V_i \nu \Sigma_{f,g'}^i \phi_{g',i} \tag{3.84}$$

$$\sum_{t=1}^{n_f+1} a_{g,i,t} \left(-D_g^i \sum_{j=1}^{n_f} S_j \overrightarrow{\nabla p_t}^{S_{i,j}} + V_i \left(\Sigma_{a,g}^i + \sum_{\substack{g'=1 \\ g' \neq g}}^{G} \Sigma_{s,g \to g'}^i \right) \overline{p}_t^{V_i} \right) -$$

$$- \sum_{\substack{g'=1 \\ g' \neq g}}^{G} \sum_{t=1}^{n_f+1} a_{g',i,t} V_i \Sigma_{s,g' \to g}^i \overline{p}_t^{V_i} = \frac{1}{\mathbf{k}} \chi_g^i \sum_{g'=1}^{G} \sum_{t=1}^{n_f+1} a_{g',i,t} V_i \nu \Sigma_{f,g'}^i \overline{p}_t^{V_i} \tag{3.85}$$

$$\sum_{\substack{j=1 \\ j \in \mathcal{F}_{bnr}}}^{n_f} S_j \left(R_{i,j,1} D_g^i \phi_{g,i} + \sum_{t=1}^{n_f} R_{i,j,t+1} D_g^i F_{g,i,t} \right) + \sum_{\substack{j=1 \\ j \in \mathcal{F}_{in}}}^{n_f} S_j u_{i,j} J_{g,j} +$$

$$+V_i \left(\Sigma_{a,g}^i + \sum_{\substack{g'=1 \\ g' \neq g}}^{G} \Sigma_{s,g \to g'}^i \right) \phi_{g,i} - \sum_{\substack{g'=1 \\ g' \neq g}}^{G} V_i \Sigma_{s,g' \to g}^i \phi_{g',i} = \frac{1}{\mathbf{k}} \chi_g^i \sum_{g'=1}^{G} V_i \nu \Sigma_{f,g'}^i \phi_{g',i} \tag{3.86}$$

Next, one can apply any of the methods explained in Section 3.2 to obtain an eigenvalue problem as that of Equation 3.87. In this equation, G is the number of energy groups and u is the first group containing upscattering terms.

$$\begin{pmatrix} L_{1,1} & 0 & & \cdots & & 0 \\ \vdots & \ddots & \ddots & & & \vdots \\ L_{u-1,1} & \cdots & L_{u-1,u-1} & 0 & \cdots & 0 \\ L_{u,1} & \cdots & \cdots & L_{u,u} & \cdots & L_{u,G} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ L_{G,1} & \cdots & \cdots & L_{G,u} & \cdots & L_{G,G} \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \vdots \\ \Phi_G \end{pmatrix} = \frac{1}{\mathbf{k}} \begin{pmatrix} M_{1,1} & \cdots & M_{1,G} \\ \vdots & \vdots & \vdots \\ M_{G,1} & \cdots & M_{G,G} \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \vdots \\ \Phi_G \end{pmatrix} \tag{3.87}$$

Matrices $M_{g,g'}$ contain the fission terms, which are the same except the fission spectrum (χ_g). Therefore, $M_{g,g'}$ can be expressed as in Equation 3.88. In this equation, C_g is a matrix depending on the methods of Section 3.2: Equation 3.89 shows C_g for the method of Section 3.2.1, Equation 3.90 shows C_g for the method of Section 3.2.2 and Equation 3.91 shows C_g for the method of Section 3.2.3.

$$M_{g,g'} = C_g M_{1,g'} \tag{3.88}$$

$$C_g = diag\left(\frac{\chi_g^1}{\chi_1^1}, \cdots, \frac{\chi_g^{N_c}}{\chi_1^{N_c}}, 0, \cdots, 0\right)_{(N_c+N_b)\times(N_c+N_b)} \tag{3.89}$$

$$C_g = diag\left(\frac{\chi_g^1}{\chi_1^1}, 0, \cdots, 0, \frac{\chi_g^2}{\chi_1^2}, 0, \cdots, 0, \cdots, \frac{\chi_g^{N_c}}{\chi_1^{N_c}}, 0, \cdots, 0\right)_{(N_c+2N_f+N_b)\times(N_c+2N_f+N_b)} \tag{3.90}$$

$$C_g = diag\left(\frac{\chi_g^1}{\chi_1^1}, \cdots, \frac{\chi_g^{N_c}}{\chi_1^{N_c}}, 0, \cdots, 0\right)_{(N_c+N_f)\times(N_c+N_f)} \tag{3.91}$$

By using $M_{g,g'}$ as in Equation 3.88, one can multiply Equation 3.87 by matrix P, defined in Equation 3.92, so one obtains the eigenvalue problem of Equation 3.93. The submatrix E appearing in matrix P is the unit matrix. Matrices $\bar{L}_{g,1}$ are calculated as in Equation 3.94.

$$P = \begin{pmatrix} E & 0 & \cdots & \cdots & 0 \\ -C_2 & E & \ddots & & \vdots \\ -C_3 & 0 & E & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -C_G & 0 & \cdots & 0 & E \end{pmatrix} \tag{3.92}$$

$$\begin{pmatrix} L_{1,1} & 0 & & \cdots & & 0 \\ \vdots & \ddots & \ddots & & & \vdots \\ \bar{L}_{u-1,1} & \cdots & L_{u-1,u-1} & 0 & \cdots & 0 \\ \bar{L}_{u,1} & \cdots & \cdots & L_{u,u} & \cdots & L_{u,G} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \bar{L}_{G,1} & \cdots & \cdots & L_{G,u} & \cdots & L_{G,G} \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \vdots \\ \Phi_G \end{pmatrix} = \frac{1}{\mathbf{k}} \begin{pmatrix} M_{1,1} & \cdots & M_{1,G} \\ 0 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \vdots \\ \Phi_G \end{pmatrix} \tag{3.93}$$

$$\bar{L}_{g,1} = L_{g,1} - C_g L_{1,1} \tag{3.94}$$

This method was applied in the Neutron Diffusion Equation and published in Bernal et al. 2017b.

3.4 Solution of the Eigenvalue Problem

From Equation 3.93, the eigenvalue problem is reduced to the eigenvalue problem of Equation 3.95, which is solved by using an iterative process, for several eigenpairs. In this iterative process, Φ_1 is the iterative eigenvector and Φ_g for $g > 1$ are calculated with Equations 3.96 and 3.97. Equation 3.96 is used for calculating sequentially Φ_g, for $g = 2, \ldots, u-1$, because there is no upscattering for these energy groups. Once they are calculated, Equation 3.97 is used for calculating Φ_g, for $g \geq u$. Then, Φ_g is substituted in the right-hand-side of Equation 3.95. Equations 3.96 and 3.97 are linear systems and are obtained from Equation 3.93. It is important to highlight that the inverse of $L_{1,1}$ is not calculated in Equation 3.95, but instead the linear system is solved in each iteration of the eigensolver.

$$\mathbf{k}\Phi_1 = L_{1,1}^{-1} \sum_{g=1}^{G} M_{1,g}\Phi_g \tag{3.95}$$

$$L_{g,g}\Phi_g = -\bar{L}_{g,1}\Phi_1 - \sum_{g'=2}^{g-1} L_{g,g'}\Phi_{g'} \quad , \quad 2 \leq g < u \tag{3.96}$$

$$\begin{pmatrix} L_{u,u} & \cdots & L_{u,G} \\ \vdots & \vdots & \vdots \\ L_{G,u} & \cdots & L_{G,G} \end{pmatrix} \begin{pmatrix} \Phi_u \\ \vdots \\ \Phi_G \end{pmatrix} = \begin{pmatrix} -\bar{L}_{u,1}\Phi_1 - \sum_{g'=2}^{u-1} L_{u,g'}\Phi_{g'} \\ \vdots \\ -\bar{L}_{G,1}\Phi_1 - \sum_{g'=2}^{u-1} L_{G,g'}\Phi_{g'} \end{pmatrix} \tag{3.97}$$

On the other hand, for the method of Section 3.2.3, one can use other equations instead of Equations 3.95 and 3.96. Actually, if one uses the subdivided matrices of $L_{g,g'}$ and $M_{g,g'}$ described in Section 3.2.3, one could use a better approach. The formulation is obtained with the following example. If one uses the multigroup formulation of Equation 3.93 with two energy groups and the subdivided matrices, one obtains Equation 3.98. From this last equation, one can define the linear system of Equation 3.99, where b_2^{cells} is defined in Equation 3.100.

$$\begin{pmatrix} LA_1 & LB_1 & 0_{N_c\times N_c} & 0_{N_c\times N_f} \\ LC_1 & LD_1 & 0_{N_f\times N_c} & 0_{N_f\times N_f} \\ L_{2,1}^{cells} - C_2^{cells}\cdot LA_1 & -C_2^{cells}\cdot LB_1 & LA_2 & LB_2 \\ 0_{N_f\times N_c} & 0_{N_f\times N_f} & LC_2 & LD_2 \end{pmatrix} \begin{pmatrix} \Phi_1^{cells} \\ \Phi_1^{faces} \\ \Phi_2^{cells} \\ \Phi_2^{faces} \end{pmatrix} =$$
$$= \frac{1}{\mathbf{k}} \begin{pmatrix} M_{1,1}^{cells} & 0_{N_c\times N_f} & M_{1,2}^{cells} & 0_{N_c\times N_f} \\ 0_{N_f\times N_c} & 0_{N_f\times N_f} & 0_{N_f\times N_c} & 0_{N_f\times N_f} \\ 0_{N_c\times N_c} & 0_{N_c\times N_f} & 0_{N_c\times N_c} & 0_{N_c\times N_f} \\ 0_{N_f\times N_c} & 0_{N_f\times N_f} & 0_{N_f\times N_c} & 0_{N_f\times N_f} \end{pmatrix} \begin{pmatrix} \Phi_1^{cells} \\ \Phi_1^{faces} \\ \Phi_2^{cells} \\ \Phi_2^{faces} \end{pmatrix} \tag{3.98}$$

$$L_{2,2}\Phi_2 = \begin{pmatrix} LA_2 & LB_2 \\ LC_2 & LD_2 \end{pmatrix} \begin{pmatrix} \Phi_2^{cells} \\ \Phi_2^{faces} \end{pmatrix} = \begin{pmatrix} b_2^{cells} \\ 0_{N_f\times 1} \end{pmatrix} \tag{3.99}$$

$$b_2^{cells} = C_2^{cells}\cdot LA_1\cdot\Phi_1^{cells} + C_2^{cells}\cdot LB_1\cdot\Phi_1^{faces} - L_{2,1}^{cells}\cdot\Phi_1^{cells} \tag{3.100}$$

Since LA_g is a diagonal matrix, as explained in Section 3.2.3, one can calculate Φ_2^{cells} with Equation 3.101, which is obtained from Equation 3.99. However, one has to determine Φ_2^{faces}, to calculate the previous equation. To do so, one can obtain the linear system of Equation 3.102 from Equation 3.99. In Equation 3.102, $\overline{LD}_2$ is defined in Equation 3.103, and b_2^{faces} is defined in Equation 3.104.

$$\Phi_2^{cells} = LA_2^{-1}\left(b_2^{cells} - LB_2\cdot\Phi_2^{faces}\right) \tag{3.101}$$

$$\overline{LD}_2 \cdot \Phi_2^{faces} = b_2^{faces} \tag{3.102}$$

$$\overline{LD}_2 = LD_2 - LC_2 \cdot LA_2^{-1} \cdot LB_2 \tag{3.103}$$

$$b_2^{faces} = -LC_2 \cdot LA_2^{-1} \cdot b_2^{cells} \tag{3.104}$$

Once Φ_2^{cells} is calculated, one can use it for the eigenvalue problem. In this eigenvalue problem, one has to calculate $\mathbf{k}\Phi_1^{cells}$ and $\mathbf{k}\Phi_1^{faces}$. Likewise, one can calculate these values with Equations 3.105 and 3.106. In these equations, b_1^{cells} is defined in Equation 3.107, $\overline{LD}_1$ is defined in Equation 3.108, and b_1^{faces} is defined in Equation 3.109.

$$\mathbf{k}\Phi_1^{cells} = LA_1^{-1}\left(b_1^{cells} - LB_1 \cdot \mathbf{k}\Phi_1^{faces}\right) \tag{3.105}$$

$$\overline{LD}_1 \cdot \mathbf{k}\Phi_1^{faces} = b_1^{faces} \tag{3.106}$$

$$b_1^{cells} = \sum_{g'=1}^{G} M_{1,g'}^{cells} \Phi_{g'}^{cells} \tag{3.107}$$

$$\overline{LD}_1 = LD_1 - LC_1 \cdot LA_1^{-1} \cdot LB_1 \tag{3.108}$$

$$b_1^{faces} = -LC_1 \cdot LA_1^{-1} \cdot b_1^{cells} \tag{3.109}$$

Consequently, one can use Equations 3.105 and 3.106, instead of Equation 3.95, to solve the eigenvalue problem. Moreover, one can define equations for any energy group g by extending Equations 3.100-3.104 to any energy group. Thus, for any energy group g, which $2 \leq g < u$, one can use Equations 3.110 and 3.111, for calculating Φ_g^{cells} and Φ_g^{faces}. In the previous equations, b_g^{cells} is defined in Equation 3.112, $\overline{LD}_g$ is defined in Equation 3.113 and b_g^{faces} is defined in Equation 3.114.

$$\Phi_g^{cells} = LA_g^{-1}\left(b_g^{cells} - LB_g \cdot \Phi_g^{faces}\right) \tag{3.110}$$

$$\overline{LD}_g \cdot \Phi_g^{faces} = b_g^{faces} \tag{3.111}$$

$$b_g^{cells} = C_g^{cells} \cdot LA_1 \cdot \Phi_1^{cells} + C_g^{cells} \cdot LB_1 \cdot \Phi_1^{faces} - \sum_{g'=1}^{g-1} L_{g,g'}^{cells} \cdot \Phi_{g'}^{cells} \tag{3.112}$$

$$\overline{LD}_g = LD_g - LC_g \cdot LA_g^{-1} \cdot LB_g \tag{3.113}$$

$$b_g^{faces} = -LC_g \cdot LA_g^{-1} \cdot b_g^{cells} \tag{3.114}$$

The eigenvalue problem of Equation 3.95, or Equations 3.105 and 3.106, is solved with the Krylov-Schur algorithm (Stewart 2002), implemented in the SLEPc library and for several modes. Furthermore, one has to solve linear systems while solving the eigenvalue problem, as mentioned above. To do so, the author used iterative and direct solvers. The use of iterative or direct solvers depends on the condition number and size of the matrices, as explained in Section 2.5. For the eigenvalue problems obtained with the method of Section 3.2.3, the author used iterative solvers. Nonetheless, for the eigenvalue problems obtained with the methods of Sections 3.2.1 and 3.2.2, one has to use direct solvers, because the matrices are not well-conditioned.

As regards the iterative solvers, the author tested all the solvers included in PETSc (Balay et al. 2017), which are based on a combination of a Krylov subspace method and a preconditioner, as mentioned in Section 2.5. PETSc includes a great variety of solvers and preconditioners, but the fastest solver is Generalized Minimal Residual (GMRES) (Saad and Schultz 1986), using the ILU preconditioner. With respect to direct solvers, the author used the LU factorization implemented in MUMPS (Amestoy, Duff, and L'excellent 2000), which can also be used from PETSc.

3.5 Parallelization

The parallelization of this method involves not only running the algorithms in several Central Processing Units (CPUs), but also storing the data in different CPUs. The parallel implementation in the code developed in this thesis uses the Message Passing Interface (MPI) standard for all message-passing commu-

nication. However, the key issue of this parallelization is the use of parallel objects of PETSc, which use MPI.

Actually, the method stores the data in parallel vectors of PETSc. It is worth using these vectors because the storage and transfer of data is user-friendly. First, the vectors are defined with global and local dimensions. The global dimension is the real size of the vector, whereas the local dimension is the size of the vector stored in each CPU. In addition, one can define the local dimension explicitly for each CPU or one can let PETSc decide the most suitable dimension. PETSc does that if one defines the global dimension and set the number of CPUs. In this thesis, the method lets PETSc decide the local dimension of the vectors. Second, one can read some value in one CPU and store this value in other CPU by means of the parallel vectors of PETSc. Third, one can easily gather all the data in one or several CPUs by means of simple functions of PETSc.

With respect to the storage, this method stores six different types of data: cells, faces, nodes, materials, regions and inner faces. All these types of data are stored in all the CPUs, but each one with its local dimension. Examples of these data are:

- Cells: materials in each cell, identification number of the nodes of the cell, identification number of the faces of the cell, coordinates of the centroid of the cell, volume of the cell (or area for 2D elements), type of cell, cell and face averaged value of polynomials in the cell, and neutron flux in the cell.
- Faces: identification number of the region of the face, identification number of the adjacent cells of the face, identification number of the nodes of the face, area of the face (or length for 2D elements), type of face and vectors defining the face, such as the normal.
- Nodes: coordinates of the node.
- Materials: cross sections, fission spectra, velocities of the neutrons, diffusion coefficients and ADFs.
- Regions: identification number of the boundary conditions.
- Inner faces: neutron current at the inner face.

Furthermore, the parallelization includes all the tasks of the code: geometry pre-processing, equations discretization, eigenvalue and linear system solvers, linear algebra operations and post-processing.

The geometry pre-processing consists in storing and calculating the geometry properties. The calculation of geometry properties involves calculating the following: areas, volumes, centroids, faces normal, and adjacent cells and faces. All these operations are run in parallel computers.

The eigenvalue solver is already programmed in parallel in SLEPc. Likewise, the linear system solvers and the linear algebra operations are coded in parallel in PETSc. To use the parallel computing capabilities of these libraries, one has to define the matrices and vectors with parallel objects. In the code developed in this thesis, all the matrices and vectors are defined with parallel objects.

The post-processing step involves three major operations. First, the power calculation. Second, collapse of the results in nodal form, which is used in reactor physics. Third, calculation of axial and radial power profiles. Finally, the results are written in VTK format because of two reasons. First, VTK format can represent different unstructured meshes. Second, a lot of visualization software can read this format. In particular, the author used ParaView because it is user-friendly, open-source and is kept up to date. ParaView (Ahrens et al. 2005) is an open-source, multi-platform data analysis and visualization application. By means of ParaView, one can explore the data interactively in 3D or using ParaView's batch processing capabilities. In addition, ParaView was developed to analyze extremely large datasets using distributed memory computing resources.

The parallelization of the method was applied in the Neutron Diffusion Equation and published in Bernal et al. 2018.

Chapter 4

Modal Method for the time dependent Neutron Diffusion Equation

4.1 Modal Method

If one applies the FVM to the transient Neutron Diffusion Equation (Equations 2.13 and 2.14), one obtains Equations 4.1 and 4.2. All the terms of these equations are defined in Sections 2.1, 3.1 and 3.3, except v_g^i, $\chi_{g,k}^{del,i}$ and C_k^i; v_g^i is the velocity of the neutron, for the energy group g, in the cell i; $\chi_{g,k}^{del,i}$ is the fission spectrum of the precursors of group k, energy group g, in the cell i; and C_k^i is the cell averaged value of the concentration of precursors of group k, in the cell i, which is defined in Equation 4.3.

$$\frac{V_i}{v_g^i}\frac{d\phi_{g,i}(t)}{dt} = -\sum_{j=1}^{n_f} S_j J_{g,i,j}(t) - V_i \left(\Sigma_{a,g}^i(t) + \sum_{\substack{g'=1 \\ g'\neq g}}^{G} \Sigma_{s,g\to g'}^i(t) \right) \phi_{g,i}(t) + \\ + V_i \sum_{\substack{g'=1 \\ g'\neq g}}^{G} \Sigma_{s,g'\to g}^i(t)\phi_{g',i}(t) + (1-\beta)\chi_g^i(t) V_i \sum_{g'=1}^{G} \nu\Sigma_{f,g'}^i(t)\phi_{g',i}(t) + \\ + \sum_{k=1}^{K} \chi_{g,k}^{del,i}(t)\lambda_k V_i C_k^i(t) \tag{4.1}$$

$$V_i \frac{dC_k^i(t)}{dt} = \beta_k V_i \sum_{g'=1}^{G} \nu\Sigma_{f,g'}^i(t)\phi_{g',i}(t) - \lambda_k V_i C_k^i(t), \qquad k = 1, ..., K \tag{4.2}$$

$$C_k^i(t) = \frac{1}{V_i}\int_{V_i} C_k(\vec{r}, t) dV \tag{4.3}$$

Equation 4.1 is similar to Equation 3.83, but adding the time dependent terms. Equation 3.83 was used to obtain the eigenvalue problem in matrix form of Equation 3.87, as explained in Section 3.3. Likewise, one can obtain the matrix equations shown in Equations 4.4 and 4.5.

$$v^{-1}\frac{d\Phi}{dt} = -L\Phi + (1-\beta)M\Phi + \sum_{k=1}^{K} \lambda_k X_k^{del} C_k \tag{4.4}$$

$$X_k^{del}\frac{dC_k}{dt} = \beta_k X_k^{del} M_c \Phi - \lambda_k X_k^{del} C_k, \qquad k = 1, \cdots, K \tag{4.5}$$

In Equations 4.4 and 4.5, β, λ_k and β_k are coefficients, Φ and C_k are vectors, and the rest of terms are matrices. Equations 4.6-4.11 define these vectors and matrices: Φ is composed of vectors Φ_g; v^{-1} is a diagonal matrix composed of matrices v_g^{-1}; L is a matrix composed of matrices $L_{g,g'}$; M is a matrix composed of matrices $M_{g,g'}$; X_k^{del} is a matrix composed of matrices $X_{g,k}^{del}$; M_c is a matrix composed of matrices $M_{c,g}$.

$$\Phi = \begin{pmatrix} \Phi_1 \\ \vdots \\ \Phi_G \end{pmatrix} \tag{4.6}$$

$$v^{-1} = diag(v_1^{-1}, \cdots, v_G^{-1}) \tag{4.7}$$

$$L = \begin{pmatrix} L_{1,1} & 0 & & \cdots & & 0 \\ \vdots & \ddots & \ddots & & & \vdots \\ L_{u-1,1} & \cdots & L_{u-1,u-1} & 0 & \cdots & 0 \\ L_{u,1} & \cdots & \cdots & L_{u,u} & \cdots & L_{u,G} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ L_{G,1} & \cdots & \cdots & L_{G,u} & \cdots & L_{G,G} \end{pmatrix} \tag{4.8}$$

$$M = \begin{pmatrix} M_{1,1} & \cdots & M_{1,G} \\ \vdots & \vdots & \vdots \\ M_{G,1} & \cdots & M_{G,G} \end{pmatrix} \tag{4.9}$$

$$X_k^{del} = \begin{pmatrix} X_{1,k}^{del} \\ \vdots \\ X_{G,k}^{del} \end{pmatrix} \tag{4.10}$$

$$M_c = (M_{c,1}, \cdots, M_{c,G}) \tag{4.11}$$

Vectors Φ_g and C_k and matrices v_g^{-1}, $L_{g,g'}$, $M_{g,g'}$, $X_{g,k}^{del}$ and $M_{c,g}$ depend on the FVM. Vector Φ_g and matrices $L_{g,g'}$ and $M_{g,g'}$ are defined for the different methods in Section 3.2. Equations 4.12-4.15 define C_k, v_g^{-1}, $X_{g,k}^{del}$ and $M_{c,g}$ for the method of Section 3.2.1.

$$C_k = \begin{pmatrix} V_1 C_k^1 \\ \vdots \\ V_{N_c} C_k^{N_c} \\ 0 \\ \vdots \\ 0 \end{pmatrix}_{(N_c+N_b)\times 1} \tag{4.12}$$

$$v_g^{-1} = diag\left(\frac{V_1}{v_g^1}, \cdots, \frac{V_{N_c}}{v_g^{N_c}}, 0, \cdots, 0\right)_{(N_c+N_b)\times(N_c+N_b)} \tag{4.13}$$

$$X_{g,k}^{del} = diag\left(\chi_{g,k}^{del,1}, \cdots, \chi_{g,k}^{del,N_c}, 0, \cdots, 0\right)_{(N_c+N_b)\times(N_c+N_b)} \tag{4.14}$$

$$M_{c,g} = diag\left(\frac{1}{\chi_1^1}, \cdots, \frac{1}{\chi_1^{N_c}}, 0, \cdots, 0\right)_{(N_c+N_b)\times(N_c+N_b)} \cdot M_{1,g} \tag{4.15}$$

Equations 4.16-4.19 define C_k, v_g^{-1}, $X_{g,k}^{del}$ and $M_{c,g}$ for the method of Section 3.2.2.

$$C_k = \begin{pmatrix} V_1 C_k^1 \\ 0 \\ \vdots \\ 0 \\ V_2 C_k^2 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ V_{N_c} C_k^{N_c} \\ 0 \\ \vdots \\ 0 \end{pmatrix}_{(N_c+2N_f+N_b)\times 1} \tag{4.16}$$

$$v_g^{-1} = diag\left(\tfrac{V_1}{v_g^1}, 0, \cdots, 0, \tfrac{V_2}{v_g^2}, 0, \cdots, 0, \cdots, \tfrac{V_{N_c}}{v_g^{N_c}}, 0, \cdots, 0\right)_{(N_c+2N_f+N_b)\times(N_c+2N_f+N_b)} \tag{4.17}$$

$$X_{g,k}^{del} = diag\left(\chi_{g,k}^{del,1}, 0, \cdots, 0, \chi_{g,k}^{del,2}, 0, \cdots, 0, \cdots, \chi_{g,k}^{del,N_c}, 0, \cdots, 0\right)_{(N_c+2N_f+N_b)\times(N_c+2N_f+N_b)} \tag{4.18}$$

$$M_{c,g} = diag\left(\tfrac{1}{\chi_1^1}, 0, \cdots, 0, \tfrac{1}{\chi_1^2}, 0, \cdots, 0, \cdots, \tfrac{1}{\chi_1^{N_c}}, 0, \cdots, 0\right)_{(N_c+2N_f+N_b)\times(N_c+2N_f+N_b)} \cdot M_{1,g} \tag{4.19}$$

Equations 4.20-4.23 define C_k, v_g^{-1}, $X_{g,k}^{del}$ and $M_{c,g}$ for the method of Section 3.2.3.

$$C_k = \begin{pmatrix} V_1 C_k^1 \\ \vdots \\ V_{N_c} C_k^{N_c} \\ 0 \\ \vdots \\ 0 \end{pmatrix}_{(N_c+N_f)\times 1} \tag{4.20}$$

$$v_g^{-1} = diag\left(\frac{V_1}{v_g^1}, \cdots, \frac{V_{N_c}}{v_g^{N_c}}, 0, \cdots, 0\right)_{(N_c+N_f)\times(N_c+N_f)} \tag{4.21}$$

$$X_{g,k}^{del} = diag\left(\chi_{g,k}^{del,1}, \cdots, \chi_{g,k}^{del,N_c}, 0, \cdots, 0\right)_{(N_c+N_f)\times(N_c+N_f)} \tag{4.22}$$

$$M_{c,g} = diag\left(\frac{1}{\chi_1^1}, \cdots, \frac{1}{\chi_1^{N_c}}, 0, \cdots, 0\right)_{(N_c+N_f)\times(N_c+N_f)} \cdot M_{1,g} \tag{4.23}$$

To obtain the time distribution of the neutron flux, the modal method (Miró et al. 2002) expands the neutron flux as in Equation 4.24, where $\Phi^l(\vec{r})$, for $1 \leq l \leq M_d$, are the dominant eigenvectors associated with the M_d dominant eigenvalues of a steady state condition of the reactor, and $n_l(t)$ are their time amplitudes. Thus, any Φ^l accomplishes Equation 4.25, where L_0 and M_0 are the values of L and M for the steady state, and $\mathbf{k}_l$ is the eigenvalue associated to Φ^l.

$$\Phi(\vec{r}, t) = \sum_{l=1}^{\infty} n_l(t)\Phi^l(\vec{r}) \approx \sum_{l=1}^{M_d} n_l(t)\Phi^l(\vec{r}) \tag{4.24}$$

$$L_0\Phi^l = \frac{1}{\mathbf{k}_l} M_0\Phi^l \tag{4.25}$$

If one applies this expansion to Equations 4.4 and 4.5, one obtains Equations 4.26 and 4.27.

$$\sum_{l=1}^{M_d} v^{-1}\Phi^l \frac{dn_l(t)}{dt} = -L\sum_{l=1}^{M_d} n_l(t)\Phi^l + (1-\beta)M\sum_{l=1}^{M_d} n_l(t)\Phi^l + \sum_{k=1}^{K}\lambda_k X_k^{del} C_k \quad (4.26)$$

$$X_k^{del}\frac{dC_k}{dt} = \beta_k X_k^{del} M_c \sum_{l=1}^{M_d} n_l(t)\Phi^l - \lambda_k X_k^{del} C_k, \qquad k = 1, \cdots, K \quad (4.27)$$

Then, one can multiply the previous equations (dot product) by the adjoint eigenvectors corresponding to mode m, Φ^{m*}, so one obtains Equations 4.28 and 4.29.

$$\sum_{l=1}^{M_d} \left\langle \Phi^{m*}, v^{-1}\Phi^l \right\rangle \frac{dn_l(t)}{dt} = -\sum_{l=1}^{M_d} n_l(t) \left\langle \Phi^{m*}, L\Phi^l \right\rangle + \\ + (1-\beta)\sum_{l=1}^{M_d} n_l(t) \left\langle \Phi^{m*}, M\Phi^l \right\rangle + \sum_{k=1}^{K}\lambda_k \left\langle \Phi^{m*}, X_k^{del} C_k \right\rangle \quad (4.28)$$

$$\left\langle \Phi^{m*}, X_k^{del}\frac{dC_k}{dt} \right\rangle = \beta_k \sum_{l=1}^{M_d} n_l(t) \left\langle \Phi^{m*}, X_k^{del} M_c \Phi^l \right\rangle - \\ - \lambda_k \left\langle \Phi^{m*}, X_k^{del} C_k \right\rangle, \; k = 1, \cdots, K \quad (4.29)$$

One can rewrite the last two equations as in Equations 4.30 and 4.31. One has obtained these equations by considering that matrices L and M can be expressed as a variation of their steady state: $L = L_0 + \delta L$ and $M = M_0 + \delta M$. The rest of variables of Equations 4.30 and 4.31 are defined in Equations 4.32-4.39. It is important to highlight that ρ_m^s, in Equation 4.33, is the static reactivity.

$$\sum_{l=1}^{M_d} \Lambda_{m,l}\frac{dn_l}{dt} = (\rho_m^s - \beta)N_m n_m - \sum_{l=1}^{M_d} A_{m,l}^L n_l + (1-\beta)\sum_{l=1}^{M_d} A_{m,l}^M n_l + \sum_{k=1}^{K}\lambda_k C_{m,k} \quad (4.30)$$

$$\frac{dC_{m,k}}{dt} = \beta_k \sum_{l=1}^{M_d} A_{m,l}^{M_c,k} n_l - \lambda_k C_{m,k}, \qquad k = 1, \cdots, K \quad (4.31)$$

$$\Lambda_{m,l} = \left\langle \Phi^{m*}, v^{-1}\Phi^{l} \right\rangle \tag{4.32}$$

$$\rho_m^s = (\mathbf{k}_m - 1)/\mathbf{k}_m \tag{4.33}$$

$$N_m = \left\langle \Phi^{m*}, M_0\Phi^{m} \right\rangle \tag{4.34}$$

$$A_{m,l}^{L} = \left\langle \Phi^{m*}, \delta L\Phi^{l} \right\rangle \tag{4.35}$$

$$A_{m,l}^{M} = \left\langle \Phi^{m*}, \delta M\Phi^{l} \right\rangle \tag{4.36}$$

$$C_{m,k} = \left\langle X_k^{del\,T}\Phi^{m*}, C_k \right\rangle \tag{4.37}$$

$$\frac{dC_{m,k}}{dt} = \left\langle X_k^{del\,T}\Phi^{m*}, \frac{dC_k}{dt} \right\rangle \tag{4.38}$$

$$A_{m,l}^{M_c,k} = \left\langle X_k^{del\,T}\Phi^{m*}, M_c\Phi^{l} \right\rangle \tag{4.39}$$

Then, if one considers all the values of m and l, ranging from 1 to M_d, and all the values of k, ranging from 1 to K, one obtains the system of differential equations of Equation 4.40, where I is the identity matrix of dimension M_d.

$$\frac{d}{dt}\begin{pmatrix} n_1 \\ n_2 \\ \vdots \\ n_{M_d} \\ C_{1,1} \\ \vdots \\ C_{M_d,1} \\ \vdots \\ C_{1,K} \\ \vdots \\ C_{M_d,K} \end{pmatrix} = \begin{pmatrix} \Lambda^{-1}\left((\rho^s - \beta I)N - A^L + (1-\beta)A^M\right) & \lambda_1\Lambda^{-1} & \cdots & \lambda_K\Lambda^{-1} \\ \beta_1 A^{M_c,1} & -\lambda_1 I & 0 & 0 \\ \vdots & 0 & \ddots & 0 \\ \beta_K A^{M_c,K} & 0 & 0 & -\lambda_K I \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \\ \vdots \\ n_{M_d} \\ C_{1,1} \\ \vdots \\ C_{M_d,1} \\ \vdots \\ C_{1,K} \\ \vdots \\ C_{M_d,K} \end{pmatrix} \tag{4.40}$$

If M_d is not a large number, the dimension of the system of Equation 4.40 is much smaller ($M_d + M_d \cdot K$) than that of Equations 4.4 and 4.5. Moreover, Equation 4.40 is a stiff system, so the author decided to use the exponential matrix method to solve this equation, since it is an analytical method. With the aim of explaining the exponential matrix method applied to Equation 4.40, the author will write this equation in a simple form, as in Equation 4.41. As explained in Section 2.6, one should integrate Equation 4.41 in time intervals $[t_{i-1}, t_i]$ changing matrix $\mathcal{A}$, so one can evaluate accurately $\mathcal{N}$. For certain time interval $[t_{i-1}, t_i]$, with matrix $\mathcal{A}_i$, one can obtain $\mathcal{N}_i$ by means of Equation 4.42, as stated in Section 2.6.

$$\frac{d\mathcal{N}}{dt} = \mathcal{A}\mathcal{N} \tag{4.41}$$

$$\mathcal{N}_i = e^{\mathcal{A}_i \cdot (t_i - t_{i-1})} \mathcal{N}_{i-1} \tag{4.42}$$

For calculating the exponential matrix, the author used the method implemented in SLEPc, which is a matrix function solver reducing the system matrix and applying the Padé method with scaling and squaring to the reduced matrix. Since one could expand $e^{\mathcal{A}_i \cdot (t_i - t_{i-1})} \mathcal{N}_{i-1}$ as the Taylor series of Equation 4.43, one can approximate $e^{\mathcal{A}_i \cdot (t_i - t_{i-1})} \mathcal{N}_{i-1}$ as an element of the Krylov subspace of matrix $\mathcal{A}_i \cdot (t_i - t_{i-1})$, vector $\mathcal{N}_{i-1}$ and dimension n. Therefore, SLEPc performs the Arnoldi decomposition of Equation 4.44, where the columns of $\mathcal{V}_n$ form an orthogonal basis of the previous Krylov subspace, and H_n is the Hessenberg matrix. Then, SLEPc computes $\widetilde{\mathcal{N}_i}$, which is the approximation of $\mathcal{N}_i$, as in Equation 4.45, where $b = ||N_{i-1}||_2$ and e_1 is the first coordinate vector. In addition, SLEPc computes the exponential of H_n with the Padé method with scaling and squaring (Higham 2009).

$$e^{\mathcal{A}_i \cdot (t_i - t_{i-1})} \mathcal{N}_{i-1} = \mathcal{N}_{i-1} + \frac{\mathcal{A}_i \cdot (t_i - t_{i-1})}{1!} \mathcal{N}_{i-1} + \frac{(\mathcal{A}_i \cdot (t_i - t_{i-1}))^2}{2!} \mathcal{N}_{i-1} + \cdots \tag{4.43}$$

$$\mathcal{A}_i \cdot (t_i - t_{i-1}) \mathcal{V}_n = \mathcal{V}_{n+1} H_n \tag{4.44}$$

$$\widetilde{\mathcal{N}_i} = b \mathcal{V}_n e^{H_n} e_1 \tag{4.45}$$

Since Equation 4.42 is a recurrence formula and one knows $\mathcal{A}_i$ for each time interval $[t_{i-1}, t_i]$, one can calculate any $\mathcal{N}_i$, if one knows the initial condition $\mathcal{N}_0$. Fortunately, the initial condition is the steady state of Equations 4.30 and 4.31. In particular, the solution of the steady state of the Neutron Diffusion Equation is the fundamental mode, that is, the eigenvector Φ^1 associated to $\mathbf{k}_1$. Therefore, if one considers the previous statement and the modal expansion of $\Phi(\vec{r}, t)$, which is in Equation 4.24, for time $t = 0$, one realizes that the initial conditions for n_l have to be: $n_1(0) = 1$ and $n_l(0) = 0$, for $2 \leq l \leq M_d$. For calculating the initial conditions of $C_{m,k}$, one has to consider that in the steady state the fission terms are divided by $\mathbf{k}_1$, as shown in Equation 4.46. Consequently, one can calculate the steady state solution of $C_{m,k}$ with Equation 4.47, which one obtains by setting to 0 the derivatives of Equation 4.31 and dividing the fission terms by $\mathbf{k}_1$. Finally, if one applies the initial conditions of n_l to Equation 4.47, one obtains the initial conditions of $C_{m,k}(0)$, as shown in Equation 4.48. In conclusion, Equation 4.49 shows $\mathcal{N}_0$.

$$L_0\Phi^1 = \frac{1}{\mathbf{k}_1} M_0\Phi^1 \tag{4.46}$$

$$0 = \frac{1}{\mathbf{k}_1}\beta_k \sum_{l=1}^{M_d} A_{m,l}^{M_c,k} n_l(0) - \lambda_k C_{m,k}(0), \qquad k = 1, \cdots, K \tag{4.47}$$

$$C_{m,k}(0) = \frac{1}{\mathbf{k}_1}\frac{\beta_k}{\lambda_k} A_{m,1}^{M_c,k}, \qquad k = 1, \cdots, K, \qquad m = 1, \cdots, M_d \tag{4.48}$$

$$\mathcal{N}_0 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ \frac{1}{\mathbf{k}_1}\frac{\beta_1}{\lambda_1} A_{1,1}^{M_c,1} \\ \vdots \\ \frac{1}{\mathbf{k}_1}\frac{\beta_1}{\lambda_1} A_{M_d,1}^{M_c,1} \\ \vdots \\ \frac{1}{\mathbf{k}_1}\frac{\beta_K}{\lambda_K} A_{1,1}^{M_c,K} \\ \vdots \\ \frac{1}{\mathbf{k}_1}\frac{\beta_K}{\lambda_K} A_{M_d,1}^{M_c,K} \end{pmatrix} \tag{4.49}$$

4.2 Adjoint calculation

The modal method needs the adjoint eigenvectors Φ^{m*} to obtain the reduced matrix. The adjoint eigenvectors are the eigenvectors of the adjoint eigenvalue problem. Taking into account the eigenvalue problem of Equation 3.87, which one can simplify in Equation 4.50, the adjoint eigenvalue problem is defined in Equation 4.51. Since matrices L and M are real, $L^* = L^T$ and $M^* = M^T$.

$$L\Phi = \frac{1}{\mathbf{k}} M\Phi \tag{4.50}$$

$$L^*\Phi^* = \frac{1}{\mathbf{k}} M^*\Phi^* \rightarrow L^T\Phi^* = \frac{1}{\mathbf{k}} M^T\Phi^* \tag{4.51}$$

An interesting property of the adjoint eigenvectors is the biorthogonal property with respect to the eigenvectors, which is shown in Equation 4.52, where $\delta_{m,l}$ is the Kronecker delta. Moreover, the adjoint eigenvalues are the same for the forward and adjoint eigenvalue problems.

$$\left\langle \Phi^{m*}, L\Phi^{l} \right\rangle = \frac{1}{\mathbf{k_l}} \left\langle \Phi^{m*}, M\Phi^{l} \right\rangle = \frac{1}{\mathbf{k_l}} \delta_{m,l} N_l \tag{4.52}$$

Unfortunately, the adjoint eigenvectors might be different from the forward eigenvectors. Furthermore, there is not a mathematical formula for obtaining the adjoint eigenvectors from the forward ones. Thus, one has to solve the adjoint eigenvalue problem to obtain the adjoint eigenvectors, which might require the same computational resources. These eigenvalue calculations might be computationally costly for large matrices.

Therefore, it would be better to develop a method for obtaining an estimation of the adjoint eigenvectors from the forward ones. Actually, other authors developed this kind of methods, such as Döring, Kalkkuhl, and Schröder 1993. This method was used in other modal methods, such as Miró et al. 2002. However, this method seems to work only for the Neutron Diffusion Equation with two energy groups, without upscattering and with $\chi_1 = 1$. In this case, one can combine the eigenvalue problem of Equation 3.95 and Equation 3.96 to obtain the eigenvalue problem of Equation 4.53. The adjoint eigenvalue problem for this case is given in Equation 4.54, which one can transform into Equation 4.55. From Equations 4.53 and 4.55, one can obtain Equation 4.56. In addition, one should also consider the eigenvalue problem of matrix A^T,

which is the transpose of matrix A. This eigenvalue problem is defined in Equation 4.57, which has the same eigenvalues as the forward and adjoint eigenvalue problems, but the eigenvector is $\Phi_1^{A^T}$. This eigenvector is important due to two reasons. First, the method of Döring, Kalkkuhl, and Schröder 1993 estimates $\Phi_1^{A^T}$ from the forward eigenvectors. Second, one can calculate the adjoint eigenvectors from $\Phi_1^{A^T}$, as shown in Equation 4.58. One can obtain this equation from Equations 4.55-4.57

$$\mathbf{k}\Phi_1 = L_{1,1}^{-1}\left(M_{1,1} - M_{1,2}L_{2,2}^{-1}L_{2,1}\right)\Phi_1 = A\Phi_1 \tag{4.53}$$

$$\begin{pmatrix} L_{1,1}^T & L_{2,1}^T \\ 0 & L_{2,2}^T \end{pmatrix}\begin{pmatrix} \Phi_1^* \\ \Phi_2^* \end{pmatrix} = \frac{1}{\mathbf{k}}\begin{pmatrix} M_{1,1}^T & 0 \\ M_{1,2}^T & 0 \end{pmatrix}\begin{pmatrix} \Phi_1^* \\ \Phi_2^* \end{pmatrix} \tag{4.54}$$

$$\mathbf{k}\Phi_1^* = {L_{1,1}^T}^{-1}\left(M_{1,1}^T - L_{2,1}^T{L_{2,2}^T}^{-1}M_{1,2}^T\right)\Phi_1^* = A^*\Phi_1^* \tag{4.55}$$

$$A^T L_{1,1}^T = L_{1,1}^T A^* \tag{4.56}$$

$$\mathbf{k}\Phi_1^{A^T} = A^T\Phi_1^{A^T} \tag{4.57}$$

$$\Phi_1^* = {L_{1,1}^T}^{-1}\Phi_1^{A^T} \tag{4.58}$$

The key issue of the method proposed in Döring, Kalkkuhl, and Schröder 1993 is that $\Phi_1^{A^T}$ is within the subspace of the forward eigenvectors Φ_1. Therefore, one can calculate $\Phi_1^{A^T}$ as a linear combination of the vectors of this subspace, as shown in Equation 4.59. In this equation, ${\Phi_1^l}^{A^T}$ is the eigenvector l of A^T, X is a matrix whose columns are orthonormal vectors of the subspace defined by the forward eigenvectors Φ_1^l, and U is the matrix containing the coefficients for calculating ${\Phi_1^l}^{A^T}$ as a linear combination of the columns of X. Since X is orthonormal, $X^TX = I$, where I is the unit matrix.

$$\left[{\Phi_1^1}^{A^T} \cdots {\Phi_1^{M_d}}^{A^T}\right] = XU \tag{4.59}$$

If one substitutes the previous equation in the eigenvalue problem of Equation 4.60, one obtains Equation 4.61. Then, one can multiply this equation by X^T, so one obtains Equation 4.62. In this equation, one uses $X^T X = I$ and defines $B = X^T A^T X$ to obtain the eigenvalue problem of Equation 4.63. The size of matrix B is M_d, which is much smaller than the size of matrix A^T, so one can perform a fast calculation of the eigenvalue problem of Equation 4.63. In particular, one solves M_d eigenvalue problems, one for each column of U, as shown in Equation 4.64. The solution of this eigenvalue problem is U, which one uses to calculate ${\Phi_1^l}^{A^T}$ with Equation 4.59. Once ${\Phi_1^l}^{A^T}$ are calculated, one can calculate ${\Phi_1^l}^*$ with Equation 4.65.

$$A^T \left[{\Phi_1^1}^{A^T} \cdots {\Phi_1^{M_d}}^{A^T} \right] = \left[{\Phi_1^1}^{A^T} \cdots {\Phi_1^{M_d}}^{A^T} \right] \cdot diag\left(\mathbf{k}_1, \cdots, \mathbf{k}_{M_d}\right) \tag{4.60}$$

$$A^T XU = XU \cdot diag\left(\mathbf{k}_1, \cdots, \mathbf{k}_{M_d}\right) \tag{4.61}$$

$$X^T A^T XU = X^T XU \cdot diag\left(\mathbf{k}_1, \cdots, \mathbf{k}_{M_d}\right) \tag{4.62}$$

$$BU = U \cdot diag\left(\mathbf{k}_1, \cdots, \mathbf{k}_{M_d}\right) \tag{4.63}$$

$$BU_l = \mathbf{k}_l U_l \qquad , 1 \leq l \leq M_d \tag{4.64}$$

$${\Phi_1^l}^* = {L_{1,1}^T}^{-1} {\Phi_1^l}^{A^T} = {L_{1,1}^T}^{-1} XU_l \qquad , 1 \leq l \leq M_d \tag{4.65}$$

In this section, the author develops a simpler, faster and more accurate method than the one developed in Döring, Kalkkuhl, and Schröder 1993, as shown in Section 6.7. Although this method is not exact, it is accurate enough and requires much less computational resources than solving the adjoint eigenvalue problem.

This method for calculating the adjoint is based on the Power Iteration method, which is explained in Section 2.5. The Power Iteration method is based on the product of the matrix by a vector, which should be an estimation of the final solution. If this vector is close to the eigenvector, the method will converge fast and one might multiply the matrix only few times.

The proposed method multiplies only once the adjoint matrix and the initial estimation, for each mode. The proposed estimation is Φ_1^l for calculating Φ^{l*} for each mode l, which is the vector containing all the vectors ${\Phi_g^l}^*$, for each energy group g. In addition, the method includes a reorthogonalization for conserving the biorthogonal property of the forward and adjoint eigenvectors.

Although the estimation vector is Φ_1^l, which only contains the values for the first energy group, the method calculates the values of the adjoint eigenvectors for all the energy groups, that is, ${\Phi_g^l}^*$. Actually, one obtains the forward eigenvectors for all the energy groups in the same way, as explained in Section 3.4.

Now, the author will obtain different equations for calculating the adjoint eigenvalue problem, as those developed in Section 3.4. The author obtained the equations of Section 3.4 from the multigroup formulation developed in Section 3.3. In fact, one does not obtain these equations from the original eigenvalue problem (Equation 3.87 or 4.50), but from this eigenvalue problem multiplied by matrix P (Equation 3.92), which is that of Equation 3.93. One can write this equation with a simple formulation, as in Equation 4.66. The main advantage of solving Equation 4.66 is that it requires less computational resources than the original one. Consequently, the author will use the same approach for the adjoint eigenvalue problem.

$$PL\Phi = \frac{1}{\mathbf{k}}PM\Phi \rightarrow \overline{L}\Phi = \frac{1}{\mathbf{k}}\overline{M}\Phi \tag{4.66}$$

Although the eigenvalue problems of Equations 4.50 and 4.66 have the same eigenvectors, the adjoint ones do not, as one can see in Equations 4.51 and 4.67. However, one realizes that the adjoint eigenvectors of Equations 4.51 and 4.67, which are Φ^* and Ψ^* respectively, are related as shown in Equation 4.68. Therefore, first, the method will calculate Ψ^* because it is easier. Second, the method calculates Φ^* with Equation 4.68, because the modal method uses Φ^*.

$$\overline{L}^T\Psi^* = \frac{1}{\mathbf{k}}\overline{M}^T\Psi^* \rightarrow L^TP^T\Psi^* = \frac{1}{\mathbf{k}}M^TP^T\Psi^* \tag{4.67}$$

$$\begin{pmatrix} \Phi_1^* \\ \Phi_2^* \\ \vdots \\ \Phi_G^* \end{pmatrix} = \Phi^* = P^T \Psi^* = \begin{pmatrix} E & -C_2 & \cdots & -C_G \\ 0 & E & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & E \end{pmatrix} \begin{pmatrix} \Psi_1^* \\ \Psi_2^* \\ \vdots \\ \Psi_G^* \end{pmatrix} = \begin{pmatrix} \Psi_1^* - \sum_{g=2}^{G} C_g \Psi_g^* \\ \Psi_2^* \\ \vdots \\ \Psi_G^* \end{pmatrix} \tag{4.68}$$

Likewise the forward eigenvalue problem explained in Section 3.4, one can reduce the adjoint eigenvalue problem of Equation 4.67 to that of Equation 4.69. In this equation, one calculates $\mathbf{k}\Psi_1^*$ from the estimation of Ψ_1^*: ${\Psi_1^l}^* = \Phi_1^l$, for $1 \leq l \leq M_d$. In addition, one needs $\mathbf{k}\Psi_g^*$, for $g \geq 2$, to calculate $\mathbf{k}\Psi_1^*$. To calculate $\mathbf{k}\Psi_g^*$, for $g \geq 2$, one starts solving the linear system of Equation 4.70, which contains the upscattering terms, because Ψ_1^* is known. Then, one solves sequentially the linear systems of Equation 4.71, from $g = u - 1$ to $g = 2$. Finally, one substitutes $\mathbf{k}\Psi_g^*$, for $g \geq 2$, in Equation 4.69, so one obtains $\mathbf{k}\Psi_1^*$. Since the final solution is $\mathbf{k}\Phi^*$ instead of $\mathbf{k}\Psi^*$, one has to substitute the calculated $\mathbf{k}\Psi_g^*$, for $g \geq 1$, in Equation 4.72. This process is performed only once, so there are not iterations.

$$\mathbf{k}\Psi_1^* = {L_{1,1}^T}^{-1} \left(M_{1,1}^T \Psi_1^* - \sum_{g=2}^{G} \bar{L}_{g,1}^T \left(\mathbf{k}\Psi_g^* \right) \right) \tag{4.69}$$

$$\begin{pmatrix} L_{u,u}^T & \cdots & L_{G,u}^T \\ \vdots & \vdots & \vdots \\ L_{u,G}^T & \cdots & L_{G,G}^T \end{pmatrix} \begin{pmatrix} \mathbf{k}\Psi_u^* \\ \vdots \\ \mathbf{k}\Psi_G^* \end{pmatrix} = \begin{pmatrix} M_{1,u}^T \Psi_1^* \\ \vdots \\ M_{1,G}^T \Psi_1^* \end{pmatrix} \tag{4.70}$$

$$L_{g,g}^T \cdot \left(\mathbf{k}\Psi_g^* \right) = M_{1,g}^T \Psi_1^* - \sum_{g'=g+1}^{G} L_{g',g}^T \left(\mathbf{k}\Psi_{g'}^* \right) \quad , \quad u > g \geq 2 \tag{4.71}$$

$$\mathbf{k}\Phi^* = \begin{pmatrix} \mathbf{k}\Phi_1^* \\ \mathbf{k}\Phi_2^* \\ \vdots \\ \mathbf{k}\Phi_G^* \end{pmatrix} = \begin{pmatrix} \mathbf{k}\Psi_1^* - \sum_{g=2}^{G} C_g \mathbf{k}\Psi_g^* \\ \mathbf{k}\Psi_2^* \\ \vdots \\ \mathbf{k}\Psi_G^* \end{pmatrix} \tag{4.72}$$

On the other hand, if one subdivides matrices $L_{g,g'}$ and $M_{g,g'}$ as explained in Sections 3.2.3 and 3.4, one has to use other equations for calculating $\mathbf{k}\Psi_g^*$, from $g = u - 1$ to $g = 1$. However, one calculates $\mathbf{k}\Psi_g^*$ for $g \geq u$ without

the subdivision and with the same Equation 4.70. By contrast, for $g < u$, one subdivides $\mathbf{k}\Psi_g^*$ as shown in Equation 4.73. Then, one calculates sequentially $\mathbf{k}\Psi_g^{faces^*}$ and $\mathbf{k}\Psi_g^{cells^*}$, from $g = u-1$ to $g = 1$. For each energy group g, first, one calculates $\mathbf{k}\Psi_g^{faces^*}$ with Equation 4.74; second, one calculates $\mathbf{k}\Psi_g^{cells^*}$ with Equation 4.75, because it depends on $\mathbf{k}\Psi_g^{faces^*}$. Equations 4.74 and 4.75 depend on vectors $b_g^{faces,*}$ and $b_g^{cells,*}$. In addition, $b_g^{faces,*}$ also depends on $b_g^{cells,*}$. Thus, one has to first calculate $b_g^{cells,*}$ with Equation 4.76 for $g \geq 2$ and Equation 4.77 for $g = 1$. Then, one calculates $b_g^{faces,*}$ with Equation 4.78 for $g \geq 2$ and Equation 4.79 for $g = 1$. Finally, one calculates $\mathbf{k}\Phi^*$ by substituting $\mathbf{k}\Psi_g^*$, for $g \geq 1$, in Equation 4.72.

$$\mathbf{k}\Psi_g^* = \begin{pmatrix} \mathbf{k}\Psi_g^{cells^*} \\ \mathbf{k}\Psi_g^{faces^*} \end{pmatrix} \tag{4.73}$$

$$\overline{LD}_g^T \cdot \left(\mathbf{k}\Psi_g^{faces^*}\right) = b_g^{faces,*} \tag{4.74}$$

$$\mathbf{k}\Psi_g^{cells^*} = LA_g^{-1} \cdot \left(b_g^{cells,*} - LC_g^T \cdot \left(\mathbf{k}\Psi_g^{faces^*}\right)\right) \tag{4.75}$$

$$b_g^{cells,*} = M_{1,g}^{cells^T} \cdot \Psi_1^{cells^*} - \sum_{g'=g+1}^{G} L_{g',g}^{cells^T} \cdot \left(\mathbf{k}\Psi_{g'}^{cells^*}\right) \qquad , u < g < 1 \tag{4.76}$$

$$b_1^{cells,*} = M_{1,1}^{cells^T} \cdot \Psi_1^{cells^*} + LA_1 \cdot \sum_{g=2}^{G} C_g^{cells} \cdot \left(\mathbf{k}\Psi_g^{cells^*}\right) - \sum_{g=2}^{G} L_{g,1}^{cells^T} \cdot \left(\mathbf{k}\Psi_g^{cells^*}\right) \tag{4.77}$$

$$b_g^{faces,*} = -LB_g^T \cdot LA_g^{-1} \cdot b_g^{cells,*} \qquad , u < g < 1 \tag{4.78}$$

$$b_1^{faces,*} = LB_1^T \cdot \left(\sum_{g=2}^{G} C_g^{cells} \cdot \left(\mathbf{k}\Psi_g^{cells^*}\right) - LA_1^{-1} \cdot b_1^{cells,*}\right) \tag{4.79}$$

Since these calculated $\mathbf{k}\Phi^*$ are not the real adjoint eigenvectors, they might not accomplish the biorthogonal property of Equation 4.52. However, one can make sure that this property is satisfied for the M_d eigenvectors calculated. To

do so, one can orthogonalize the calculated $\mathbf{k}\Phi^*$ with respect to $L\Phi$. Another approach is to calculate the estimation Ψ_1^{m*}, for each mode m, with Equation 4.80. In this equation, one calculates Ψ_1^{m*} as a linear combination of Φ_1^l with coefficients $U_{l,m}$, which are calculated to accomplish the biorthogonal property.

$$\Psi_1^{m*} = \sum_{l=1}^{M_d} U_{l,m} \Phi_1^l \tag{4.80}$$

For the sake of simplicity, the author will calculate $U_{l,m}$ with $\left\langle \Psi^{m*}, \overline{L}\Phi^l \right\rangle$ instead of $\left\langle \Phi^{m*}, L\Phi^l \right\rangle$. Nevertheless, one can see in Equation 4.81 that both dot products are equivalent. Moreover, one can express the calculated $\mathbf{k}\Psi^*$ in matrix form as in Equation 4.82. In this equation, $\overline{L}^{T^{-1}} \cdot \overline{M}^T$ is the adjoint matrix and $\left[\Phi^1 \cdots \Phi^{M_d}\right] \cdot U$ is the estimation of $[\Psi_1^{1*} \cdots \Psi_1^{M_d*}]$. Then, one can substitute Equation 4.82 in the dot product of Equation 4.83. As this dot product should be the unit matrix to accomplish the biorthogonal property, one has to calculate U as in Equation 4.84. Although one has to calculate an inverse in Equation 4.84, this is straightforward because the size of the matrix is M_d. The value of matrix U for the row l and column m is the coefficient $U_{l,m}$.

$$\left\langle \Psi^{m*}, \overline{L}\Phi^l \right\rangle = \left\langle \Psi^{m*}, PL\Phi^l \right\rangle = \left\langle P^T\Psi^{m*}, L\Phi^l \right\rangle = \left\langle \Phi^{m*}, L\Phi^l \right\rangle \tag{4.81}$$

$$\left[\mathbf{k}_1\Psi^{1*} \cdots \mathbf{k}_{M_d}\Psi^{M_d*}\right] = \overline{L}^{T^{-1}} \cdot \overline{M}^T \cdot \left[\Phi^1 \cdots \Phi^{M_d}\right] \cdot U \tag{4.82}$$

$$\begin{aligned} &\left\langle \left[\mathbf{k}_1\Psi^{1*} \cdots \mathbf{k}_{M_d}\Psi^{M_d*}\right], \overline{L}\left[\Phi^1 \cdots \Phi^{M_d}\right] \right\rangle = \\ &= \left[\mathbf{k}_1\Psi^{1*} \cdots \mathbf{k}_{M_d}\Psi^{M_d*}\right]^T \cdot \overline{L} \cdot \left[\Phi^1 \cdots \Phi^{M_d}\right] \\ &= U^T \cdot \left[\Phi^1 \cdots \Phi^{M_d}\right]^T \cdot \overline{M} \cdot \left[\Phi^1 \cdots \Phi^{M_d}\right] \end{aligned} \tag{4.83}$$

$$
\begin{aligned}
U &= \left(\left[\Phi^1 \cdots \Phi^{M_d}\right]^T \cdot \overline{M}^T \cdot \left[\Phi^1 \cdots \Phi^{M_d}\right]\right)^{-1} \\
&= \begin{pmatrix} \left\langle \sum_{g=1}^{G} M_{1,g} \cdot \Phi_g^1, \Phi_1^1 \right\rangle & \cdots & \left\langle \sum_{g=1}^{G} M_{1,g} \cdot \Phi_g^1, \Phi_1^{M_d} \right\rangle \\ \vdots & & \vdots \\ \left\langle \sum_{g=1}^{G} M_{1,g} \cdot \Phi_g^{M_d}, \Phi_1^1 \right\rangle & \cdots & \left\langle \sum_{g=1}^{G} M_{1,g} \cdot \Phi_g^{M_d}, \Phi_1^{M_d} \right\rangle \end{pmatrix}^{-1}_{M_d \times M_d}
\end{aligned}
\tag{4.84}
$$

4.3 Updating modes

In real transients, one should take into account that the cross sections are dependent not only on time, but also on the space variables. Consequently, the spatial distribution of the modes might also change over time, and therefore the values of $\Phi(\vec{r}, t)$ might not be accurate. However, one can use two different approaches to obtain accurate solutions. One the one hand, one can use a large number of modes (M_d) in the expansion of Equation 4.24, but this is not efficient from a computational point of view. On the other hand, one can use only a small number of M_d, but updating the spatial distribution of the modes at certain time steps. The code developed in this thesis can use both approaches, but the author strongly recommends using the second one.

The update of modes is the calculation of the eigenvalue problem for the conditions at certain time t_i. Equation 4.85 shows this eigenvalue problem. In this equation, L^i and M^i are matrices L and M for the time t_i, whereas $\mathbf{k}_l^i$ and $\Phi^{l,i}$ are the eigenvalue and eigenvector of the mode l for the eigenvalue problem at time t_i. In addition, one should also calculate the adjoint eigenvectors $\Phi^{l,i*}$, as explained in Section 4.2.

$$
L^i \Phi^{l,i} = \frac{1}{\mathbf{k}_l^i} M^i \Phi^{l,i} \tag{4.85}
$$

Furthermore, one can also use the same equations of Section 4.1, but with the updated terms at time t_i, to continue calculating the time amplitudes in other time intervals. Nonetheless, one has to recalculate the initial conditions ($\mathcal{N}_i$) for the new conditions at time t_i. These new initial conditions will be used for the next time interval $[t_i, t_{i+1}]$. The new initial conditions are $n_l^i(t_i)$ and $C_{m,k}^i(t_i)$, which are the values of $n_l(ti)$ and $C_{m,k}(t_i)$ for the conditions of time t_i. Therefore, they might differ from the previous time t_{i-1}, because the

eigenvalues, eigenvectors and matrices might be different. Since the solution of $\Phi(t)$ has to be continuous, one should consider the value of $\Phi(t_i)$ for both conditions of the times t_{i-1} and t_i, as shown in Equations 4.86 and 4.87 respectively. If one combines these two equations, one obtains the new initial conditions $n_m^i(t_i)$ as shown in 4.88.

$$\Phi(t_i) = \sum_{l=1}^{M_d} n_l^{i-1}(t_i)\Phi^{l,i-1} \tag{4.86}$$

$$\Phi(t_i) = \sum_{l=1}^{M_d} n_l^{i}(t_i)\Phi^{l,i} \tag{4.87}$$

$$n_m^i(t_i) = \frac{\left\langle \Phi^{m,i^*}, M^i\Phi(t_i)\right\rangle}{\left\langle \Phi^{m,i^*}, M^i\Phi^{m,i}\right\rangle} = \frac{\left\langle \Phi^{m,i^*}, M^i \sum_{l=1}^{M_d} n_l^{i-1}(t_i)\Phi^{l,i-1}\right\rangle}{\left\langle \Phi^{m,i^*}, M^i\Phi^{m,i}\right\rangle} \tag{4.88}$$

Unfortunately, the calculation of the initial conditions $C_{m,k}^i(t_i)$ is not as straightforward as the previous one. Thus, one could calculate them by using some approach, as that of Equation 4.89. In this equation, one has to calculate the coefficients $a_{m,l}$ to accomplish the equation. Actually, one can calculate $a_{m,l}$ as in Equation 4.90.

$$\Phi^{m,i^*} \approx \sum_{l=1}^{M_d} a_{m,l}\Phi^{l,i-1^*} \tag{4.89}$$

$$a_{m,l} = \frac{\left\langle \Phi^{m,i^*}, M^{i-1}\Phi^{l,i-1}\right\rangle}{\left\langle \Phi^{m,i-1^*}, M^{i-1}\Phi^{l,i-1}\right\rangle} \tag{4.90}$$

Finally, if one uses these two equations, one can obtain $C_{m,k}^i(t_i)$, as explained in Equation 4.91.

$$\begin{aligned} C_{m,k}^i(t_i) = \left\langle {X_k^{del}}^T\Phi^{m,i^*}, C_k(t_i)\right\rangle &= \sum_{l=1}^{M_d} a_{m,l}\left\langle {X_k^{del}}^T\Phi^{l,i-1^*}, C_k(t_i)\right\rangle = \\ &= \sum_{l=1}^{M_d} a_{m,l}C_{l,k}^{i-1}(t_i) \end{aligned} \tag{4.91}$$

Chapter 5

Steady State of the Neutron Transport Equation with the Discrete Ordinates formulation and the Finite Volume Method

5.1 Discrete Ordinates formulation

In this section, the author applies the FVM to the Steady State of the Discrete Ordinates Neutron Transport Equation. Section 2.2 explains how to obtain this equation (Equation 2.55). One can rewrite this equation as in Equation 5.1, where μ_m and φ_m express the direction dependence.

$$\nabla\vec{\Omega}_m\psi_g(\mu_m,\varphi_m,\vec{r})+\Sigma_{t,g}(\vec{r})\psi_g(\mu_m,\varphi_m,\vec{r})-$$
$$-\sum_{g'=1}^{G}\sum_{l=0}^{L}(2l+1)\Sigma_{s,g'\to g,l}(\vec{r})\Bigg\{P_l(\mu_m)\phi_{g',l}(\vec{r})+$$
$$+2\sum_{k=1}^{l}\frac{(l-k)!}{(l+k)!}P_l^k(\mu_m)\cdot\left[\phi^k_{C,g',l}(\vec{r})\cos(k\varphi_m)+\phi^k_{S,g',l}(\vec{r})\sin(k\varphi_m)\right]\Bigg\}=$$
$$=\frac{\chi_g(\vec{r})}{\mathbf{k}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r})\phi_{g',0}(\vec{r}) \tag{5.1}$$

In the previous equation, the moments $\phi_{g',l}$, $\phi^k_{C,g',l}$ and $\phi^k_{S,g',l}$ were already defined in Section 2.2 (Equations 2.56-2.58). However, in this chapter, the author will use the moments of Equations 5.2-5.4, because the author will normalize the weights w_n to 8: $\sum_{n=1}^{N_d} w_n = 8$. This is a typical normalization, as there are eight octants.

$$\phi_{g',l}(\vec{r})=\frac{1}{8}\sum_{n=1}^{N_d}w_nP_l(\mu_n)\psi_{g'}(\vec{r},\mu_n,\varphi_n) \tag{5.2}$$

$$\phi^k_{C,g',l}(\vec{r})=\frac{1}{8}\sum_{n=1}^{N_d}w_nP_l^k(\mu_n)\cos(k\varphi_n)\psi_{g'}(\vec{r},\mu_n,\varphi_n) \tag{5.3}$$

$$\phi^k_{S,g',l}(\vec{r})=\frac{1}{8}\sum_{n=1}^{N_d}w_nP_l^k(\mu_n)\sin(k\varphi_n)\psi_{g'}(\vec{r},\mu_n,\varphi_n) \tag{5.4}$$

Before applying the FVM to Equation 5.1, the author defines the cell averaged values of the angular flux and moments, as shown in Equations 5.5-5.8.

$$\psi_{g,n,i}=\frac{1}{V_i}\int_{V_i}\psi_g(\vec{r},\mu_n,\varphi_n)dV \tag{5.5}$$

$$\begin{aligned}\phi_{g',l,i} &= \frac{1}{V_i}\int_{V_i}\phi_{g',l}(\vec{r})dV = \\ &= \frac{1}{8}\sum_{n=1}^{N_d} w_n P_l(\mu_n)\frac{1}{V_i}\int_{V_i}\psi_{g'}(\vec{r},\mu_n,\varphi_n)dV = \\ &= \frac{1}{8}\sum_{n=1}^{N_d} w_n P_l(\mu_n)\psi_{g',n,i} \end{aligned} \tag{5.6}$$

$$\begin{aligned}\phi^k_{C,g',l,i} &= \frac{1}{V_i}\int_{V_i}\phi^k_{C,g',l}(\vec{r})dV = \\ &= \frac{1}{8}\sum_{n=1}^{N_d} w_n P^k_l(\mu_n)\cos(k\varphi_n)\frac{1}{V_i}\int_{V_i}\psi_{g'}(\vec{r},\mu_n,\varphi_n)dV = \\ &= \frac{1}{8}\sum_{n=1}^{N_d} w_n P^k_l(\mu_n)\cos(k\varphi_n)\psi_{g',n,i} \end{aligned} \tag{5.7}$$

$$\begin{aligned}\phi^k_{S,g',l,i} &= \frac{1}{V_i}\int_{V_i}\phi^k_{S,g',l}(\vec{r})dV = \\ &= \frac{1}{8}\sum_{n=1}^{N_d} w_n P^k_l(\mu_n)\sin(k\varphi_n)\frac{1}{V_i}\int_{V_i}\psi_{g'}(\vec{r},\mu_n,\varphi_n)dV = \\ &= \frac{1}{8}\sum_{n=1}^{N_d} w_n P^k_l(\mu_n)\sin(k\varphi_n)\psi_{g',n,i} \end{aligned} \tag{5.8}$$

Since the FVM transforms the cell averaged value of the divergence term into face averaged values, Equation 5.9 shows the face averaged value of the angular flux $\psi_{g,n}$.

$$\psi_{g,n,j} = \frac{1}{S_j}\int_{S_j}\psi_g(\vec{r},\mu_n,\varphi_n)dS \tag{5.9}$$

Now, one applies the FVM to Equation 5.1. First, one obtains cell averaged values of the mentioned equation in each cell i of the discretized geometry, as shown in Equation 5.10.

$$\oint \vec{\Omega}_m \psi_g(\mu_m, \varphi_m, \vec{r}) d\vec{S} + \int_{V_i} \Sigma_{t,g}(\vec{r}) \psi_g(\mu_m, \varphi_m, \vec{r}) dV -$$
$$- \int_{V_i} dV \sum_{g'=1}^{G} \sum_{l=0}^{L} (2l+1) \Sigma_{s,g' \to g,l}(\vec{r}) \Bigg\{ P_l(\mu_m) \phi_{g',l}(\vec{r}) +$$
$$+ 2 \sum_{k=1}^{l} \frac{(l-k)!}{(l+k)!} P_l^k(\mu_m) \cdot \left[\phi_{C,g',l}^k(\vec{r}) \cos(k\varphi_m) + \phi_{S,g',l}^k(\vec{r}) \sin(k\varphi_m) \right] \Bigg\} =$$
$$= \frac{1}{\mathbf{k}} \int_{V_i} dV \chi_g(\vec{r}) \sum_{g'=1}^{G} \nu \Sigma_{f,g'}(\vec{r}) \phi_{g',0}(\vec{r}) \tag{5.10}$$

Second, one sets the cross sections and fission spectrum corresponding to the material of the cell i, as shown in Equation 5.11.

$$\sum_{j=1}^{n_f} \int_{S_j} \psi_g(\mu_m, \varphi_m, \vec{r}) \vec{\Omega}_m \cdot d\vec{S} + \Sigma_{t,g}^i \int_{V_i} \psi_g(\mu_m, \varphi_m, \vec{r}) dV -$$
$$- \sum_{g'=1}^{G} \sum_{l=0}^{L} (2l+1) \Sigma_{s,g' \to g,l}^i \Bigg\{ P_l(\mu_m) \int_{V_i} \phi_{g',l}(\vec{r}) dV +$$
$$+ 2 \sum_{k=1}^{l} \frac{(l-k)!}{(l+k)!} P_l^k(\mu_m) \cdot \Bigg[\cos(k\varphi_m) \int_{V_i} \phi_{C,g',l}^k(\vec{r}) dV +$$
$$+ \sin(k\varphi_m) \int_{V_i} \phi_{S,g',l}^k(\vec{r}) dV \Bigg] \Bigg\} = \frac{1}{\mathbf{k}} \chi_g^i \sum_{g'=1}^{G} \nu \Sigma_{f,g'}^i \int_{V_i} \phi_{g',0}(\vec{r}) dV \tag{5.11}$$

Third, one obtains the cell averaged values of the angular fluxes and the moments. In addition, one calculates the face averaged values on each face of the cell. This is shown in Equation 5.12, where $u_{i,j,x}$, $u_{i,j,y}$ and $u_{i,j,z}$ are the direction cosines of the normal of face j.

$$\sum_{j=1}^{n_f}\left(\mu_m u_{i,j,x}+\eta_m u_{i,j,y}+\xi_m u_{i,j,z}\right)S_j\psi_{g,m,j}+\Sigma_{t,g}^{i}V_i\psi_{g,m,i}-$$
$$-\sum_{g'=1}^{G}\sum_{l=0}^{L}(2l+1)\Sigma_{s,g'\rightarrow g,l}^{i}\Bigg\{P_l(\mu_m)V_i\phi_{g',l,i}+$$
$$+2\sum_{k=1}^{l}\frac{(l-k)!}{(l+k)!}P_l^k(\mu_m)\cdot\left[\cos(k\varphi_m)V_i\phi_{C,g',l,i}^{k}+\sin(k\varphi_m)V_i\phi_{S,g',l,i}^{k}\right]\Bigg\}=$$
$$=\frac{1}{\mathbf{k}}\chi_g^i\sum_{g'=1}^{G}\nu\Sigma_{f,g'}^{i}V_i\phi_{g',0,i} \tag{5.12}$$

Fourth, one substitutes the moments of Equations 5.6-5.8 in the previous equation, so one obtains Equation 5.13.

$$\sum_{j=1}^{n_f}\left(\mu_m u_{i,j,x}+\eta_m u_{i,j,y}+\xi_m u_{i,j,z}\right)S_j\psi_{g,m,j}+\Sigma_{t,g}^{i}V_i\psi_{g,m,i}-$$
$$-V_i\sum_{g'=1}^{G}\sum_{l=0}^{L}(2l+1)\Sigma_{s,g'\rightarrow g,l}^{i}\Bigg\{P_l(\mu_m)\frac{1}{8}\sum_{n=1}^{N_d}w_nP_l(\mu_n)\psi_{g',n,i}+$$
$$+2\sum_{k=1}^{l}\frac{(l-k)!}{(l+k)!}P_l^k(\mu_m)\cdot\Bigg[\cos(k\varphi_m)\frac{1}{8}\sum_{n=1}^{N_d}w_nP_l^k(\mu_n)\cos(k\varphi_n)\psi_{g',n,i}+$$
$$+\sin(k\varphi_m)\frac{1}{8}\sum_{n=1}^{N_d}w_nP_l^k(\mu_n)\sin(k\varphi_n)\psi_{g',n,i}\Bigg]\Bigg\}=$$
$$=\frac{1}{\mathbf{k}}\chi_g^i\sum_{g'=1}^{G}\nu\Sigma_{f,g'}^{i}V_i\frac{1}{8}\sum_{n=1}^{N_d}w_nP_0(\mu_n)\psi_{g',n,i} \tag{5.13}$$

Then, one can reorder the previous equation to obtain Equation 5.14, where one has used $P_0(\mu_n)=1$.

$$\sum_{j=1}^{n_f}\left(\mu_m u_{i,j,x}+\eta_m u_{i,j,y}+\xi_m u_{i,j,z}\right)S_j\psi_{g,m,j}+\Sigma^i_{t,g}V_i\psi_{g,m,i}-$$
$$-\frac{V_i}{8}\sum_{g'=1}^{G}\sum_{n=1}^{N_d}w_n\psi_{g',n,i}\sum_{l=0}^{L}(2l+1)\Sigma^i_{s,g'\to g,l}\Bigg\{P_l(\mu_m)P_l(\mu_n)+$$
$$+2\sum_{k=1}^{l}\frac{(l-k)!}{(l+k)!}P_l^k(\mu_m)P_l^k(\mu_n)\left[\cos(k\varphi_m)\cos(k\varphi_n)+\sin(k\varphi_m)\sin(k\varphi_n)\right]\Bigg\}=$$
$$=\frac{1}{\mathbf{k}}\frac{V_i}{8}\chi^i_g\sum_{g'=1}^{G}\nu\Sigma^i_{f,g'}\sum_{n=1}^{N_d}w_n\psi_{g',n,i} \tag{5.14}$$

Finally, one can define $\Sigma^i_{s,g'\to g,m,n}$ as in Equation 5.15. If one substitutes this value in Equation 5.14, one obtains Equation 5.16. Therefore, $\Sigma^i_{s,g'\to g,m,n}$ represents the scattering macroscopic cross section, from the energy group g' to g and from the direction $\vec{\Omega}_m$ to $\vec{\Omega}_n$, in Equation 5.16.

$$\begin{aligned}\Sigma^i_{s,g'\to g,m,n} &= \sum_{l=0}^{L}(2l+1)\Sigma^i_{s,g'\to g,l}\Bigg\{P_l(\mu_m)P_l(\mu_n)+\\ &+ 2\sum_{k=1}^{l}\frac{(l-k)!}{(l+k)!}P_l^k(\mu_m)P_l^k(\mu_n)\cos\left[k(\varphi_m-\varphi_n)\right]\Bigg\}\end{aligned} \tag{5.15}$$

$$\sum_{j=1}^{n_f}S_j\left(\mu_m u_{i,j,x}+\eta_m u_{i,j,y}+\xi_m u_{i,j,z}\right)\psi_{g,m,j}+V_i\Sigma^i_{t,g}\psi_{g,m,i}-$$
$$-\frac{V_i}{8}\sum_{g'=1}^{G}\sum_{n=1}^{N_d}w_n\Sigma^i_{s,g'\to g,m,n}\psi_{g',n,i}=\frac{1}{\mathbf{k}}\frac{V_i}{8}\chi^i_g\sum_{g'=1}^{G}\nu\Sigma^i_{f,g'}\sum_{n=1}^{N_d}w_n\psi_{g',n,i} \tag{5.16}$$

Equation 5.16 is the Discrete Ordinates formulation of the Steady State of the Neutron Transport Equation with the Finite Volume Method. The main advantage of this equation is its simplicity, since all the direction variables are included in $\Sigma^i_{s,g'\to g,m,n}$. Furthermore, it should be highlighted that $\Sigma^i_{s,g'\to g,m,n}$ depends only on the directions $\vec{\Omega}_m$ and $\vec{\Omega}_n$ and on $\Sigma^i_{s,g'\to g,l}$. Thus, one can calculate it a priori.

On the other hand, one can obtain another simplification of $\Sigma^i_{s,g'\to g,m,n}$ for 2D cases. In these cases, half of the directions are symmetric to the other half, so one can calculate only the first set. To define the symmetry of the directions, one should consider the 2D domain and the direction cosines. Let us consider that the domain is defined in the XY plane and the direction cosines are those of Figure 2.1 and Equation 2.25. For these conditions, the symmetry of the direction is in ξ. Thus, if direction n_2 is the symmetric direction of n_1, then $\mu_{n_2} = \mu_{n_1}$, $\eta_{n_2} = \eta_{n_1}$ and $\xi_{n_2} = -\xi_{n_1}$. Moreover, one can also define this symmetry in terms of φ: $\cos(\varphi_{n_1}) = \cos(\varphi_{n_2})$ and $\sin(\varphi_{n_1}) = -\sin(\varphi_{n_2})$. From these, one can easily prove two statements: first, if $\cos(\varphi_{n_1}) = \cos(\varphi_{n_2})$, then $\cos(k\varphi_{n_1}) = \cos(k\varphi_{n_2})$; second, if $\sin(\varphi_{n_1}) = -\sin(\varphi_{n_2})$, then $\sin(k\varphi_{n_1}) = -\sin(k\varphi_{n_2})$. Taking into account these statements, one can sum $\Sigma^i_{s,g'\to g,m,n_1}$ and $\Sigma^i_{s,g'\to g,m,n_2}$, for these directions n_1 and n_2, as shown in Equation 5.17.

$$
\begin{aligned}
&\Sigma^i_{s,g'\to g,m,n_1} + \Sigma^i_{s,g'\to g,m,n_2} = \sum_{l=0}^{L}(2l+1)\Sigma^i_{s,g'\to g,l}\Bigg\{ P_l(\mu_m)\left[P_l(\mu_{n_1}) + P_l(\mu_{n_2})\right] + \\
&+ 2\sum_{k=1}^{l}\frac{(l-k)!}{(l+k)!}P_l^k(\mu_m)\left(P_l^k(\mu_{n_1})\cos\left[k(\varphi_m-\varphi_{n_1})\right] + \right. \\
&\left. + P_l^k(\mu_{n_2})\cos\left[k(\varphi_m-\varphi_{n_2})\right]\right)\Bigg\} = \sum_{l=0}^{L}(2l+1)\Sigma^i_{s,g'\to g,l}\Bigg\{ 2P_l(\mu_m)P_l(\mu_{n_1}) + \\
&+ 2\sum_{k=1}^{l}\frac{(l-k)!}{(l+k)!}P_l^k(\mu_m)P_l^k(\mu_{n_1})\left(\cos\left[k(\varphi_m-\varphi_{n_1})\right] + \cos\left[k(\varphi_m-\varphi_{n_2})\right]\right)\Bigg\} = \\
&= \sum_{l=0}^{L}(2l+1)\Sigma^i_{s,g'\to g,l}\Bigg\{ 2P_l(\mu_m)P_l(\mu_{n_1}) + \\
&+ 2\sum_{k=1}^{l}\frac{(l-k)!}{(l+k)!}P_l^k(\mu_m)P_l^k(\mu_{n_1})2\cos(k\varphi_m)\cos(k\varphi_{n_1})\Bigg\} = \\
&= 2\sum_{l=0}^{L}(2l+1)\Sigma^i_{s,g'\to g,l}\Bigg\{ P_l(\mu_m)P_l(\mu_{n_1}) + \\
&+ 2\sum_{k=1}^{l}\frac{(l-k)!}{(l+k)!}P_l^k(\mu_m)P_l^k(\mu_{n_1})\cos(k\varphi_m)\cos(k\varphi_{n_1})\Bigg\}
\end{aligned}
\tag{5.17}
$$

Equation 5.17 is similar to Equation 5.15, so one can define a similar $\Sigma^i_{s,g'\to g,m,n}$ for 2D cases as shown in Equation 5.18.

$$\begin{aligned}\Sigma^{2D,i}_{s,g'\to g,m,n} &= \sum_{l=0}^{L}(2l+1)\Sigma^{i}_{s,g'\to g,l}\Bigg\{P_l(\mu_m)P_l(\mu_n)+\\ &+ 2\sum_{k=1}^{l}\frac{(l-k)!}{(l+k)!}P_l^k(\mu_m)P_l^k(\mu_n)\cos(k\varphi_m)\cos(k\varphi_n)\Bigg\} \quad (5.18)\end{aligned}$$

To obtain the simplification for 2D cases, one can simplify the scattering terms as shown in Equation 5.19. In this equation, one takes into account that for a direction n_2 which is the symmetric one of n_1: $w_{n_1} = w_{n_2}$ and $\psi_{g',n_1,i} = \psi_{g',n_2,i}$. Finally, one obtains a similar scattering term to that of Equation 5.16. Nevertheless, in Equation 5.19, one uses $\Sigma^{2D,i}_{s,g'\to g,m,n_1}$ and the following weights: $w^{2D}_{n_1} = 2w_{n_1}$.

$$\begin{aligned}&\sum_{n=1}^{N_d} w_n\Sigma^{i}_{s,g'\to g,m,n}\psi_{g',n,i} = \sum_{n_1=1}^{\frac{N_d}{2}} w_{n_1}\Sigma^{i}_{s,g'\to g,m,n_1}\psi_{g',n_1,i}+\\ &+\sum_{n_2=1}^{\frac{N_d}{2}} w_{n_2}\Sigma^{i}_{s,g'\to g,m,n_2}\psi_{g',n_2,i} = \sum_{n_1=1}^{\frac{N_d}{2}} w_{n_1}\psi_{g',n_1,i}\left(\Sigma^{i}_{s,g'\to g,m,n_1}+\Sigma^{i}_{s,g'\to g,m,n_2}\right) =\\ &= \sum_{n_1=1}^{\frac{N_d}{2}} w_{n_1}\psi_{g',n_1,i}2\Sigma^{2D,i}_{s,g'\to g,m,n_1} = \sum_{n_1=1}^{\frac{N_d}{2}} w^{2D}_{n_1}\Sigma^{2D,i}_{s,g'\to g,m,n_1}\psi_{g',n_1,i} \quad (5.19)\end{aligned}$$

On the other hand, the continuity condition of the angular flux is accomplished implicitly, because one defines only one angular flux for each inner face j: $\psi_{g,m,j}$. Although this value is not known, the author describes in Section 5.3 two methods for calculating the face averaged values from the cell averaged values of the angular flux.

As regards the boundary conditions, the author only used two type of boundary conditions for the method developed in this thesis: vacuum and reflective. As discussed in Section 2.2, the boundary conditions are applied to the incoming angular fluxes. Mathematically, one can determine if an angular flux $\psi_{g,m,j}$, whose direction is defined by the unit vector of Equation 5.20, will be incoming to a boundary face, whose normal vector is defined by Equation 5.21. If the dot product of both vectors, which is shown in Equation 5.22, is negative, the angular flux is incoming. Once one knows the incoming angular fluxes ($\psi_{g,m_{in},j}$), one can easily set the vacuum boundary conditions: $\psi_{g,m_{in},j} = 0$.

$$\vec{u}_m = \mu_m \vec{i} + \eta_m \vec{j} + \xi_m \vec{k} \tag{5.20}$$

$$\vec{u}_{i,j} = u_{i,j,x} \vec{i} + u_{i,j,y} \vec{j} + u_{i,j,z} \vec{k} \tag{5.21}$$

$$\langle \vec{u}_m, \vec{u}_{i,j} \rangle = \mu_m u_{i,j,x} + \eta_m u_{i,j,y} + \xi_m u_{i,j,z} \tag{5.22}$$

Regarding reflective boundary conditions, for an incoming angular flux with direction $\vec{\Omega}_m$, whose reflective angular flux is defined by direction $\vec{\Omega}_{r_m}$, the boundary condition is: $\psi_{g,m,j} = \psi_{g,r_m,j}$. However, one should determine the reflective direction for each of the incoming angular fluxes. To do so, one can consider Figure 5.1, where $\vec{u}_m$ is the unit vector of an incoming angular flux, $\vec{u}_{i,j}$ is the unit vector of the normal of the boundary face, and $\vec{u}_{r_m}$ is the unit vector of the reflective direction. From this figure, one realizes that $\vec{u}_{r_m}$ can be calculated with Equation 5.23, for certain coefficient a, which is not known. Since the norm of a unit vector must be one, one can calculate this coefficient a with Equation 5.24. Thus, one obtains $a = 2 \langle \vec{u}_m, \vec{u}_{i,j} \rangle$, and consequently one can calculate the reflective direction with Equation 5.25.

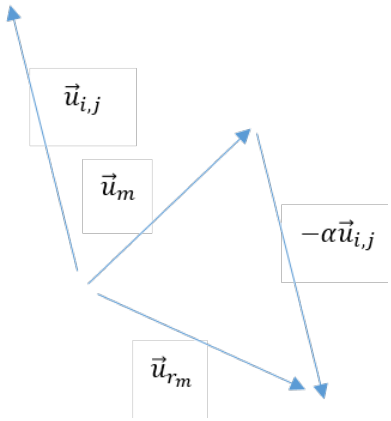


Figure 5.1: Reflective direction

$$\vec{u}_{r_m} = \vec{u}_m - a\vec{u}_{i,j} = (\mu_m - a u_{i,j,x})\vec{i} + (\eta_m - a u_{i,j,y})\vec{j} + (\xi_m - a u_{i,j,z})\vec{k} \tag{5.23}$$

$$1 = |\vec{u}_{r_m}| = (\mu_m - a u_{i,j,x})^2 + (\eta_m - a u_{i,j,y})^2 + (\xi_m - a u_{i,j,z})^2 \tag{5.24}$$

$$\vec{u}_{r_m} = (\mu_m - 2u_{i,j,x} \langle \vec{u}_m, \vec{u}_{i,j} \rangle) \vec{i} + (\eta_m - 2u_{i,j,y} \langle \vec{u}_m, \vec{u}_{i,j} \rangle) \vec{j} + (\xi_m - 2u_{i,j,z} \langle \vec{u}_m, \vec{u}_{i,j} \rangle) \vec{k} \tag{5.25}$$

Once one knows the unit vector of the reflective direction $\vec{u}_{r_m}$, one has to identify which outgoing direction is. If the directions of the quadrature are not symmetric, there will not exist an outgoing direction with unit vector $\vec{u}_{r_m}$. Nevertheless, one can choose the direction with the closest unit vector to $\vec{u}_{r_m}$. For this purpose, one should calculate the dot product of $\vec{u}_{r_m}$ and all the directions of the quadrature set, as shown in Equation 5.26. Then, one calculates R_n with Equation 5.27 for all the directions. Finally, the direction with the minimum value of R_n is the closest one to the reflective direction.

$$\langle \vec{u}_{r_m}, \vec{u}_n \rangle = \langle \vec{u}_m - 2 \langle \vec{u}_m, \vec{u}_{i,j} \rangle \vec{u}_{i,j}, \vec{u}_n \rangle = \langle \vec{u}_m, \vec{u}_n \rangle - 2 \langle \vec{u}_m, \vec{u}_{i,j} \rangle \langle \vec{u}_{i,j}, \vec{u}_n \rangle \tag{5.26}$$

$$R_n = |\langle \vec{u}_{r_m}, \vec{u}_n \rangle - 1| \tag{5.27}$$

If one applies Equation 5.16 to the N_d directions and N_c cells of the discretized geometry, one obtains the same eigenvalue problem as that of the Neutron Diffusion Equation (Equation 3.87), but with different vectors Ψ_g and matrices $L_{g,g'}$ and $M_{g,g'}$. Equation 5.28 shows Ψ_g and Equation 5.29 shows matrices $M_{g,g'}$. These matrices are composed of matrices $M_{g,g',i}$ for each cell i, which are defined in Equation 5.30. As regards matrices $L_{g,g'}$, Equation 5.31 shows matrices $L_{g,g'}$ for $g \neq g'$, which are composed of matrices $L_{g,g',i}$ for each cell i. One can see these matrices $L_{g,g',i}$ in Equation 5.32. Unfortunately, it is difficult to give the composition of matrices $L_{g,g}$, because of the face averaged values of the angular flux. Thus, the author will only specify the terms of Equation 5.16 that one has to use to build these matrices: $S_j (\mu_m u_{i,j,x} + \eta_m u_{i,j,y} + \xi_m u_{i,j,z})$, $V_i \Sigma^i_{t,g}$ and $-\frac{V_i}{8} w_n \Sigma^i_{s,g \to g,m,n}$.

$$\Psi_g = \begin{pmatrix} \psi_{g,1,1} \\ \vdots \\ \psi_{g,N_d,1} \\ \psi_{g,1,2} \\ \vdots \\ \psi_{g,N_d,2} \\ \vdots \\ \psi_{g,1,N_c} \\ \vdots \\ \psi_{g,N_d,N_c} \end{pmatrix}_{(N_d \cdot N_c)\times 1} \tag{5.28}$$

$$M_{g,g'} = \begin{pmatrix} M_{g,g',1} & 0_{N_d\times N_d} & \cdots & 0_{N_d\times N_d} \\ 0_{N_d\times N_d} & M_{g,g',2} & \ddots & 0_{N_d\times N_d} \\ \vdots & \ddots & \ddots & \ddots \\ 0_{N_d\times N_d} & \ddots & 0_{N_d\times N_d} & M_{g,g',N_c} \end{pmatrix}_{(N_d \cdot N_c)\times(N_d \cdot N_c)} \tag{5.29}$$

$$M_{g,g',i} = \frac{V_i}{8}\chi_g^i \nu \Sigma_{f,g'}^i \cdot \begin{pmatrix} w_1 & \cdots & w_{N_d} \\ \vdots & \vdots & \vdots \\ w_1 & \cdots & w_{N_d} \end{pmatrix}_{N_d\times N_d} \tag{5.30}$$

$$L_{g,g'} = \begin{pmatrix} L_{g,g',1} & 0_{N_d\times N_d} & \cdots & 0_{N_d\times N_d} \\ 0_{N_d\times N_d} & L_{g,g',2} & \ddots & 0_{N_d\times N_d} \\ \vdots & \ddots & \ddots & \ddots \\ 0_{N_d\times N_d} & \ddots & 0_{N_d\times N_d} & L_{g,g',N_c} \end{pmatrix}_{(N_d \cdot N_c)\times(N_d \cdot N_c)} \quad , g \neq g' \tag{5.31}$$

$$L_{g,g',i} = -\frac{V_i}{8} \cdot \begin{pmatrix} w_1\Sigma_{s,g'\to g,1,1}^i & \cdots & w_{N_d}\Sigma_{s,g'\to g,1,N_d}^i \\ \vdots & \vdots & \vdots \\ w_1\Sigma_{s,g'\to g,N_d,1}^i & \cdots & w_{N_d}\Sigma_{s,g'\to g,N_d,N_d}^i \end{pmatrix}_{N_d\times N_d} \quad , g \neq g' \tag{5.32}$$

5.2 Gauss-Legendre Product Quadrature

Section 2.2 states that there is a number of quadrature sets for the Discrete Ordinates, such as: level-symmetric, Legendre-Chebyshev and product quadrature. The most known and used is level-symmetric, because it conserves the symmetry in each octant and it conserves the moments of the direction cosines. However, level-symmetric is not the best quadrature for integrating any function with respect to the direction variables. Actually, the goal of the Discrete Ordinates is to use few directions to perform the integrals of the functions depending on the direction variables. As a result, one applies the product quadrature to obtain more accuracy in the integration. As explained in Section 2.2, particularly in Equation 2.47, the product quadrature uses a set of collocation points and weights for each of the variables $\cos(\theta)$ and φ. The typical product quadrature uses the Gauss-Legendre quadrature to integrate with respect to $\cos(\theta)$. Nonetheless, this quadrature typically uses a different approach for φ. In fact, it divides the whole domain of φ, which is 2π, in n number of collocation points and set the same value for all the weights.

In this section, the author develops another product quadrature using the Gauss-Legendre quadrature for $\cos(\theta)$ and φ. It should be highlighted that this product quadrature is a new development. First, the method considers N_p collocation points for $\cos(\theta)$. This value N_p is an even number, with the aim of conserving the symmetry with respect to θ. For each one of the N_p collocation points, the method uses the weights w_i^x and collocation points x_i of the Gauss-Legendre quadrature. Then, one can integrate any function f by means of this quadrature as shown in Equation 5.33.

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{N_p} w_i^x f(x_i) \tag{5.33}$$

The Gauss-Legendre quadrature is suitable for $\cos(\theta)$, because the domain of the integration ranges from -1 to 1, which is the same as the Gauss-Legendre quadrature. By contrast, the domain of integration of φ ranges from 0 to 2π. For this reason, if one wants to use the Gauss-Legendre quadrature, one has to perform the change of variable of Equation 5.34. In this equation, w_j^y and y_j are the weights and collocation points for N_a points of the Gauss-Legendre quadrature.

$$\int_a^b f(y)dy = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}y + \frac{a+b}{2}\right)dy \approx$$
$$\approx \frac{b-a}{2}\sum_{j=1}^{N_a} w_j^y f\left(\frac{b-a}{2}y_j + \frac{a+b}{2}\right) \tag{5.34}$$

The author proposed to perform four integrals with respect to φ, one in each quadrant of the domain of φ. Therefore, in this method one chooses N_a collocation points, so one integrates any function with respect to φ in each quadrant q, as explained in Equation 5.35.

$$\int_{\frac{(q-1)\pi}{2}}^{\frac{q\pi}{2}} f(y)dy \approx \frac{\pi}{4}\sum_{j=1}^{N_a} w_j^y f\left(\frac{\pi}{4}y_j + \frac{(2q-1)\pi}{4}\right) \qquad ,1 \leq q \leq 4 \tag{5.35}$$

To sum up, the collocation points and weights are shown in Equations 5.36-5.38. Since the collocation points of the gaussian quadrature, x_i and y_j, are symmetric in $[-1, 1]$, the collocation points $\cos(\theta_i)$ and $\varphi_{g,j}$ will be symmetric too. Regarding the weights, one multiplies the weights by *constant*, as one can see in Equation 5.38. This value is a normalization coefficient to normalize weights to 8; thus, one calculates *constant* as in Equation 5.39.

$$\cos(\theta_i) = x_i \qquad ,1 \leq i \leq N_p \tag{5.36}$$

$$\varphi_{q,j} = \frac{\pi}{4}y_j + \frac{(2q-1)\pi}{4} \qquad ,1 \leq j \leq N_a \qquad ,1 \leq q \leq 4 \tag{5.37}$$

$$w_{i,q,j} = constant \cdot w_i^x \cdot w_j^y \qquad ,1 \leq j \leq N_a \qquad ,1 \leq q \leq 4 \qquad ,1 \leq i \leq N_p \tag{5.38}$$

$$constant = \frac{8}{\sum_{i=1}^{N_p}\sum_{q=1}^{4}\sum_{j=1}^{N_a} w_i^x \cdot w_j^y} \tag{5.39}$$

Finally, one calculates the direction cosines with Equation 5.40.

$$\begin{aligned} \mu_{i,q,j} &= \cos(\theta_i) \\ \xi_{i,q,j} &= \sqrt{1-\cos^2(\theta_i)} \cdot \sin(\varphi_{q,j}) \\ \eta_{i,q,j} &= \sqrt{1-\cos^2(\theta_i)} \cdot \cos(\varphi_{q,j}) \end{aligned} \tag{5.40}$$

Nevertheless, in this method, one can apply the polar variable to any direction cosine, not only μ. If one uses η as the polar variable, the direction cosines are those of Equation 5.41. Likewise, if one uses ξ as the polar variable, the direction cosines are those of Equation 5.42.

$$\begin{aligned} \eta_{i,q,j} &= \cos(\theta_i) \\ \xi_{i,q,j} &= \sqrt{1-\cos^2(\theta_i)} \cdot \sin(\varphi_{q,j}) \\ \mu_{i,q,j} &= \sqrt{1-\cos^2(\theta_i)} \cdot \cos(\varphi_{q,j}) \end{aligned} \tag{5.41}$$

$$\begin{aligned} \xi_{i,q,j} &= \cos(\theta_i) \\ \eta_{i,q,j} &= \sqrt{1-\cos^2(\theta_i)} \cdot \sin(\varphi_{q,j}) \\ \mu_{i,q,j} &= \sqrt{1-\cos^2(\theta_i)} \cdot \cos(\varphi_{q,j}) \end{aligned} \tag{5.42}$$

5.3 Interpolation schemes for the face values

This section explains two interpolation schemes for calculating the face averaged values of the angular flux: linear-step and multislope second order upwind. The first one is used for calculating the face averaged values at the inner faces. The second one is used for calculating these values at the boundary faces.

5.3.1 Linear-step

The first scheme is based on a combination of a step method and a linear method. The step method is the same method as the upwind method described in Section 2.4, particularly in Figure 2.6. However, the author calls it *step*, because this is the classical name used in the Discrete Ordinates. The step method calculates the face averaged value of the angular flux with Equation 5.43. In this equation, $\psi_{g,m,i_{out}}$ is the value of the angular flux in cell i_{out}.

This cell is defined in Figure 5.2 and it is the cell from which the angular flux outgoes.

$$\psi_{g,m,j} = \psi_{g,m,i_{out}} \tag{5.43}$$

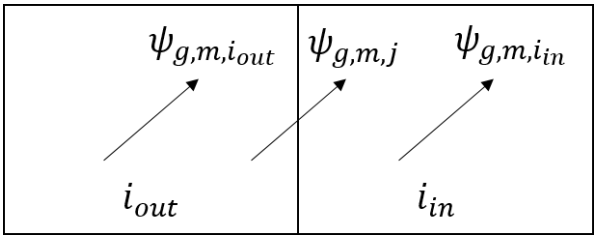


Figure 5.2: Definition of cells i_{out} and i_{in}

The step method has two main advantages. First, its simplicity. Second, its convergence. However, it has a major drawback: it overestimates the leakage terms. A more accurate approach is the linear method, which is the same method as the central difference scheme, which is defined in Figure 2.5 of Section 2.4. Equation 5.44 displays the calculation of $\psi_{g,m,j}$ by means of the linear method. In this equation, $r_{i,j}$ is the distance from the centroid of cell i to the centroid of face j. Although the linear method improves the accuracy, it has two main drawbacks. First, the convergence is worse than the step method. Second, it might provide results without physical sense, like negative values of the angular fluxes.

$$\psi_{g,m,j} = \frac{r_{i_{out},j}}{r_{i_{in},j} + r_{i_{out},j}} \psi_{g,m,i_{in}} + \frac{r_{i_{in},j}}{r_{i_{in},j} + r_{i_{out},j}} \psi_{g,m,i_{out}} \tag{5.44}$$

To deal with the advantages of the previous methods, the author defines a linear-step method, as shown in Equation 5.45. In this equation, δ is a value ranging from 0 to 1. The advantage of this method is that it gets the best of each method. The linear method improves the accuracy of the solution. The step method improves the convergence and avoid obtaining results without physical sense. The downside of the method is that it is difficult to set a general value of this coefficient δ. As a rule of thumb, one has to use the closest value to zero that provides results with physical sense. Moreover, if the calculation does not converge, one should also increase slightly this value of δ. The author recommends values in the following range: [0.005, 0.25].

$$\psi_{g,m,j} = \delta \cdot \psi^{step}_{g,m,j} + (1-\delta)\psi^{linear}_{g,m,j} = \frac{(1-\delta) r_{i_{out},j}}{r_{i_{in},j} + r_{i_{out},j}} \psi_{g,m,i_{in}} + \frac{r_{i_{in},j} + \delta \cdot r_{i_{out},j}}{r_{i_{in},j} + r_{i_{out},j}} \psi_{g,m,i_{out}} \tag{5.45}$$

5.3.2 Multislope second order upwind

Since the boundary faces do not have two adjacent cells, one cannot use the linear method, but one can use only the step method. However, the step method is first order accurate, so one should use another interpolation scheme, which was at least second order accurate.

A simple and second order accurate scheme is the second order upwind. This scheme calculates the face averaged value from the slope defined by the cell averaged values of two cells. The first cell is the adjacent cell to the face. The second cell is the adjacent cell to the first cell. This scheme is defined in Equation 5.46, where $p_{i,j}$ is the slope, i is the adjacent cell to the boundary face j, and $r_{i,j}$ is the distance between the centroids of the cell i and face j.

$$\psi_{g,m,j} = \psi_{g,m,i} + p_{i,j} r_{i,j} \tag{5.46}$$

Nevertheless, for 2D and 3D geometries, there are several adjacent cells to the cell i. For this reason, one should consider the information of all the neighbor cells. Therefore, this slope $p_{i,j}$ for certain cell i and face j should take into account the slope in all the adjacent cells to cell i. To do so, the author proposes calculating this slope with Equation 5.47. In this equation, i is the adjacent cell to face j, whereas i_{k_l} are the adjacent cells to cell i, n_{ac} is the number of adjacent cells to cell i, and $w_{i,j,l}$ are weights for calculating $p_{i,j}$ as a weighted sum of the slopes of each cell i_{k_l}. As $p_{i,j}$ takes into account the slopes of all the neighbor cells, the author called this scheme multislope second order upwind.

$$p_{i,j} = \sum_{l=1}^{n_{ac}} w_{i,j,l} \frac{\psi_{g,m,i} - \psi_{g,m,i_{k_l}}}{r_{i,i_{k_l}}} = \left(\sum_{l=1}^{n_{ac}} \frac{w_{i,j,l}}{r_{i,i_{k_l}}} \right) \psi_{g,m,i} - \sum_{l=1}^{n_{ac}} \frac{w_{i,j,l}}{r_{i,i_{k_l}}} \psi_{g,m,i_{k_l}} \tag{5.47}$$

As regards these weights $w_{i,j,l}$, one can calculate them to accomplish Equation 5.48. In this equation, $\vec{u}_{i,j}$ is the unit vector from the centroid of cell i to the centroid of face j, and $\vec{u}_{i_{k_l},i}$ is the unit vector from the centroid of cell i_{k_l} to the centroid of cell i. If one multiplies (dot product) Equation 5.48 and $\vec{u}_{i_{k_m},i}$, one obtains Equation 5.49. From this equation, one can build the linear system of

Equation 5.50. Since one can calculate all the dot products appearing in this linear system, one can determine $w_{i,j,l}$.

$$\vec{u}_{i,j} = \sum_{l=1}^{n_{ac}} w_{i,j,l} \vec{u}_{i_{k_l},i} \tag{5.48}$$

$$\left\langle \vec{u}_{i,j}, \vec{u}_{i_{k_m},i} \right\rangle = \sum_{l=1}^{n_{ac}} w_{i,j,l} \left\langle \vec{u}_{i_{k_l},i}, \vec{u}_{i_{k_m},i} \right\rangle \tag{5.49}$$

$$\begin{pmatrix} \left\langle \vec{u}_{i_{k_1},i}, \vec{u}_{i_{k_1},i} \right\rangle & \cdots & \left\langle \vec{u}_{i_{k_1},i}, \vec{u}_{i_{k_{n_{ac}}},i} \right\rangle \\ \vdots & \ddots & \vdots \\ \left\langle \vec{u}_{i_{k_{n_{ac}}},i}, \vec{u}_{i_{k_1},i} \right\rangle & \cdots & \left\langle \vec{u}_{i_{k_{n_{ac}}},i}, \vec{u}_{i_{k_{n_{ac}}},i} \right\rangle \end{pmatrix} \begin{pmatrix} w_{i,j,k_1} \\ \vdots \\ w_{i,j,k_{n_{ac}}} \end{pmatrix} = \begin{pmatrix} \left\langle \vec{u}_{i,j}, \vec{u}_{i_{k_1},i} \right\rangle \\ \vdots \\ \left\langle \vec{u}_{i,j}, \vec{u}_{i_{k_{n_{ac}}},i} \right\rangle \end{pmatrix} \tag{5.50}$$

Once one has calculated these weights, one can calculate $p_{i,j}$ by means of Equation 5.47. Then, if one combines Equations 5.46 and 5.47, one obtains Equation 5.51 for calculating $\psi_{g,m,j}$.

$$\psi_{g,m,j} = \left(1 + r_{i,j} \sum_{l=1}^{n_{ac}} \frac{w_{i,j,l}}{r_{i,i_{k_l}}} \right) \psi_{g,m,i} - \sum_{l=1}^{n_{ac}} \left(w_{i,j,l} \frac{r_{i,j}}{r_{i,i_{k_l}}} \right) \psi_{g,m,i_{k_l}} \tag{5.51}$$

Chapter 6

Results

6.1 Evaluation of the results

There are two important issues for evaluating the results. First, the accuracy of the solution. Second, the computational cost. The author will assess the computational cost by means of the computational time required for running the simulation. Furthermore, the author will assess the performance of the parallelization by means of the speedup. The speedup for N processors is defined as the ratio of the computational time with one processor to the computational time with N processors. In addition, the author ran the simulations on two different computers. Almost all the results were obtained on an AMD Opteron(TM) Processor 6272 with the CentOS 6.8 operating system. However, there are some cases in which the author ran the simulations on an Intel Core i7-3770 CPU (3.4GHz), with the CentOS 6.8 operating system. In these cases, the author will mention it.

Regarding the accuracy of the solution, the author will check the accuracy of the values of the crucial variables in Nuclear Safety Analyses, which are the multiplication factor and the power. On the one hand, the multiplication factor is the largest value of the eigenvalue $\mathbf{k}$. Since the method of this thesis is a modal method, the author calculated several eigenvalues, particularly the five largest eigenvalues in almost all the cases. On the other hand, the power

for each cell (P_i) is defined in Equation 6.1. The *constant* is a normalization factor to obtain Mean Power (MP) equals 1.0, which is defined in Equation 6.2.

$$P_i = constant \cdot \sum_{g=1}^{G} \Sigma_{f,g}^{i} \phi_{g,i} \tag{6.1}$$

$$MP = \frac{\sum_i |P_i| V_i}{\sum_i V_i} \tag{6.2}$$

To evaluate the accuracy of the eigenvalues and power, the author will use the Power Errors (PE) and Eigenvalue Errors (EE). These variables are defined in Equations 6.3 and 6.4. With the aim of reducing the extent of this document, one will also use the Mean Power Error (MPE) to assess the power results, and it is defined in Equation 6.5. In this thesis, one exhibits all the eigenvalue errors, but one only shows the power errors corresponding to the first eigenvector, due to the extent of the results.

$$PE_i(\%) = \frac{|P_i - P_{i,ref}|}{P_{i,ref}} \cdot 100 \tag{6.3}$$

$$EE(pcm) = \frac{\mathbf{k} - \mathbf{k}_{ref}}{\mathbf{k}_{ref}} \cdot 10^5 \tag{6.4}$$

$$MPE(\%) = \frac{\sum_i PE_i(\%)|P_i|V_i}{\sum_i |P_i| V_i} \tag{6.5}$$

In the previous equations, the values with subindex ref correspond to the values of a reference solution. The ideal reference solution is the analytical solution of the Neutron Diffusion or Transport Equations. However, one can obtain the analytical solution only for simple cases, like homogeneous reactors whose geometry is a rectangle or parallepiped. For general cases, one might also use reference solutions of benchmarks. Normally, one uses validated methods to obtain the results of these benchmarks. Consequently, the author also calculates the solution with other validated methods; thus, the author can check any case, including commercial nuclear reactors. Among the different validated methods, the author used three Neutron Diffusion Codes and one

Neutron Transport Code. The Neutron Diffusion codes are: PARCS (Downar et al. 2006), VALKIN (Verdú et al. 1994, Miró et al. 2002) and TRIVAC (Hébert and Sekki 2010). The Neutron Transport Code is TITAN (Yi 2009).

PARCS is a 3D reactor core simulator very well known and used. It solves the Steady State and time-dependent, multigroup Neutron Diffusion and low order Transport Equations in orthogonal and non-orthogonal geometries. PARCS only calculates the first eigenvalue. Regarding the spatial discretization, PARCS uses a Coarse Mesh Finite Difference Method (CMFD) to solve the Neutron Diffusion Equation in each node. These nodes are coupled at each inner face by the leakage terms. To calculate the coupling terms, PARCS has different numerical methods: Analytic Nodal Method (ANM), Nodal Expansion Method (NEM), a hybrid method combining ANM and NEM, and Finite Difference Method (FDM).

VALKIN is a Nodal Modal method that solves the 2-energy group Neutron Diffusion Equation, without upscattering and with fission neutrons produced in the first energy group. VALKIN solves this equation in Cartesian meshes by using the Nodal Collocation Method (Hébert 1987). VALKIN can calculate several eigenvalues and eigenvectors. In addition, for the time-dependent Neutron Diffusion Equation, VALKIN can use a Modal Method or a Quasi-Static Method.

TRIVAC is a computer code intended to compute the neutron flux in a fractional or in a full core representation of a nuclear reactor. It can solve the multigroup and multidimensional form of the Steady State and time-dependent Neutron Diffusion Equation or simplified P_n equation. Moreover, it allows the discretization of 1-D geometries (slab and cylindrical), 2-D geometries (Cartesian, cylindrical and hexagonal) and 3-D geometries (Cartesian and hexagonal). TRIVAC can apply different numerical methods: Finite Difference Method (FDM), Nodal Collocation Method (NCM) and Finite Element Method (FEM). In addition, TRIVAC can calculate several eigenvalues. Actually, the original version uses the Hotelling Deflation technique to do so, which is not the most efficient approach. However, the author of this thesis developed an algorithm for applying the Krylov-Schur method of SLEPc in TRIVAC, which was published in Bernal et al. 2017a. This Krylov-Schur method is very efficient for calculating several eigenvalues.

TITAN is a deterministic radiation transport simulation code in 3D Cartesian geometry. TITAN numerically solves the time-independent first order transport equation. It can solve the Transport Equation with the Discrete Ordinates method and the Method of Characteristics. For the Discrete Ordinates, it can

use two numerical methods for the spatial discretization: Diamond Difference and Directional Theta-Weighted. In addition, it only calculates one eigenvalue. TITAN has a parallel version using MPI, which only does angular decomposition. Regarding the quadrature sets, TITAN uses two types: level-symmetric and Legendre-Chebyshev.

The author also performed a sensitivity analysis of the following parameters: mesh, parameters of the FVM, linear system solvers and parameters of the modal method. First, the author used different structured and unstructured meshes with different sizes of the meshes.

Second, the sensitivity analysis of the parameters of the FVM depends on the type of FVM. For the Moving Least Squares method of Section 3.2.1, the parameter is the diffusion coefficient for the inner faces D_g^j. Regarding the polynomial expansion methods of Sections 3.2.2 and 3.2.3, the parameters are the polynomials $x^{\alpha_t} y^{\beta_t} z^{\gamma_t}$. With respect to the linear-step method of Section 5.3.1, the parameter is the coefficient δ.

Third, the author tested different linear system solvers and preconditioners of the PETSc library. In particular, the author tested the following linear system solvers: BiConjugate Gradient, GMRES, Generalized Conjugate Residual, BiCGSTAB and Conjugate Gradient Squared. The author used these solvers because they can be applied to non-symmetric matrices. These solvers were used with the following preconditioners: Jacobi, SOR and Additive Schwarz, which is the same as Incomplete LU for one processor. The author used the default tolerances of PETSc.

Finally, there are two important parameters of the modal method. First, the number of modes (M_d) of the expansion of the eigenvectors in Equation 4.24. Second, the time step for updating the modes.

6.2 Moving Least Squares method

In this section, the author tests the capabilities of the Moving Least Squares method of Section 3.2.1, to solve the Steady State Neutron Diffusion Equation, with two energy groups, without upscattering and with fission neutrons produced in the first energy group.

The author applied the method to 4 reactors: 2D and 3D homogeneous and heterogeneous. The homogeneous reactors are used to check the accuracy of calculation of the face averaged values. Then, the author tests the capability of

the diffusion coefficient D_g^j in heterogeneous reactors. The 2D heterogeneous reactor is Biblis. The 3D heterogeneous reactor is Langenbuch. In all the cases, five eigenvalues are calculated.

For solving the linear systems, the author used direct solvers, because the matrices were not well-conditioned. In particular, the author used the LU decomposition of MUMPS (Amestoy, Duff, and L'excellent 2000).

The results of this section were published in Bernal et al. 2014.

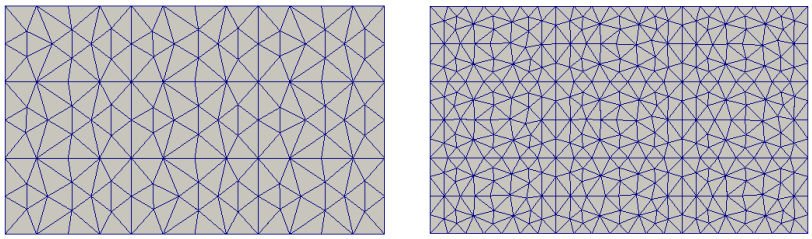
6.2.1 2D homogeneous reactor

This reactor is a rectangle of the following dimensions: 100 cm x 60 cm. It is composed of only one material, whose cross sections are those of Table 6.1.

Table 6.1: Cross sections of the homogeneous reactor

Group	$D_g\ (cm)$	$\Sigma_{a,g}\ (cm^{-1})$	$\nu\Sigma_{f,g}\ (cm^{-1})$	$\Sigma_{s,g\to g+1}\ (cm^{-1})$
1	1.28205128205	0.01	0.01	0.075
2	0.666667	0.1	0.109017634020268	

The author used six meshes for modeling this reactor, three structured and three unstructured. Mesh 1 is a structured mesh composed of 12 x 6 identical rectangles (72 rectangles). Mesh 2 is a structured mesh composed of 24 x 12 identical rectangles (288 rectangles). Mesh 3 is a structured mesh composed of 54 x 30 identical rectangles (1620 rectangles). Figure 6.1 shows Meshes 4 and 5, which have 270 and 732 triangles respectively. Figure 6.2 shows Mesh 6, which has 4208 triangles.



Mesh 4 Mesh 5

Figure 6.1: Meshes 4 and 5 for the 2D homogeneous reactor

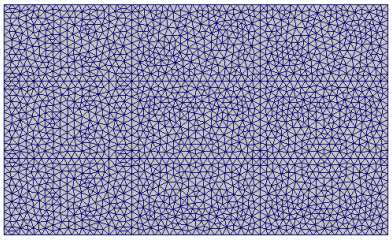


Figure 6.2: Mesh 6 for the 2D homogeneous reactor

Since this reactor is homogeneous, the choice of the diffusion coefficient D_g^j does not matter.

The author ran all the cases with boundary conditions of zero flux.

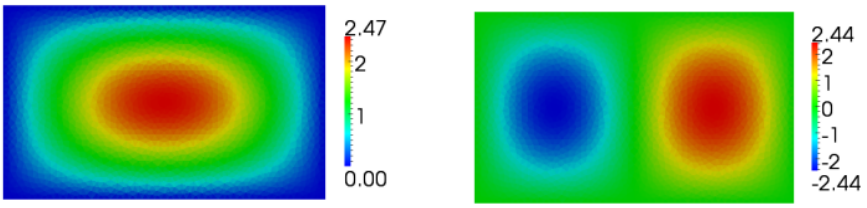
As this case has analytical solution, this is the reference solution. The reference solution is calculated in a structured mesh of 3 x 3 identical rectangles. The reference eigenvalues are: 1.000000, 0.943323, 0.859663, 0.854512 and 0.810300.

Table 6.2 shows the computational time, eigenvalue errors and maximum power errors for each mesh. One can see in this table that the calculation of the power and the first eigenvalue is accurate for any mesh, with power errors below 0.14 % and eigenvalue errors below 140 pcm. Nevertheless, one has to use fine meshes for obtaining accurate eigenvalue results for other eigenvalues.

Table 6.2: Results for the 2D homogeneous reactor: Computational time (s), eigenvalue errors (pcm) and maximum power error (%)

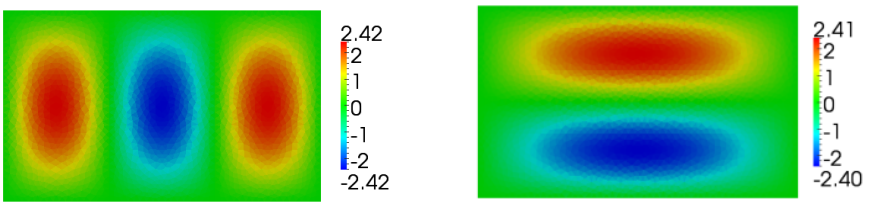
Mesh	Time	EE_1	EE_2	EE_3	EE_4	EE_5	$\max(PE_i)$
Mesh 1	0.38	139.90	309.79	1207.77	1587.05	1948.23	0.02
Mesh 2	0.85	35.08	77.40	246.02	462.01	492.74	0.00
Mesh 3	0.11	5.73	14.01	47.22	74.56	80.71	0.00
Mesh 4	0.74	29.65	117.77	437.26	221.29	436.13	0.10
Mesh 5	1.52	12.92	38.61	179.91	128.80	185.87	0.14
Mesh 6	10.41	2.05	6.12	18.74	20.95	29.28	0.02

Finally, Figures 6.3-6.5 show the power distribution for the five modes and Mesh 6.



First mode Second mode

Figure 6.3: Power (first and second modes) for the 2D homogeneous reactor with Mesh 6



Third mode Fourth mode

Figure 6.4: Power (third and fourth modes) for the 2D homogeneous reactor with Mesh 6

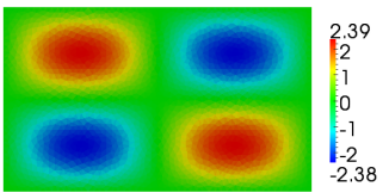


Figure 6.5: Power (fifth mode) for the 2D homogeneous reactor with Mesh 6

6.2.2 Biblis reactor

Biblis is a 2D heterogeneous reactor composed of 8 materials. Figure 6.6 shows its geometry and material composition. Table 6.3 exhibits its cross sections.

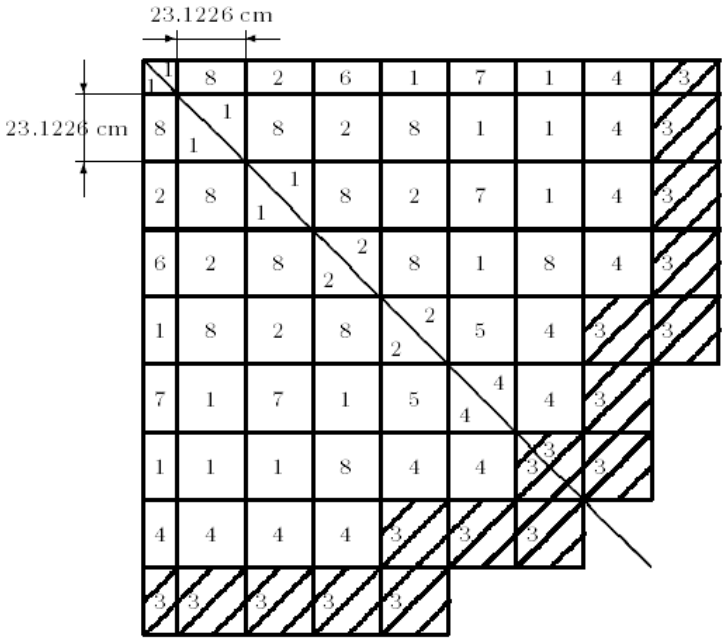


Figure 6.6: Biblis reactor

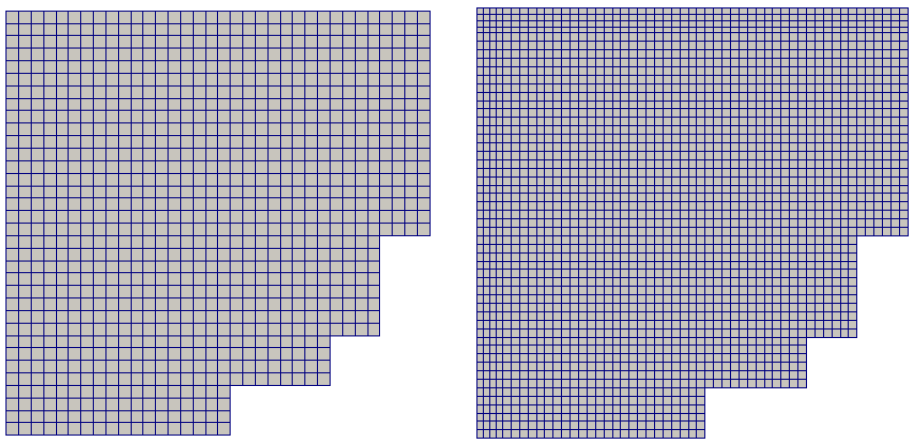
The author used six meshes for modeling this reactor, three structured and three unstructured. Figures 6.7-6.9 show these meshes. Meshes 1-3 are composed of 1028, 2416 and 9252 rectangles respectively. Meshes 4-6 are composed of 1808, 4542 and 23972 triangles respectively.

Furthermore, the author also performed a sensitivity analysis of the four diffusion coefficients D_g^j of Section 3.2.1, whose names are: Cell i, Cell l, Homogenized and Linear.

The author ran the simulations with the geometry of Figure 6.6, and consequently one has to use boundary conditions of zero flux at the east and south boundaries, but one has to use reflective conditions at the west and north boundaries. The author calculated five eigenvalues, but there are only reference results for the first eigenpair. These results are calculated for the nodes

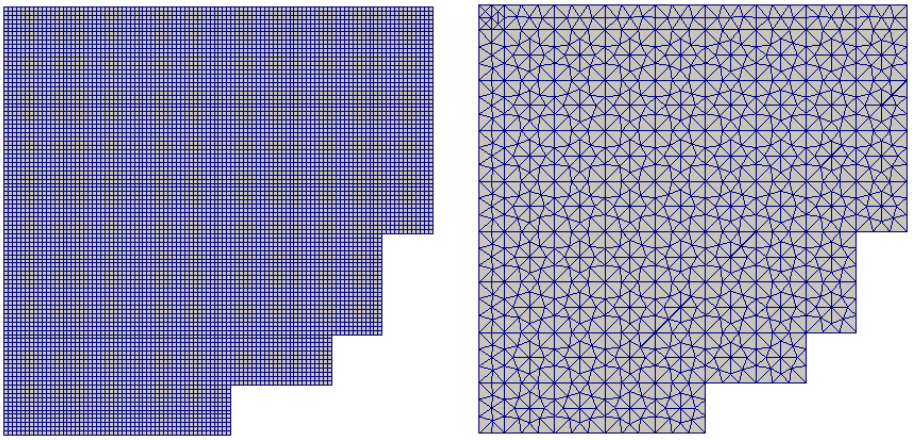
Table 6.3: Cross sections of Biblis reactor

Material	Group	$D_g\ (cm)$	$\Sigma_{a,g}\ (cm^{-1})$	$\nu\Sigma_{f,g}\ (cm^{-1})$	$\Sigma_{s,g\to g+1}\ (cm^{-1})$
1	1	1.4360	0.0095042	0.0058708	0.017754
	2	0.3635	0.0750580	0.0960670	
2	1	1.4366	0.0096785	0.0061908	0.017621
	2	0.3636	0.0784360	0.1035800	
3	1	1.3200	0.0026562	0.0000000	0.023106
	2	0.2772	0.0715960	0.0000000	
4	1	1.4389	0.0103630	0.0074527	0.017101
	2	0.3638	0.0914080	0.1323600	
5	1	1.4381	0.0100030	0.0061908	0.017290
	2	0.3665	0.0848280	0.1035800	
6	1	1.4385	0.0101320	0.0064285	0.017192
	2	0.3665	0.0873140	0.1091100	
7	1	1.4389	0.0101650	0.0061908	0.017125
	2	0.3679	0.0880240	0.1035800	
8	1	1.4393	0.0102940	0.0064285	0.017027
	2	0.3680	0.0905100	0.1091100	



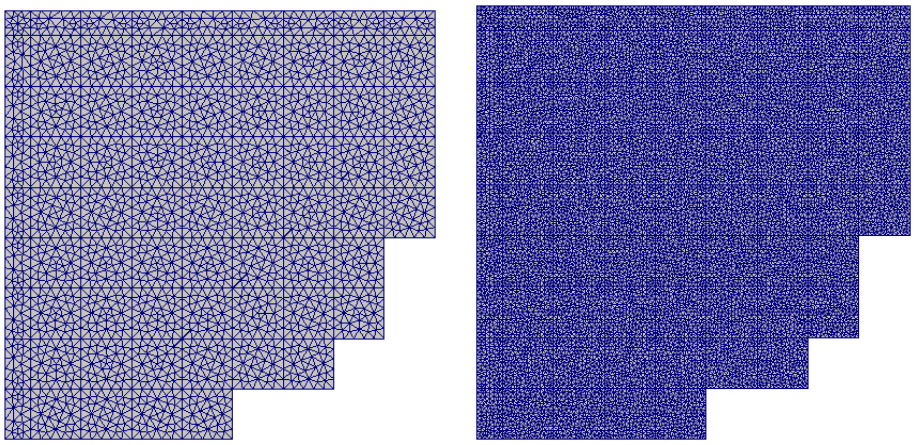
Mesh 1 Mesh 2

Figure 6.7: Meshes 1 and 2 for Biblis reactor



Mesh 3 Mesh 4

Figure 6.8: Meshes 3 and 4 for Biblis reactor



Mesh 5 Mesh 6

Figure 6.9: Meshes 5 and 6 for Biblis reactor

of Figure 6.6. One can find this reference solution in Müller and Weiss 1991, which was obtained by means of PANIC analytic nodal code using a 4 x 4 mesh. The reference eigenvalue is 1.025110.

As regards the results, Tables 6.4-6.6 show the computational time, eigenvalue errors and maximum power errors respectively. From Table 6.5 one draws two conclusions. First, any approach of D_g^j and mesh gives accurate eigenvalue results, since the eigenvalue errors are below 100 pcm. Second, there are slight differences among the results obtained with the different D_g^j. Furthermore, one draws the same conclusion from the power results of Table 6.6. However, these results show that one has to use fine meshes for obtaining maximum power errors below 2 %. As a conclusion, the author recommends using the Homogenized D_g^j and fine meshes, but not the finest ones. The reason is that the finest meshes increase drastically the computational time as shown in Table 6.4.

Table 6.4: Computational time (min:s) of Biblis reactor

D_g^j	Mesh 1	Mesh 2	Mesh 3	Mesh 4	Mesh 5	Mesh 6
Cell i	0:4	0:7	0:41	0:4	0:13	3:34
Cell l	0:3	0:7	0:41	0:4	0:13	3:43
Homogenized	0:3	0:7	0:41	0:5	0:13	3:35
Linear	0:3	0:7	0:41	0:4	0:13	3:44

Table 6.5: Eigenvalue error (pcm) of Biblis reactor

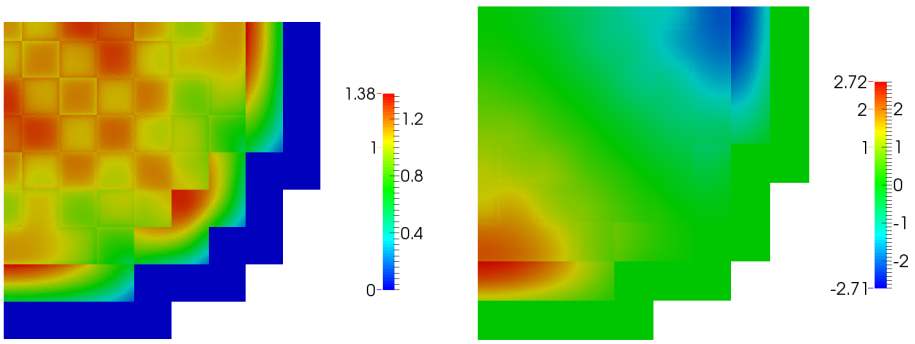
D_g^j	Mesh 1	Mesh 2	Mesh 3	Mesh 4	Mesh 5	Mesh 6
Cell i	97.88	42.21	11.21	49.52	19.92	4.54
Cell l	40.94	7.78	3.52	15.31	0.10	1.95
Homogenized	67.64	23.97	3.46	31.41	9.45	1.16
Linear	69.18	24.80	3.73	32.25	9.86	1.26

Table 6.6: Maximum power error (%) of Biblis reactor

D_g^j	Mesh 1	Mesh 2	Mesh 3	Mesh 4	Mesh 5	Mesh 6
Cell i	5.30	2.06	0.47	2.70	0.94	0.19
Cell l	1.70	1.80	1.17	1.16	1.26	0.53
Homogenized	3.22	0.86	0.63	1.41	0.60	0.29
Linear	3.33	0.86	0.61	1.47	0.57	0.28

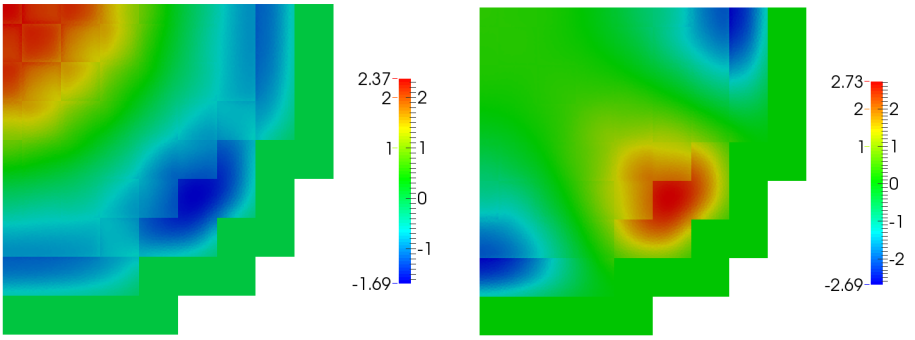
Finally, Figures 6.10-6.12 display the power for the five calculated eigenvectors, for Mesh 6 and using the Homogenized D_g^j. It is important to highlight

that these eigenvectors are not those corresponding to the largest eigenvalues, because the author did not simulate the whole geometry, but a quarter of it. However, the author did that for testing the reflective boundary condition.



First mode Second mode

Figure 6.10: Power (first and second modes) for Biblis reactor with Mesh 6 and Homogenized D_g^j



Third mode Fourth mode

Figure 6.11: Power (third and fourth modes) for Biblis reactor with Mesh 6 and Homogenized D_g^j

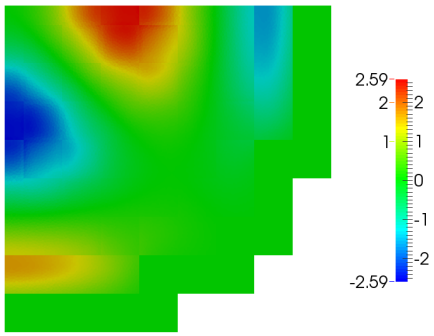


Figure 6.12: Power (fifth mode) for Biblis reactor with Mesh 6 and Homogenized D_g^j

6.2.3 3D homogeneous reactor

This reactor is a parallelepiped of the following dimensions: 100 cm x 60 cm x 180 cm. It is composed of only one material, whose cross sections are those of Table 6.1.

In this case, the author only used three structured meshes. Mesh 1 is composed of 10 x 6 x 18 identical parallelepipeds (1080 parallelepipeds). Mesh 2 is composed of 20 x 12 x 36 identical parallelepipeds (8640 parallelepipeds). Mesh 3 is composed of 50 x 30 x 90 identical parallelepipeds (135000 parallelepipeds).

As this reactor is homogeneous, the choice of the diffusion coefficient D_g^j does not matter.

The author ran all the cases with boundary conditions of zero flux.

Since this case has analytical solution, this is the reference solution. The reference solution is calculated for a structured mesh of 3 x 3 x 6 identical parallelepipeds. The reference eigenvalues are: 0.993919, 0.976030, 0.947343, 0.937787 and 0.921486

Table 6.7 shows the computational time, eigenvalue errors and maximum power errors for each mesh. One can conclude that the calculation of the power and the first eigenvalue is accurate for Meshes 2 and 3, with power errors below 0.29 % and eigenvalue errors below 103.76 pcm. Nevertheless, the computational time of Mesh 3 is extremely high, so the author recommends using Mesh 2.

Table 6.7: Results for the 3D homogeneous reactor: Computational time (h:min:s), eigenvalue errors (pcm) and maximum power error (%)

Mesh	Time	EE_1	EE_2	EE_3	EE_4	EE_5	$\max(PE_i)$
Mesh 1	0:0:9	148.49	174.64	276.89	390.55	416.48	1.07
Mesh 2	0:2:52	37.14	43.47	68.86	97.62	103.76	0.29
Mesh 3	12:41:9	5.93	6.91	10.94	15.56	16.50	0.05

Finally, Figures 6.13-6.15 show the power distribution for the five modes and Mesh 3.

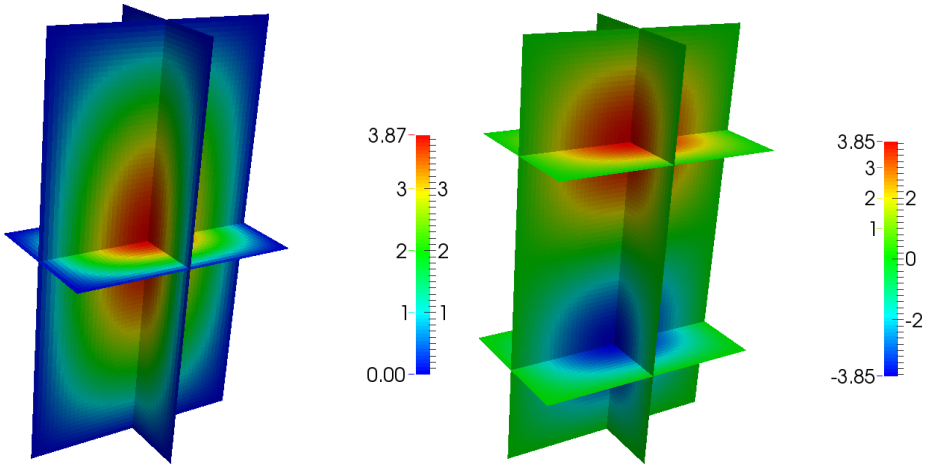
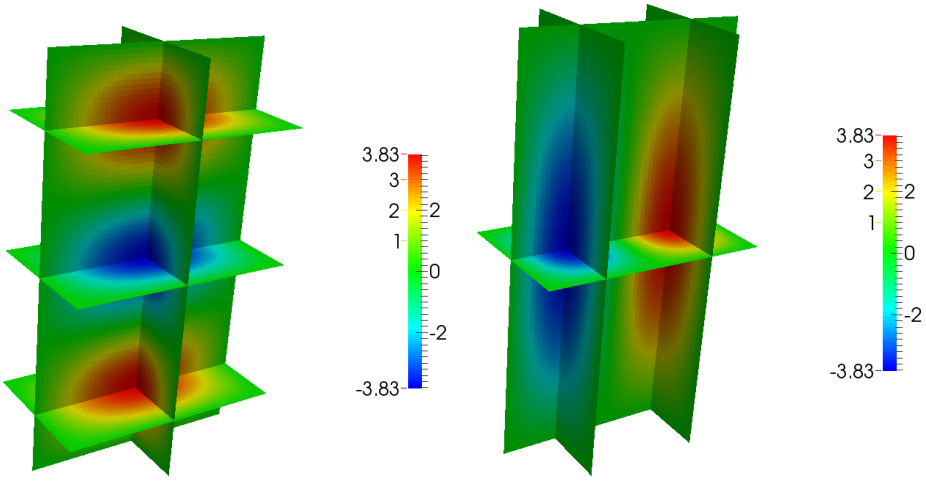


Figure 6.13: Power (first and second modes) for the 3D homogeneous reactor with Mesh 3

6.2.4 Langenbuch reactor

Langenbuch is a 3D heterogeneous reactor composed of 4 materials. Figure 6.16 displays its geometry and material composition. Table 6.8 shows its cross sections.

The author used two structured meshes for modeling this reactor. Figure 6.17 shows these meshes, which are composed of 2800 and 18720 parallelepipeds respectively.



Third mode Fourth mode

Figure 6.14: Power (third and fourth modes) for the 3D homogeneous reactor with Mesh 3

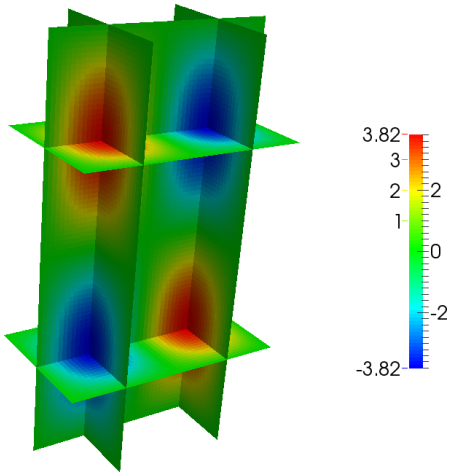


Figure 6.15: Power (fifth mode) for the 3D homogeneous reactor with Mesh 3

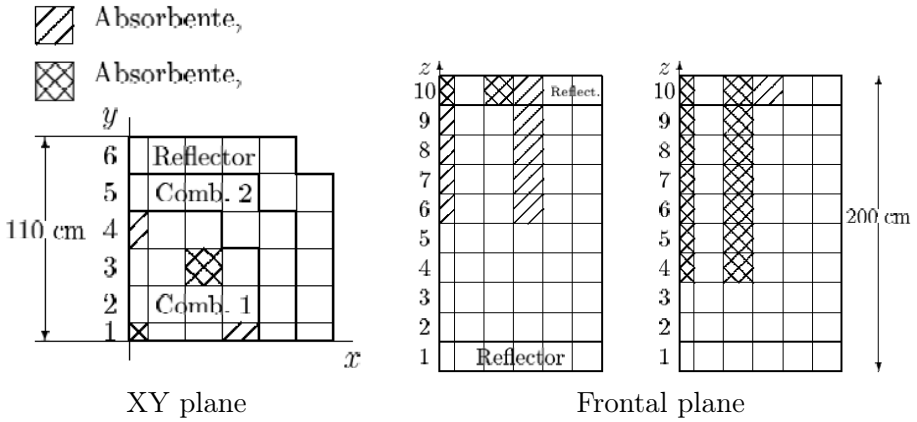
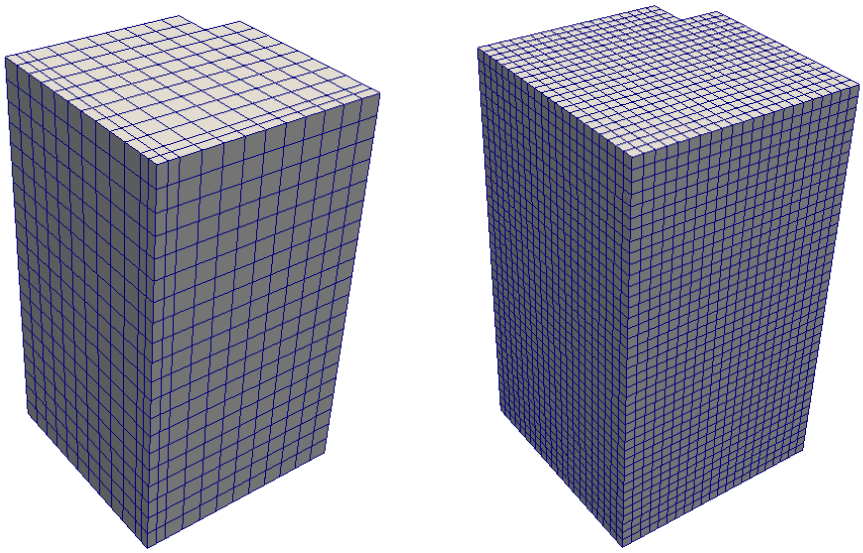


Figure 6.16: Langenbuch reactor

Table 6.8: Cross sections of Langenbuch reactor

Material	Group	$D_g\,(cm)$	$\Sigma_{a,g}\,(cm^{-1})$	$\nu\Sigma_{f,g}\,(cm^{-1})$	$\Sigma_{s,g\to g+1}\,(cm^{-1})$
Comb.1	1	1.423913	0.010402060	0.006477691	0.01755550
	2	0.356306	0.087662170	0.112732800	
Comb.2	1	1.425611	0.010992630	0.007503284	0.01717768
	2	0.350574	0.099256340	0.137800400	
Absorbent	1	1.423913	0.010952060	0.006477691	0.01755550
	2	0.356306	0.091462170	0.112732280	
Reflector	1	1.634227	0.002660573	0.000000000	0.02759693
	2	0.264002	0.049363510	0.000000000	



Mesh 1 Mesh 2

Figure 6.17: Meshes 1 and 2 for Langenbuch reactor

In addition, the author also performed a sensitivity analysis of the four diffusion coefficients D_g^j of Section 3.2.1.

The author ran the simulation with the geometry of Figure 6.16, and therefore one has to use boundary conditions of zero flux at the east, north, top and bottom boundaries, but one has to use reflective conditions at the west and south boundaries. The author calculated the reference solution with VALKIN code and obtained five eigenvalues. These results are calculated for the nodes of Figure 6.16. The reference eigenvalues are: 0.994881, 0.948211, 0.911892, 0.907632 and 0.877972.

Regarding the results, Table 6.9 shows the computational time; Tables 6.10-6.11 display the eigenvalue errors; Table 6.12 exhibits the maximum power errors (max(PE)) and mean power errors (MPE). From these tables, one concludes that only Mesh 2 gives accurate results, that is, MPE below 1 % and EE below 200 pcm for almost all the eigenvalues. Moreover, one can see that "Cell l" gives the worst results. As a conclusion, the author recommends using the Homogenized D_g^j and Mesh 2, although the computational time is about minutes, as shown in Table 6.9.

Table 6.9: Computational time (min:s) of Langenbuch reactor

D_g^j	Mesh 1	Mesh 2
Cell i	0:26	10:55
Cell l	0:26	11:24
Homogenized	0:26	10:56
Linear	0:26	10:44

Table 6.10: EE_1, EE_2 and EE_3 (pcm) of Langenbuch reactor

	EE_1		EE_2		EE_3	
D_g^j	Mesh 1	Mesh 2	Mesh 1	Mesh 2	Mesh 1	Mesh 2
Cell i	83.38	12.50	226.12	7.72	364.90	0.20
Cell l	212.85	63.06	350.52	80.21	768.15	242.30
Homogenized	143.15	22.69	281.00	39.99	556.71	115.28
Linear	150.09	25.26	292.19	44.24	574.25	121.93

Finally, Figures 6.18-6.20 display the power for the five calculated eigenvectors, for Mesh 2 and using the Homogenized D_g^j. It is important to highlight that these eigenvectors are not necessarily those corresponding to the largest eigenvalues, because the author did not simulate the whole geometry, but a

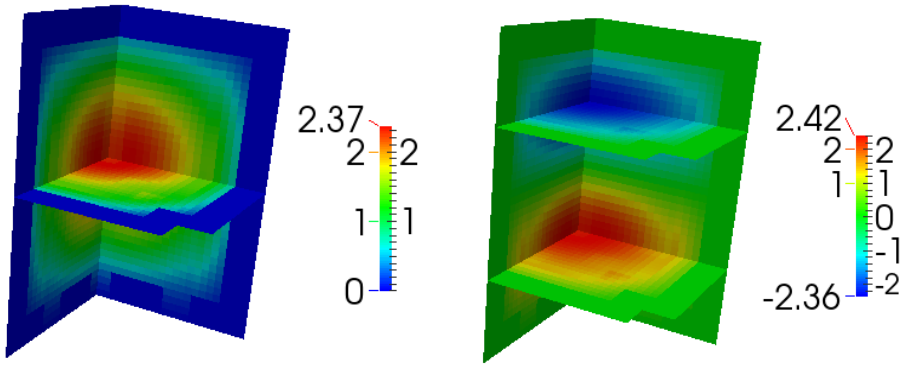
Table 6.11: EE_4 and EE_5 (pcm) of Langenbuch reactor

	EE_4		EE_5	
D_g^j	Mesh 1	Mesh 2	Mesh 1	Mesh 2
Cell i	357.08	8.73	584.10	67.89
Cell l	695.07	204.28	701.53	135.88
Homogenized	518.77	93.34	633.48	96.47
Linear	533.07	98.53	649.97	102.76

Table 6.12: Maximum Power Error and Mean Power Error (%) of Langenbuch reactor

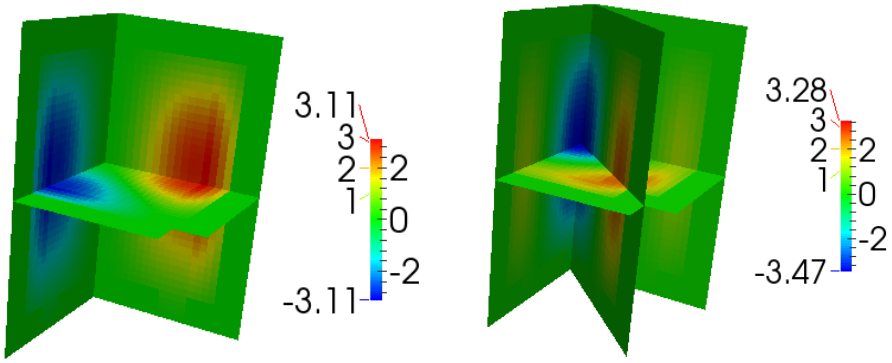
	max(PE)		MPE	
D_g^j	Mesh1	Mesh2	Mesh1	Mesh2
Cell i	10.28	3.59	2.23	0.63
Cell l	17.49	7.22	4.34	1.56
Homogenized	10.47	3.00	2.98	0.68
Linear	10.91	3.13	3.08	0.72

quarter of it. However, the author did that for testing the reflective boundary condition.



First mode Second mode

Figure 6.18: Power (first and second modes) for Langenbuch reactor with Mesh 2 and Homogenized D_g^j



Third mode Fourth mode

Figure 6.19: Power (third and fourth modes) for Langenbuch reactor with Mesh 2 and Homogenized D_g^j

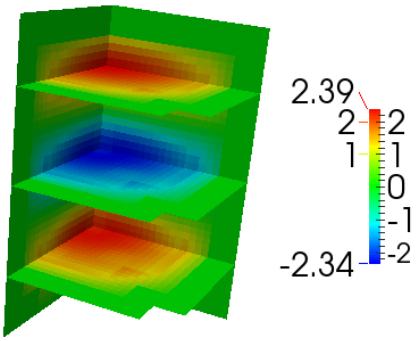


Figure 6.20: Power (fifth mode) for Langenbuch reactor with Mesh 2 and Homogenized D_g^j

6.3 Inter-cells polynomial expansion method

In this section, the author tests the capabilities of the inter-cells polynomial expansion method of Section 3.2.2, to solve the Steady State Neutron Diffusion Equation, with two energy groups, without upscattering and with fission neutrons produced in the first energy group.

The author applied the method to two 3D reactors, which are the same as those of the previous section. In both cases, one calculated five eigenvalues. For solving the linear systems, the author used the LU decomposition of MUMPS (Amestoy, Duff, and L'excellent 2000), because the matrices were not well-conditioned.

The results of this section were published in Bernal et al. 2016b.

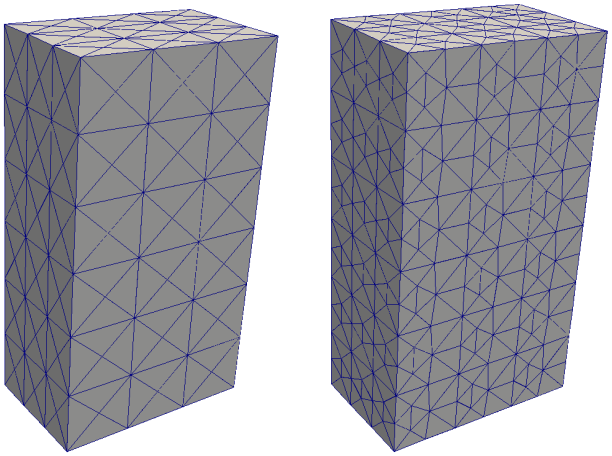
6.3.1 3D homogeneous reactor

Since this reactor is the same as that of Section 6.2.3, one can find the geometry and cross sections in the mentioned section.

In this case, the author used three structured and four unstructured meshes. Mesh 1 is a structured mesh composed of 3 x 3 x 6 identical parallelepipeds (54 parallelepipeds). Mesh 2 is a structured mesh composed of 6 x 6 x 12 identical parallelepipeds (432 parallelepipeds). Mesh 3 is a structured mesh composed of 12 x 12 x 24 identical parallelepipeds (3456 parallelepipeds). Figure 6.21 shows Meshes 4 and 5, which are composed of 1296 and 3402 tetrahedra respectively. Figure 6.22 shows Meshes 6 and 7, which are composed of 12146 and 5184 tetrahedra respectively.

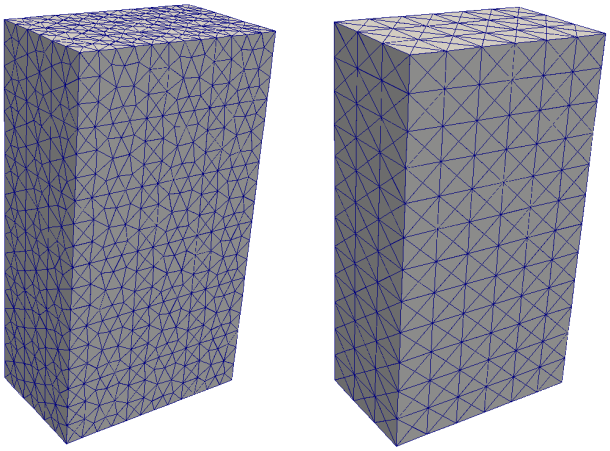
As discussed in Section 3.2.2, there are only several possible combinations of polynomials giving valid results. For meshes composed of hexahedra, there is only one combination giving valid results in this reactor: 1, x, y, z, x^2, y^2, z^2. In case of meshes composed of tetrahedra, there are three combinations providing valid results in this reactor. The first combination is 1, x, y, z, x^2; the second one is 1, x, y, z, y^2; and the third one is 1, x, y, z, z^2. In addition, the author tested other combinations with terms of higher orders and he realized that there is also a fourth combination giving valid and accurate results for coarse unstructured meshes: 1, x, y, z, x^2y^2.

The author ran all the cases with boundary conditions of zero flux.



Mesh 4 Mesh 5

Figure 6.21: Meshes 4 and 5 for the 3D homogeneous reactor



Mesh 6 Mesh 7

Figure 6.22: Meshes 6 and 7 for the 3D homogeneous reactor

As mentioned in Section 6.2.3, this case has analytical solution, so this is the reference solution. The reference power is calculated in Mesh 1. The reference eigenvalues are: 0.993919, 0.976030, 0.947343, 0.937787 and 0.921486.

Regarding the results of Meshes 1-3, Table 6.13 shows the computational time and eigenvalue errors. The power errors for these meshes are 0.00 %. From Table 6.13 one draws the following conclusion. Only Mesh 3 provides accurate eigenvalue results, with a low computational time.

Table 6.13: Results for the 3D homogeneous reactor with Meshes 1-3: Computational time (s) and eigenvalue errors (pcm)

Mesh	Time	EE_1	EE_2	EE_3	EE_4	EE_5
Mesh 1	1	755.70	957.66	1845.58	3427.56	3586.69
Mesh 2	1	215.84	265.33	483.48	1004.20	1043.10
Mesh 3	5	83.75	95.74	149.10	363.77	372.34

On the other hand, for Meshes 4-7, Table 6.14 shows the computational time, Tables 6.15-6.19 exhibit the eigenvalue errors and Table 6.20 contains the Mean Power Error for each mesh and polynomial set. In Table 6.14, one can see that the computational times are competitive and they hardly depends on the polynomial set. By contrast, one can see a significant effect of the polynomial set on the eigenvalue errors, for the coarsest mesh, but not for the finer ones. In addition, Mesh 4 obtains eigenvalue errors above 100-200 pcm, so one should use Meshes 5-7 to obtain accurate eigenvalue results for any polynomial set and mode. Finally, one can see excellent power results in Table 6.20, although there are some errors about 0.7 % for Meshes 5 and 6. However, these errors are very low and they are produced because of the fact that Meshes 5 and 6 are not symmetric.

Table 6.14: Computational time (s) of the 3D homogeneous reactor with Meshes 4-7

Polynomial set	Mesh 4	Mesh 5	Mesh 6	Mesh 7
Combination 1	2	3	13	5
Combination 2	2	3	13	5
Combination 3	2	3	13	5
Combination 4	2	4	15	6

Table 6.15: EE_1 (pcm) of the 3D homogeneous reactor with Meshes 4-7

Polynomial set	Mesh 4	Mesh 5	Mesh 6	Mesh 7
Combination 1	147.31	22.16	13.13	20.16
Combination 2	14.82	61.97	56.36	43.45
Combination 3	153.96	63.69	14.71	16.56
Combination 4	34.69	19.76	35.65	43.42

Table 6.16: EE_2 (pcm) of the 3D homogeneous reactor with Meshes 4-7

Polynomial set	Mesh 4	Mesh 5	Mesh 6	Mesh 7
Combination 1	275.18	62.78	1.63	11.70
Combination 2	77.27	37.28	50.88	8.31
Combination 3	215.23	53.89	5.75	2.81
Combination 4	117.50	4.71	25.26	10.06

Table 6.17: EE_3 (pcm) of the 3D homogeneous reactor with Meshes 4-7

Polynomial set	Mesh 4	Mesh 5	Mesh 6	Mesh 7
Combination 1	509.14	164.77	25.57	66.12
Combination 2	175.51	42.23	29.07	57.84
Combination 3	286.14	20.13	18.44	26.24
Combination 4	261.40	91.99	3.94	51.79

Table 6.18: EE_4 (pcm) of the 3D homogeneous reactor with Meshes 4-7

Polynomial set	Mesh 4	Mesh 5	Mesh 6	Mesh 7
Combination 1	127.74	162.62	168.48	148.89
Combination 2	58.43	17.57	99.89	63.93
Combination 3	416.60	187.86	14.80	69.07
Combination 4	29.04	84.49	142.15	152.58

Table 6.19: EE_5 (pcm) of the 3D homogeneous reactor with Meshes 4-7

Polynomial set	Mesh 4	Mesh 5	Mesh 6	Mesh 7
Combination 1	254.04	120.86	148.79	90.85
Combination 2	117.15	45.95	67.87	15.15
Combination 3	471.14	182.65	11.31	22.69
Combination 4	43.28	60.70	112.24	85.31

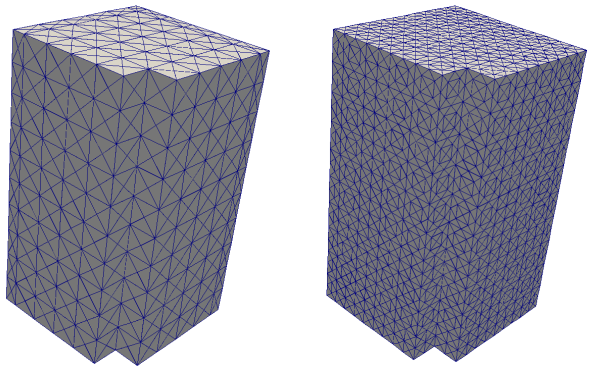
Table 6.20: MPE (%) of the 3D homogeneous reactor with Meshes 4-7

Polynomial set	Mesh 4	Mesh 5	Mesh 6	Mesh 7
Combination 1	0.00	0.19	0.06	0.00
Combination 2	0.00	0.18	0.05	0.00
Combination 3	0.00	0.30	0.12	0.00
Combination 4	0.00	0.70	0.26	0.00

6.3.2 Langenbuch reactor

As this reactor is the same as that of Section 6.2.4, one can find the geometry and cross sections in the cited section.

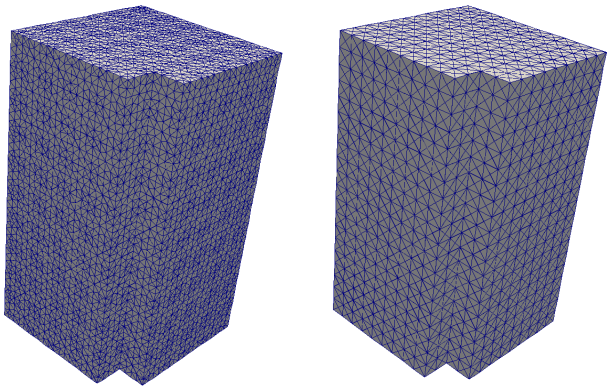
In this case, the author also used three structured and four unstructured meshes. Mesh 1 is the same mesh as that of Figure 6.16. One obtains Mesh 2 by subdividing each hexahedron of Mesh 1 in 2 x 2 x 2 identical hexahedra. Mesh 3 is obtained by subdividing each hexahedron of Mesh 1 in 4 x 4 x 4 identical hexahedra. Figure 6.23 shows Meshes 4 and 5. Figure 6.24 shows Meshes 6 and 7. Meshes 1-3 are composed of the following number of hexahedra: 350, 2800 and 22400. Meshes 4-7 are composed of the following number of tetrahedra: 8400, 41290, 111154 and 56160.



Mesh 4 Mesh 5

Figure 6.23: Meshes 4 and 5 for Langenbuch reactor

With respect to the polynomial set, one draws the same conclusion as in the previous section. Therefore, for meshes composed of hexahedra, only one combination of second order gave valid results: 1, x, y, z, x^2, y^2, z^2. In case of



Mesh 6 Mesh 7

Figure 6.24: Meshes 6 and 7 for Langenbuch reactor

meshes composed of tetrahedra, only three combinations of second order gave valid results. The first combination is 1, x, y, z, x^2; the second one is 1, x, y, z, y^2; and the third one is 1, x, y, z, z^2. Furthermore, the fourth combination used in the previous section also gives valid and accurate results for coarse unstructured meshes. This combination was the following: 1, x, y, z, x^2y^2.

As mentioned in Section 6.2.4, the author used boundary conditions of zero flux at the east, north, top and bottom boundaries, but one has to use reflective conditions at the west and south boundaries. The author calculated the reference solution with VALKIN and obtained five eigenvalues. These results are calculated for the nodes of Figure 6.16. The reference eigenvalues are: 0.994881, 0.948211, 0.911892, 0.907632 and 0.877972.

The results of Meshes 1-3 are exhibited in Table 6.21, which includes the computational time, eigenvalue errors and mean power errors. From this table, the finer the mesh, the more accurate the results. Although Mesh 3 is the most accurate, it is better to use Mesh 2, because the results are accurate enough with a computational time of 8 s. Moreover, Mesh 2 of this section is the same as Mesh 1 of Section 6.2.4. Therefore, if one compares the results of this method with those of the Moving Least Squares method, one draws three conclusions. First, this method is faster. Second, this method provides more accurate eigenvalue results. Third, this method gives more accurate power results.

Table 6.21: Results for Langenbuch reactor with Meshes 1-3: Computational time (s), eigenvalue errors (pcm) and mean power errors (%)

Mesh	Time	EE_1	EE_2	EE_3	EE_4	EE_5	MPE
Mesh 1	5	126.46	53.96	162.51	203.98	604.45	3.11
Mesh 2	8	9.94	32.14	119.07	132.74	250.26	0.58
Mesh 3	52	12.51	2.79	5.37	23.47	70.95	0.71

With respect to Meshes 4-7, Table 6.22 shows the computational time, Tables 6.23-6.27 exhibit the eigenvalue errors and Table 6.28 contains the Mean Power Error for each mesh and polynomial set. From these tables one draws four conclusions. First, the finer the mesh, the more accurate the results. Second, the finer the mesh, the less sensitivity to the polynomial set. Third, one obtains accurate results with all these meshes and polynomial sets, but Mesh 4 requires only 13 seconds of computational time. Fourth, in Mesh 4, the polynomial set has a significant effect on the eigenvalue errors for each mode. Actually, one can see that the eigenvalue errors for Mesh 4 and Combination 3 are below 100 pcm for modes 1, 2 and 5; whereas they are above 200 pcm for modes 3 and 4.

Table 6.22: Computational time (min:s) of Langenbuch reactor with Meshes 4-7

Polynomial set	Mesh 4	Mesh 5	Mesh 6	Mesh 7
Combination 1	0:13	1:27	6:23	2:31
Combination 2	0:14	1:26	6:21	2:30
Combination 3	0:13	1:24	6:13	2:24
Combination 4	0:13	1:27	6:28	2:32

Table 6.23: EE_1 (pcm) of Langenbuch reactor with Meshes 4-7

Polynomial set	Mesh 4	Mesh 5	Mesh 6	Mesh 7
Combination 1	58.02	42.78	26.36	26.80
Combination 2	58.07	41.86	26.46	26.85
Combination 3	62.92	41.74	22.57	22.70
Combination 4	45.42	47.33	27.80	28.14

Table 6.24: EE_2 (pcm) of Langenbuch reactor with Meshes 4-7

Polynomial set	Mesh 4	Mesh 5	Mesh 6	Mesh 7
Combination 1	111.19	78.03	35.59	33.71
Combination 2	111.32	77.39	35.98	33.68
Combination 3	61.13	42.50	22.58	25.41
Combination 4	99.87	92.57	39.60	35.94

Table 6.25: EE_3 (pcm) of Langenbuch reactor with Meshes 4-7

Polynomial set	Mesh 4	Mesh 5	Mesh 6	Mesh 7
Combination 1	169.41	143.20	79.58	74.93
Combination 2	169.53	137.74	79.15	74.83
Combination 3	261.70	178.97	84.27	75.33
Combination 4	99.62	137.95	79.49	70.41

Table 6.26: EE_4 (pcm) of Langenbuch reactor with Meshes 4-7

Polynomial set	Mesh 4	Mesh 5	Mesh 6	Mesh 7
Combination 1	129.56	110.06	57.25	56.66
Combination 2	129.64	105.80	57.17	56.49
Combination 3	238.94	169.11	64.72	56.28
Combination 4	75.53	126.64	58.82	53.83

Table 6.27: EE_5 (pcm) of Langenbuch reactor with Meshes 4-7

Polynomial set	Mesh 4	Mesh 5	Mesh 6	Mesh 7
Combination 1	192.40	158.16	57.53	39.44
Combination 2	192.54	159.45	58.52	39.44
Combination 3	19.38	22.44	8.57	13.59
Combination 4	185.22	193.63	65.37	44.36

Table 6.28: MPE (%) of Langenbuch reactor with Meshes 4-7

Polynomial set	Mesh 4	Mesh 5	Mesh 6	Mesh 7
Combination 1	1.32	1.13	0.92	0.92
Combination 2	1.32	1.11	0.92	0.92
Combination 3	1.54	1.19	0.86	0.84
Combination 4	1.07	1.25	0.95	0.98

6.4 Improved inter-cells polynomial expansion method

In this section, the author tests the capabilities of the improved inter-cells polynomial expansion method of Section 3.2.3, to solve the Steady State Neutron Diffusion Equation, with two energy groups, without upscattering and with fission neutrons produced in the first energy group.

In Section 6.3, the author proved that the polynomial expansion method is better than the Moving Least Squares method. The method of this section does not change the discretization of the FVM with respect to the polynomial expansion method, but improves the quality of the matrices. This improvement is based on two issues. First, the reduction of the size of the matrices. Second, the reduction of the condition number.

The author applied the method to a BWR reactor, but for two different cases. The first one without control rods and the second one with them inserted.

The results of this section were published in Bernal et al. 2016a.

6.4.1 BWR reactor without control rods

This is a commercial BWR reactor, whose geometry is displayed in Figure 6.25. This reactor is composed of 624 fuel assemblies and 503 different compositions. The reactor was modeled with 19980 nodes, which are cubes whose side length is 15.24 cm. The outer radial nodes and the nodes of the bottom and top axial planes model the reflector. Each node has different values of moderator density and fuel temperature, but there are not control rods. The values of the moderator densities, fuel temperatures and the corresponding cross sections were obtained by applying the SIMTAB methodology (Miró et al. 2006) developed by the UPV together with Iberdrola Ingeniería y Construcción (Iberinco), for the following conditions: 92% of full power, 66.3% of flow and burnup of 11.196 GWd/MT. However, the values of the cross sections are not included in this document, because the number of cross sections for this case is too high. In addition, the author also used ADFs obtained with the same methodology.

The author used four structured and one unstructured meshes. Mesh 1 is the same mesh as that of Figure 6.25. Mesh 2 is obtained by generating 2 x 2 x 2 identical hexahedra in each hexahedron of Mesh 1. Mesh 3 is obtained by generating 3 x 3 x 3 identical hexahedra in each hexahedron of Mesh 1. Mesh 4 is obtained by generating 4 x 4 x 4 identical hexahedra in each hexahedron of Mesh 1. Figure 6.26 shows Mesh 5.

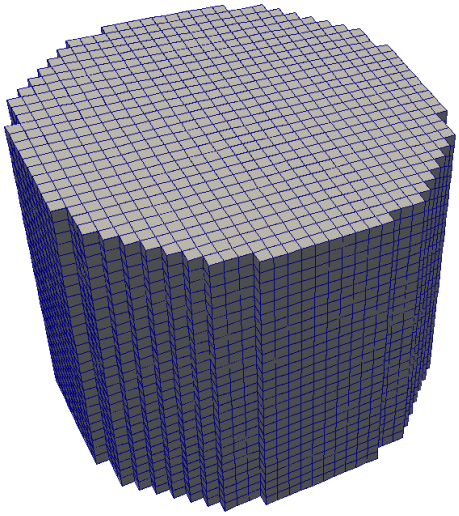


Figure 6.25: BWR reactor

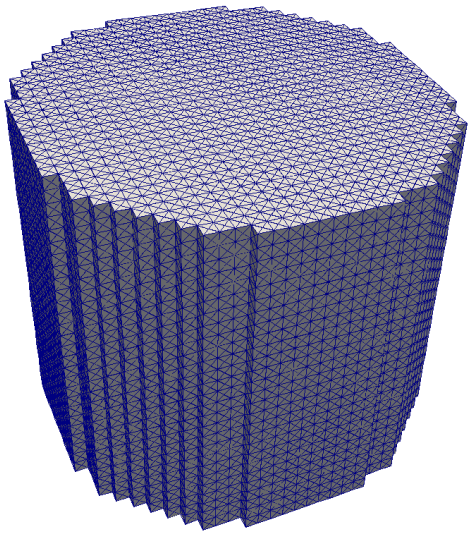


Figure 6.26: Mesh 5 of BWR reactor

As regards the polynomial set, one draws the same conclusion as in the previous sections. For meshes composed of hexahedra, only one polynomial set of second order gave valid results: 1, x, y, z, x^2, y^2, z^2. For meshes composed of tetrahedra, four polynomial sets gave valid results. Set 1 is 1, x, y, z, x^2; Set 2 is 1, x, y, z, y^2; Set 3 is 1, x, y, z, z^2; Set 4 is 1, x, y, z, x^2y^2.

The author used boundary conditions of zero flux. In addition, the reference solution was calculated with VALKIN code, obtaining five eigenvalues. The results are calculated for the nodes of Figure 6.25. The reference eigenvalues are: 1.021729, 1.010299, 1.010038, 1.004688 and 0.998619.

First, the author compares the matrices obtained with this method and with the polynomial expansion method. Table 6.29 displays the number of cells (N_c), the number of inner faces (N_f) and the number of boundary faces (N_b), for the different meshes. The last column of this table shows the reduction of the size of the matrices, due to the implicit definition of the current continuity and boundary conditions in this method.

Table 6.29: Geometrical elements of the meshes used for modeling BWR reactor

Mesh	N_c	N_f	N_b	$Reduction = \frac{N_c+N_f}{N_c+2N_f+N_b}$
1	19980	57580	4720	0.55
2	159840	470080	18880	0.56
3	539460	1587140	42480	0.57
4	1278720	3798400	75520	0.57
5	479520	949600	18880	0.60

Not only are these matrices of lower size, but they also have better quality, in spite of using the same polynomial expansion and equations. The quality of the matrices can be quantified by the condition number, which is shown in Table 6.30. Actually, this table shows the condition number of matrices $L_{1,1}$ and $L_{2,2}$ for the polynomial expansion method (PEM), the improved polynomial expansion method (IPEM) and the improved polynomial expansion method with preconditioner (IPEM-P), which is single-domain ASM in this case. Table 6.30 shows only the condition number for Mesh 1, which is similar for the other meshes, but is quickly calculated.

Since the matrices have a good condition number, the author used iterative solvers for solving the linear systems. In particular, the author used GMRES with ASM preconditioner.

Table 6.30: Condition number of the matrices of BWR reactor without control rods for Mesh 1

	Condition number of $L_{g,g}$		
Matrix	PEM	IPEM	IPEM-P
$L_{1,1}$	7654.40	107.80	1.56
$L_{2,2}$	1648.51	116.24	1.10

Regarding the results, Table 6.31 contains the computational times, eigenvalue errors and mean power errors for each mesh. One draws four conclusions from this table. First, all the meshes give excellent eigenvalue results, since the eigenvalue errors are below 20 pcm. Second, the mean power errors are also good for all meshes, but particularly for Meshes 2-5, with errors below 1 %. Third, Mesh 5 is almost insensitive to the polynomial set. Fourth, the computational times are acceptable for all meshes, except for Mesh 4, which is higher than 30 min.

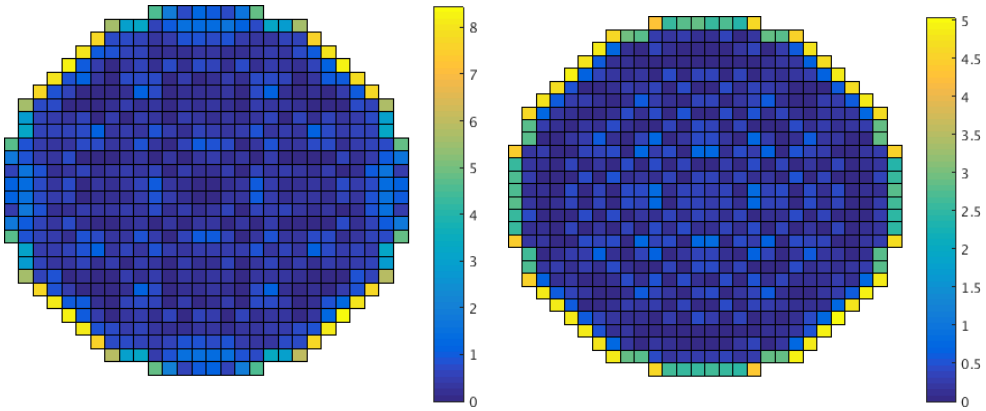
Table 6.31: Results for BWR reactor without control rods: Computational time (min:s), eigenvalue errors (pcm) and mean power errors (%)

Mesh	Time	EE_1	EE_2	EE_3	EE_4	EE_5	MPE
Mesh 1	0:17	4.6	11.14	10.75	9.55	11.17	1.55
Mesh 2	3:15	4.26	5.60	5.68	11.43	8.77	0.60
Mesh 3	10:53	2.04	2.43	2.47	4.80	3.43	0.24
Mesh 4	34:27	0.08	0.48	0.46	1.05	1.00	0.10
Mesh 5 (Set 1)	16:38	6.9	13.54	11.45	3.23	18.06	0.79
Mesh 5 (Set 2)	17:21	6.87	11.54	13.47	3.22	18.12	0.79
Mesh 5 (Set 3)	13:12	2.34	4.44	4.46	8.01	4.86	0.91
Mesh 5 (Set 4)	16:1	5.87	11.42	11.31	6.39	17.91	0.57

Moreover, the author shows the axial power errors in Table 6.32, and the radial power errors in Figures 6.27-6.30. From these errors, one realizes that Mesh 1 does not give accurate results, because the maximum axial and radial power error is about 9 %. However, Meshes 3 and 4 provide excellent results. With respect to Mesh 5, one can a see a significant effect of the polynomial set on the power distribution. On the one hand, Set 3 is the best one for the axial power distribution. On the other hand, Set 4 is the best one for the radial power distribution. As a conclusion, one should use meshes composed of hexahedra, because they provide the best results, due to the higher number of polynomial terms in the polynomial expansion for each cell.

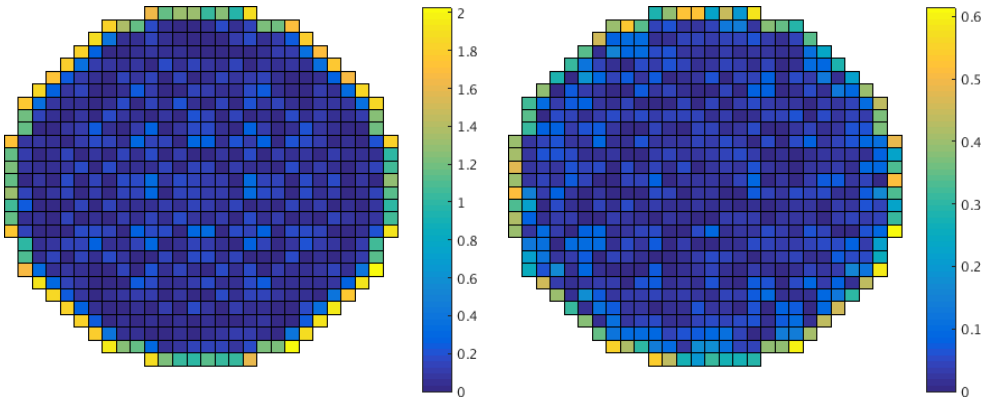
Table 6.32: Axial power errors (%) for BWR reactor without control rods

Axial level	Mesh 1	Mesh 2	Mesh 3	Mesh 4	Mesh 5 (Set 1)	Mesh 5 (Set 2)	Mesh 5 (Set 3)	Mesh 5 (Set 4)
26	1.53	0.84	0.32	0.15	1.71	1.71	1.88	1.19
25	4.19	1.87	0.81	0.41	2.67	2.67	2.00	2.00
24	2.82	1.14	0.51	0.24	1.50	1.50	1.91	0.87
23	2.58	1.09	0.43	0.19	1.44	1.44	1.85	0.82
22	2.86	1.14	0.48	0.22	1.61	1.58	1.78	1.00
21	2.53	0.96	0.40	0.18	1.35	1.35	1.64	0.81
20	2.39	0.89	0.36	0.17	1.32	1.32	1.53	0.81
19	2.35	0.83	0.34	0.15	1.25	1.25	1.40	0.79
18	2.24	0.77	0.31	0.14	1.21	1.21	1.28	0.80
17	1.94	0.64	0.25	0.11	1.04	1.04	1.10	0.68
16	1.78	0.54	0.22	0.10	0.93	0.93	0.96	0.64
15	1.72	0.52	0.21	0.10	0.91	0.91	0.80	0.67
14	1.39	0.37	0.15	0.08	0.71	0.71	0.64	0.52
13	1.16	0.29	0.12	0.06	0.60	0.60	0.48	0.48
12	1.00	0.24	0.10	0.06	0.52	0.52	0.32	0.44
11	0.71	0.12	0.06	0.04	0.35	0.35	0.16	0.34
10	0.43	0.03	0.02	0.03	0.22	0.22	0.01	0.25
9	0.18	0.05	0.01	0.01	0.07	0.07	0.16	0.15
8	0.10	0.12	0.05	0.01	0.07	0.07	0.32	0.05
7	0.47	0.23	0.09	0.03	0.27	0.27	0.48	0.10
6	0.91	0.35	0.14	0.06	0.52	0.52	0.63	0.31
5	1.46	0.48	0.20	0.09	0.86	0.86	0.75	0.63
4	1.74	0.58	0.24	0.12	1.23	1.23	0.85	0.99
3	2.10	0.20	0.09	0.11	1.04	1.04	0.98	0.80
2	8.77	2.72	1.18	0.63	3.05	3.05	1.23	2.80



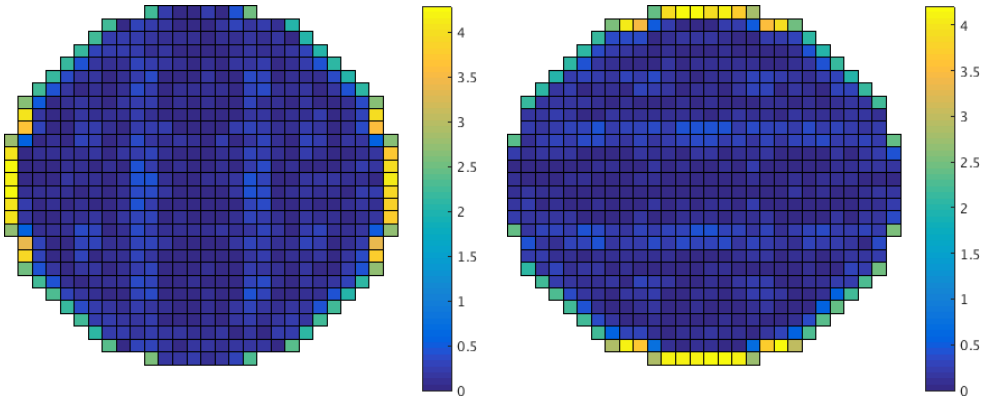
Mesh 1 Mesh 2

Figure 6.27: Radial power errors (%) of BWR reactor without control rods for Meshes 1 and 2



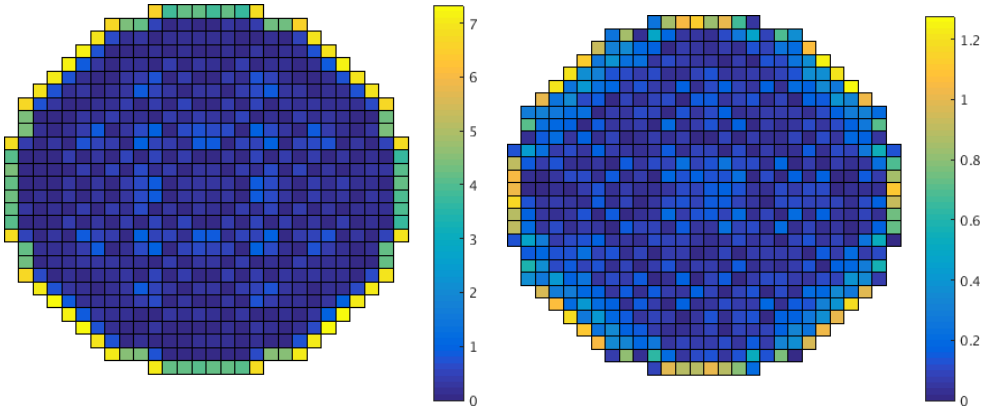
Mesh 3 Mesh 4

Figure 6.28: Radial power errors (%) of BWR reactor without control rods for Meshes 3 and 4



Set 1 Set 2

Figure 6.29: Radial power errors (%) of BWR reactor without control rods for Mesh 5 with polynomials sets 1 and 2



Set 3 Set 4

Figure 6.30: Radial power errors (%) of BWR reactor without control rods for Mesh 5 with polynomials sets 3 and 4

Finally, Figures 6.31-6.33 show the power distribution for the five modes, Mesh 5 and Set 4.

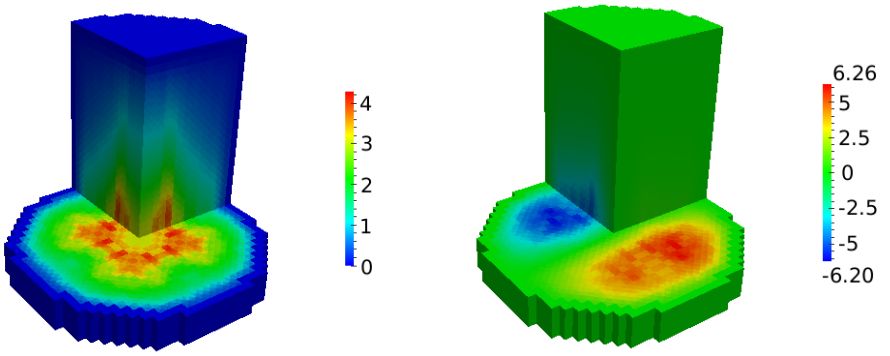


Figure 6.31: Power (first and second modes) for BWR reactor without control rods, for Mesh 5 with polynomials set 4

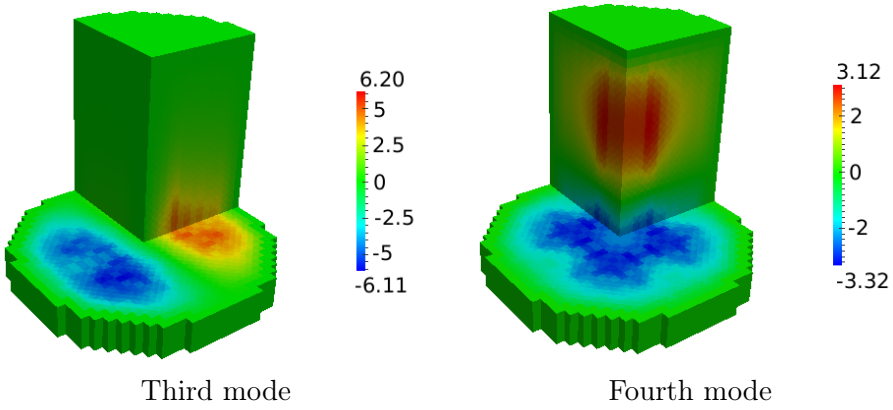


Figure 6.32: Power (third and fourth modes) for BWR reactor without control rods, for Mesh 5 with polynomials set 4

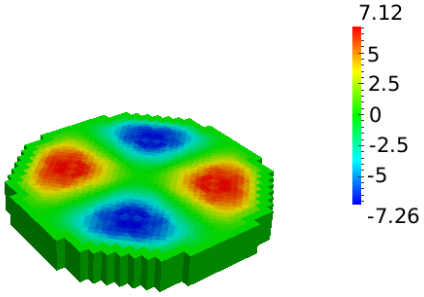


Figure 6.33: Power (fifth mode) for BWR reactor without control rods, for Mesh 5 with polynomials set 4

6.4.2 BWR reactor with control rods

This case is the same as the one of Section 6.4.1, but with the control rods of Figure 6.34. As in the previous case, the values of the cross sections are not included, because the number of cross sections is too high.

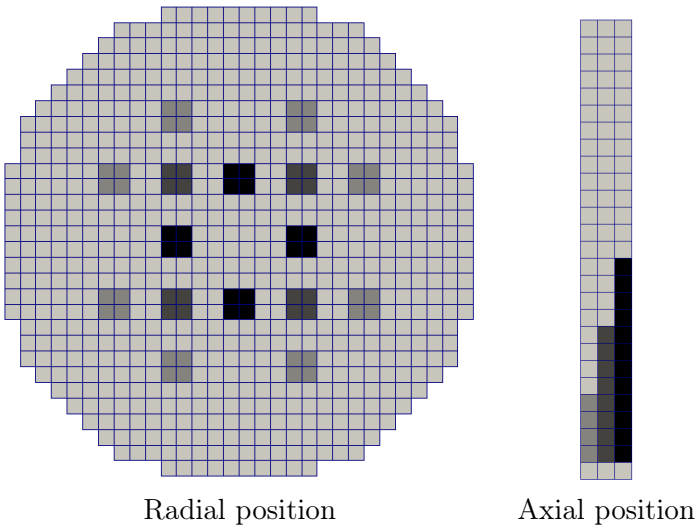


Figure 6.34: Control rod map of BWR reactor

Likewise, the author used the same meshes and polynomial sets as in Section 6.4.1.

The author also used boundary conditions of zero flux. Furthermore, the reference solution was calculated with VALKIN, obtaining five eigenvalues. The results are calculated for the nodalization of Figure 6.25. The reference eigenvalues are: 1.009375, 0.998101, 0.997857, 0.995140 and 0.986519.

As regards the results, Table 6.33 shows the computational times, eigenvalue errors and mean power errors. The computational times and eigenvalue errors are similar to those of the case without control rods. However, the mean power errors are higher. Moreover, one realizes that the mean power error decreases slightly with the refinement of the mesh.

Table 6.33: Results for BWR reactor with control rods: Computational time (min:s), eigenvalue errors (pcm) and mean power errors (%)

Mesh	Time	EE_1	EE_2	EE_3	EE_4	EE_5	MPE
Mesh 1	0:10	4.90	31.97	17.04	30.95	30.89	2.87
Mesh 2	2:49	3.78	6.00	7.52	5.12	1.06	1.76
Mesh 3	11:56	0.56	10.00	3.52	6.31	4.62	1.37
Mesh 4	37:49	3.60	13.82	0.31	8.56	9.93	1.26
Mesh 5 (Set 1)	17:35	6.53	24.72	11.72	24.16	27.78	1.81
Mesh 5 (Set 2)	19:51	6.52	25.05	11.25	24.03	27.79	1.85
Mesh 5 (Set 3)	19:40	2.43	19.91	6.71	21.52	18.52	2.35
Mesh 5 (Set 4)	16:37	7.28	23.40	9.89	20.39	26.55	1.46

In addition, one finds further details of the power results in Table 6.34 and Figures 6.35-6.38. Table 6.34 exhibits the axial power errors for the different meshes. From this table, one concludes that only Mesh 4 gives very accurate results, since it is the only one giving errors close to 1 %. Actually, one draws the same conclusion from the radial power errors, which are shown in Figures 6.35-6.38. Nevertheless, the radial power errors are higher than the axial ones and they are located close to the control rods. This might be due to the high gradients of the neutron flux at these locations.

Finally, Figures 6.39-6.41 show the power distribution for the five modes, Mesh 5 and Set 4.

Table 6.34: Axial power errors (%) for BWR reactor with control rods

Axial level	Mesh 1	Mesh 2	Mesh 3	Mesh 4	Mesh 5 (Set 1)	Mesh 5 (Set 2)	Mesh 5 (Set 3)	Mesh 5 (Set 4)
26	1.64	1.10	0.67	0.55	1.82	1.82	2.36	1.16
25	4.42	2.18	1.18	0.78	2.69	2.69	2.53	1.99
24	3.05	1.45	0.84	0.62	1.53	1.53	2.42	0.84
23	2.84	1.40	0.82	0.60	1.49	1.49	2.36	0.82
22	3.11	1.47	0.85	0.62	1.65	1.63	2.29	1.00
21	2.79	1.28	0.75	0.57	1.40	1.40	2.17	0.81
20	2.65	1.21	0.72	0.54	1.36	1.36	2.04	0.81
19	2.58	1.13	0.68	0.52	1.30	1.29	1.90	0.79
18	2.43	1.05	0.62	0.48	1.22	1.22	1.72	0.77
17	2.08	0.87	0.53	0.42	1.02	1.02	1.51	0.63
16	1.82	0.73	0.45	0.36	0.87	0.87	1.27	0.54
15	1.61	0.63	0.38	0.30	0.78	0.78	1.02	0.53
14	1.05	0.36	0.23	0.20	0.45	0.45	0.70	0.28
13	0.65	0.19	0.12	0.10	0.31	0.31	0.34	0.25
12	0.22	0.03	0.02	0.02	0.06	0.06	0.06	0.10
11	0.39	0.30	0.18	0.13	0.27	0.27	0.50	0.12
10	1.06	0.58	0.35	0.26	0.61	0.61	0.95	0.33
9	1.66	0.84	0.50	0.38	0.86	0.86	1.43	0.46
8	2.34	1.12	0.68	0.52	1.23	1.23	1.94	0.69
7	3.17	1.44	0.86	0.65	1.63	1.63	2.46	0.96
6	4.11	1.82	1.07	0.80	2.13	2.13	2.98	1.33
5	5.06	2.14	1.25	0.95	2.53	2.52	3.51	1.60
4	5.74	2.44	1.42	1.09	3.11	3.10	3.94	2.09
3	6.33	2.18	1.36	1.14	3.07	3.07	4.28	2.00
2	13.18	4.73	2.46	1.66	5.04	5.04	4.67	3.94

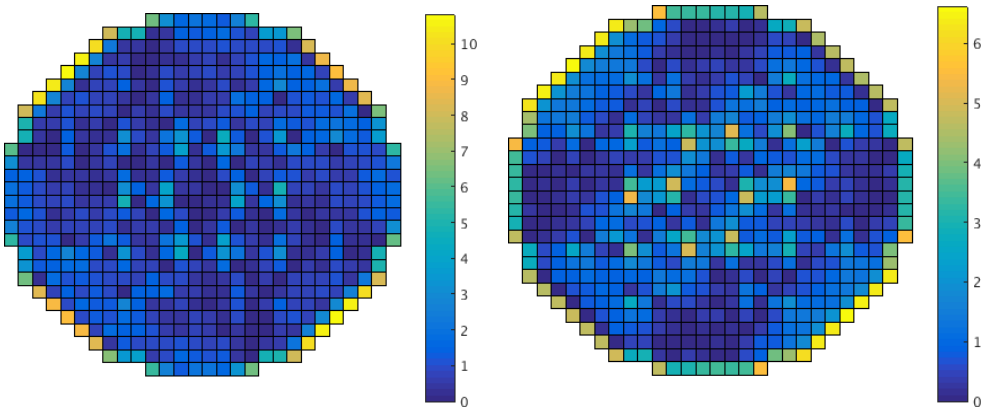


Figure 6.35: Radial power errors (%) of BWR reactor with control rods for Meshes 1 and 2

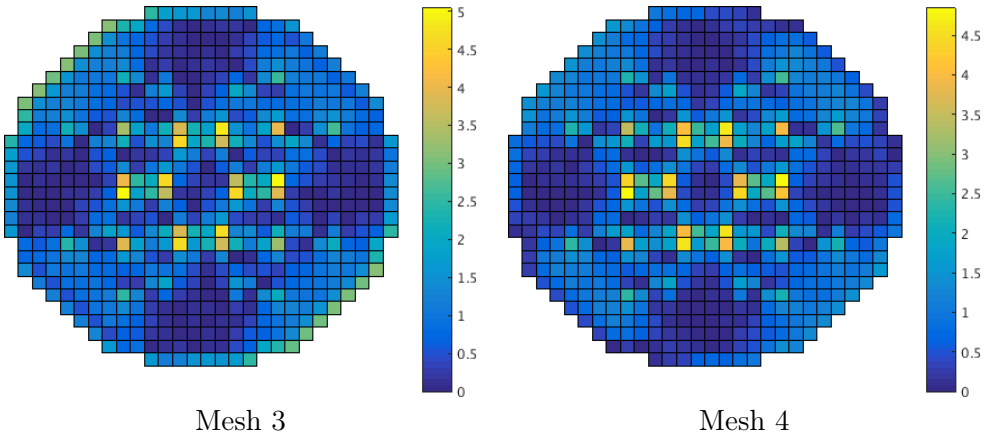
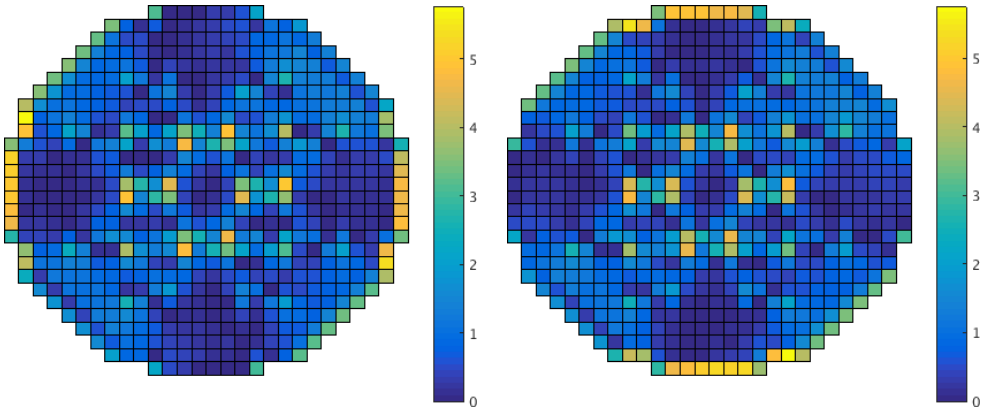
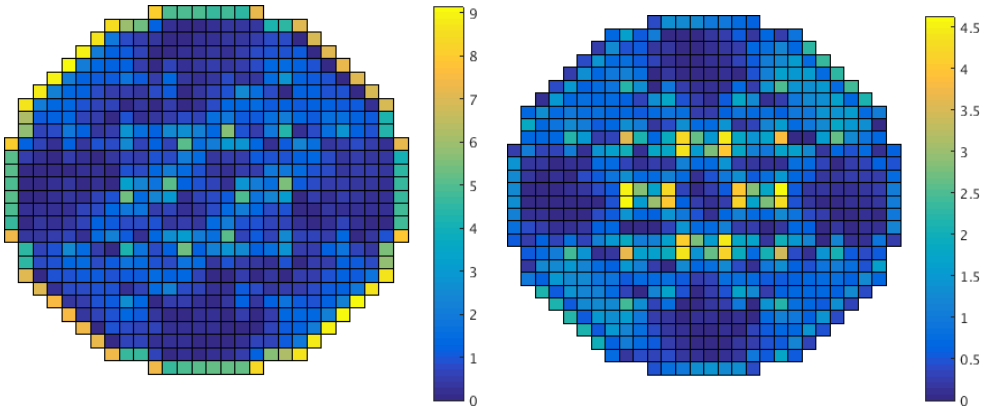


Figure 6.36: Radial power errors (%) of BWR reactor with control rods for Meshes 3 and 4



Set 1 Set 2

Figure 6.37: Radial power errors (%) of BWR reactor with control rods for Mesh 5 with polynomials sets 1 and 2



Set 3 Set 4

Figure 6.38: Radial power errors (%) of BWR reactor with control rods for Mesh 5 with polynomials sets 3 and 4

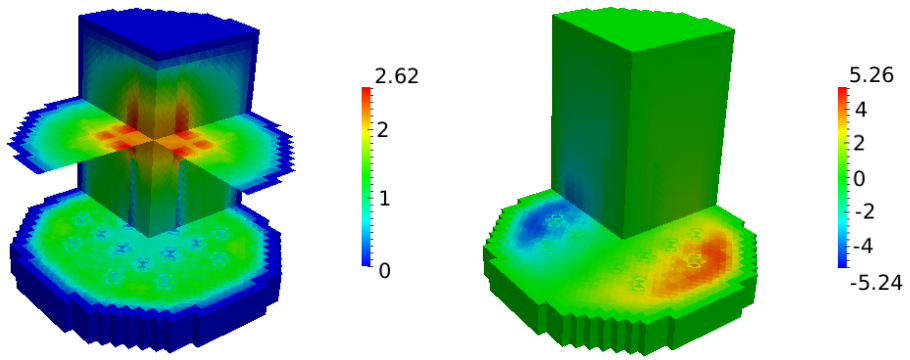


Figure 6.39: Power (first and second modes) for BWR reactor with control rods, for Mesh 5 with polynomials set 4

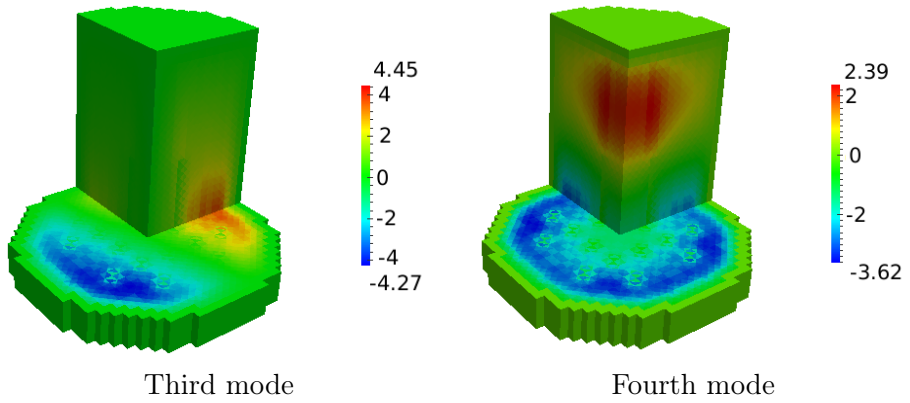


Figure 6.40: Power (third and fourth modes) for BWR reactor with control rods, for Mesh 5 with polynomials set 4

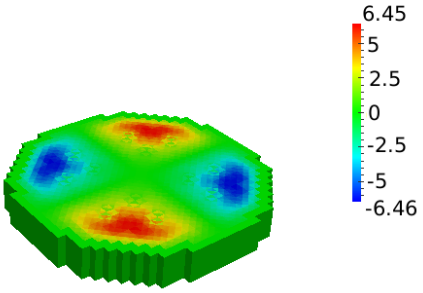


Figure 6.41: Power (fifth mode) for BWR reactor with control rods, for Mesh 5 with polynomials set 4

6.5 Multigroup formulation

This section shows the capabilities of the multigroup formulation, including upscattering and fission neutrons produced in several energy groups, to solve the Steady State Neutron Diffusion Equation. In particular, the author used the improved inter-cells polynomial expansion method of Section 3.2.3, because it is demonstrated that it is the best one.

The author tested the multigroup formulation in two MOX benchmarks. The first one is C5G7 MOX Benchmark (Lewis et al. 2001). Particularly, the author only tested the 2D benchmark. The second one is PWR MOX/UO_2 Core Transient Benchmark (Kozlowski and Downar 2007). This is a 3D benchmark including the transient state, yet the author only tested here the steady state.

The author ran the simulations of both cases on an Intel Core i7-3770 CPU (3.4 GHz), with the CentOS 6.8 operating system.

The results of this section were published in Bernal et al. 2017b.

6.5.1 2D C5G7 reactor

Figure 6.42 displays the geometry and composition of this benchmark (Lewis et al. 2001). The overall dimensions of Figure 6.42 are 64.26 cm × 64.26 cm, while each assembly is 21.42 cm × 21.42 cm. Each fuel assembly is made up of a 17 × 17 lattice of square pin level cells, as exhibited in Figures 6.43 and 6.44. The side length of each pin cell is 1.26 cm and all of the fuel pins and guide tubes have a 0.54 cm radius. A single moderator composition is given for use in all of the pin cells and for use in the water moderator (reflector) surrounding the assemblies. The composition layout for all four assemblies is shown in Figure 6.44. This benchmark uses 7 energy groups and the cross sections for each material are provided in Appendix A in Lewis et al. 2001. In addition, Figure 6.42 shows the boundary conditions.

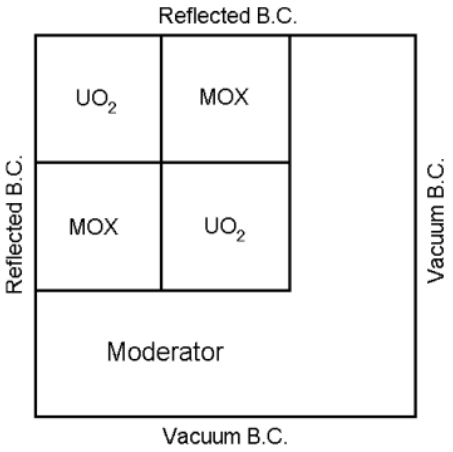


Figure 6.42: Core configuration for 2D C5G7 reactor

This benchmark was modeled with the mesh of Figure 6.45. Figure 6.46 displays a zoom of the mesh. Furthermore, Figure 6.46 shows that the fuel-clad circle was modeled as a regular hexadecagon. The radius of this polygon (R_p) was calculated to obtain the same area as the circle of radius (R_c) 0.56 cm, so the reaction rates are conserved. Equation 6.6 gives the analytical expression for R_p.

$$R_p = R_c \cdot \sqrt{\frac{\pi}{8 \sin\left(\frac{\pi}{8}\right)}} = 1.013 \cdot R_c \tag{6.6}$$

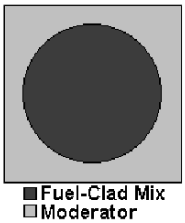


Figure 6.43: Pin cell geometry for 2D C5G7 reactor

Since the cells of these meshes are quadrangles, the polynomial expansion is limited to 5, as discussed in Section 3.2.2. There are six 2D monomials of order up to 2: 1, x, y, x^2, y^2 and xy. Thus, there are six possible 5-combinations of the set composed of these six monomials. The author tested these six combinations, and only one of them gave valid results: 1, x, y, x^2, y^2.

The reference results for this case are given in the benchmark (Lewis et al. 2001). The benchmark specifies that the reference results were obtained with MCNP code (Briesmeister Judith 2000), which is a Monte Carlo code.

Five modes were calculated, although there are only reference results for the fundamental mode (first mode). It is important to point out that this benchmark was solved by using direct solvers for the linear systems, since the matrices were ill-conditioned. Particularly, the author used the LU decomposition of the MUMPS library (Amestoy, Duff, and L'excellent 2000).

The computational time was 34 seconds. The five eigenvalues are: 1.182958, 0.903905, 0.859309, 0.702847 and 0.562519. These eigenvalues correspond to a quarter of the core, but they are not necessarily the five largest ones. Nonetheless, the author also ran the simulation of the full core, which required 2 minutes and 38 seconds, obtaining the following eigenvalues: 1.182958, 1.038667, 0.949189, 0.903905 and 0.859309. The eigenvalue error was 302.75 pcm, the mean power error was 1.44 % and the maximum power error was 4.04%. Furthermore, Figure 6.47 shows the power errors, where one can see high errors at the boundaries of MOX assemblies. These errors are acceptable, since the reference solution is obtained with a Monte Carlo method.

Finally, Figures 6.48-6.50 display the power for the different modes of the quarter of the reactor.

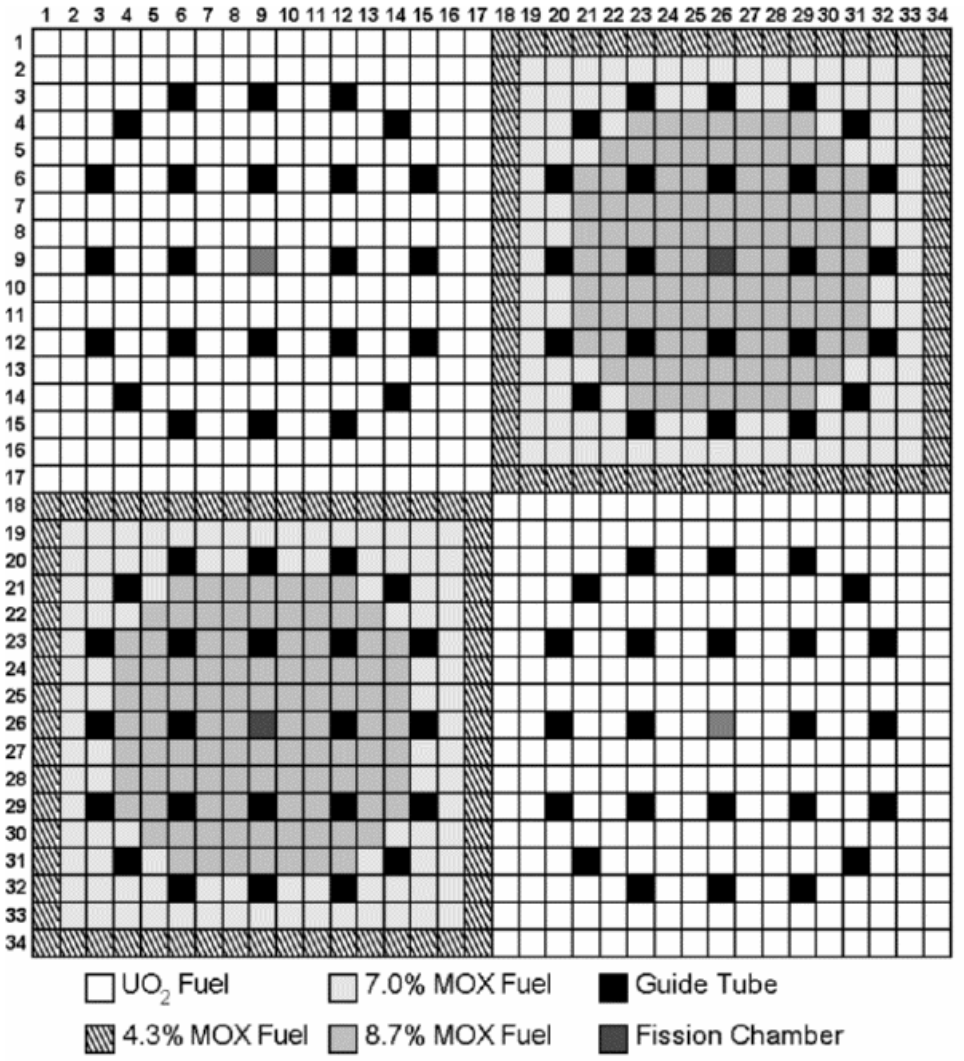


Figure 6.44: Pin cell compositions for 2D C5G7 reactor

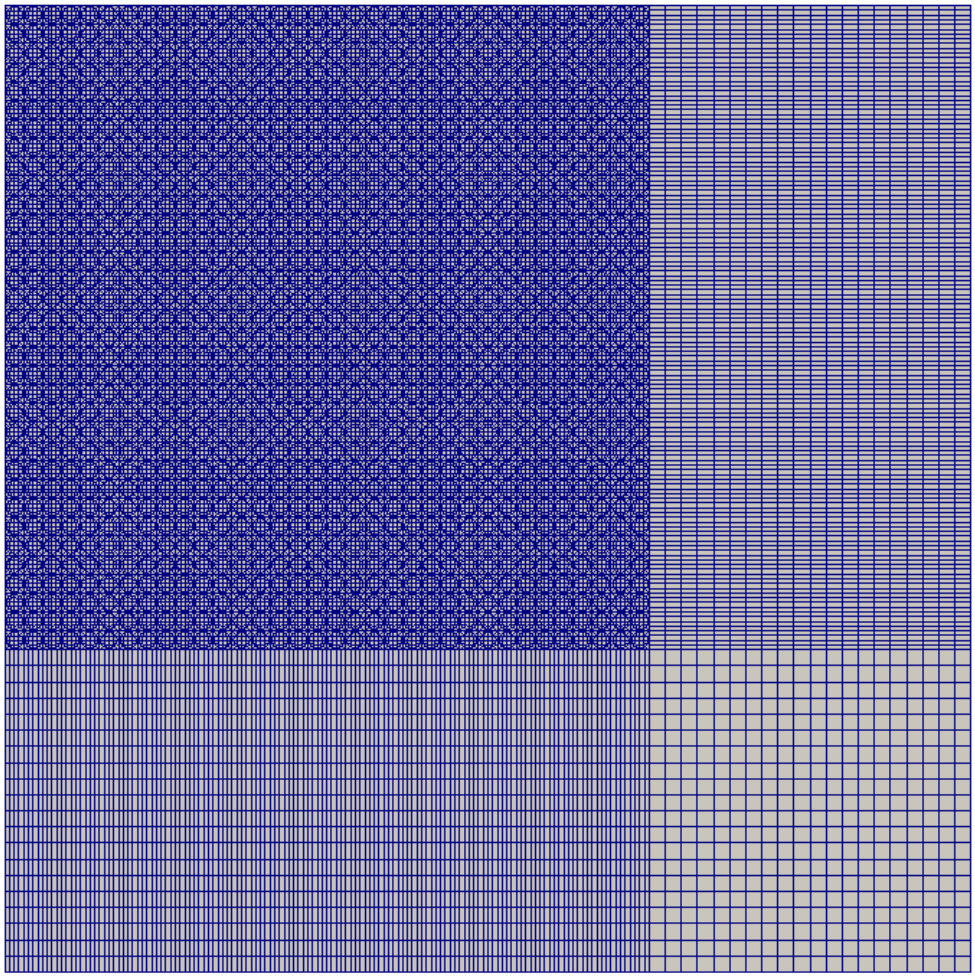


Figure 6.45: Mesh for 2D C5G7 reactor

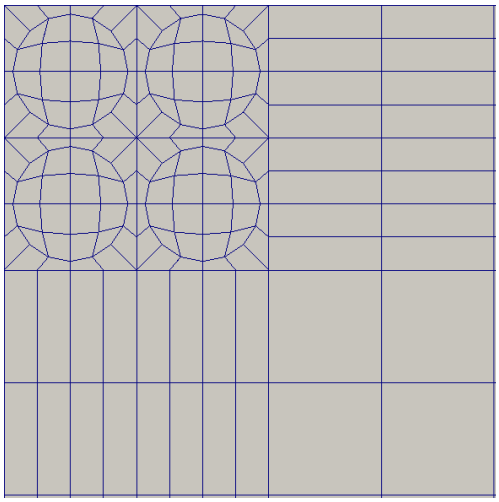


Figure 6.46: Zoom of the mesh for 2D C5G7 reactor

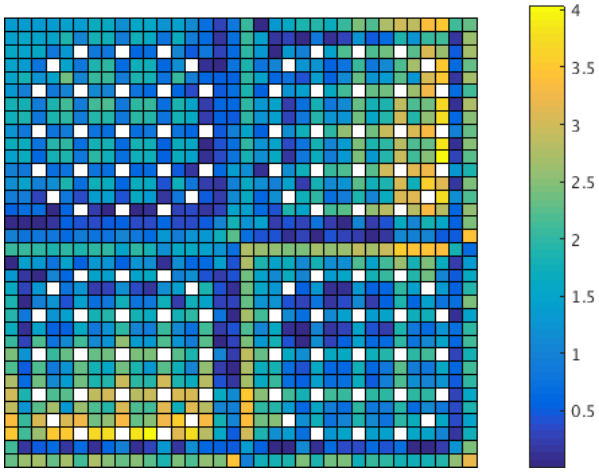


Figure 6.47: Power errors for 2D C5G7 reactor

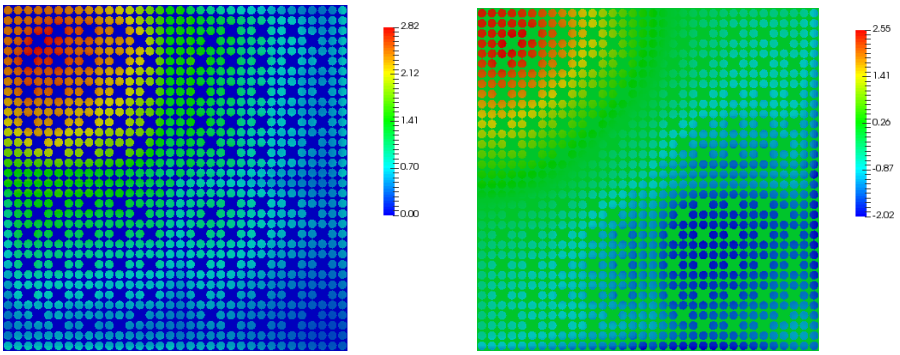


Figure 6.48: Power (first and second modes) for 2D C5G7 reactor

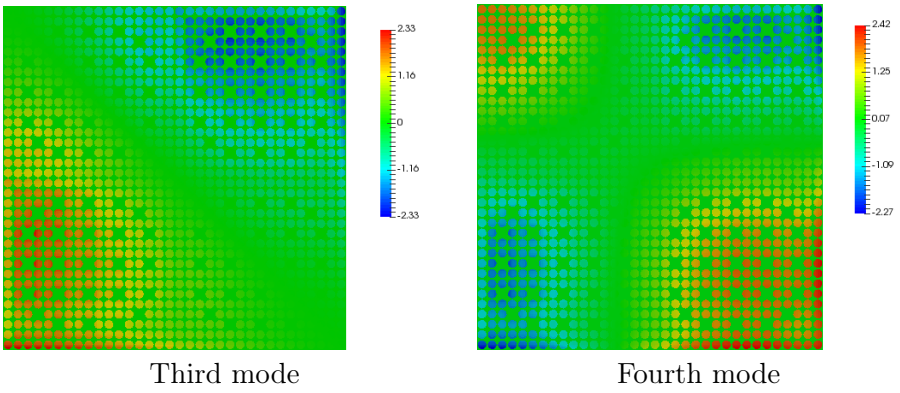


Figure 6.49: Power (third and fourth modes) for 2D C5G7 reactor

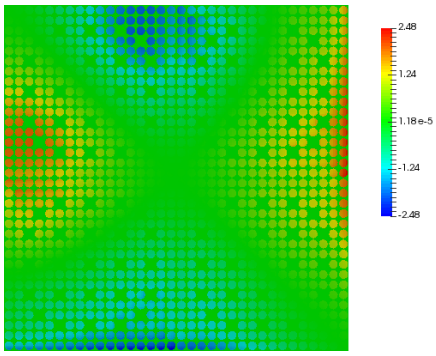


Figure 6.50: Power (fifth mode) for 2D C5G7 reactor

6.5.2 PWR MOX/UO2 reactor

This reactor is a PWR consisting of 224 fuel assemblies of different compositions and burnup (Kozlowski and Downar 2007). The active fuel length is 365.76 cm and was modeled with 20 axial levels. The assembly pitch is 21.42 cm. The reflector width is 21.42 cm. Figure 6.51 exhibits a quarter of the core. In this figure, the composition U 4.2 corresponds to UO_2 fuel type and enrichment of 4.2%. In contrast, M 4.0 corresponds to MOX fuel type and enrichment of 4.0%. One can find further details of the composition of these fuels in Kozlowski and Downar 2007. The core has uniform fuel composition in axial direction. The axial reflector has the same width as the fuel assembly pitch. Therefore, the model includes two more axial levels for the reflector, so it is modeled with 22 axial levels.

The benchmark has several sets of cross sections for 2, 4 and 8 energy groups and for different values of thermal-hydraulic conditions. In this case, the author used the set of 8 energy groups and the following thermal-hydraulic conditions: inlet coolant temperature of 560 K and inlet pressure of 15.5 MPa. From Figure 6.51, one concludes that there are 18 different sets of cross sections, so the author did not include them in this document due to the extent of the data. However, it is important to point out that there are upscattering terms in energy groups 6 and 7. With respect to the ADFs, the author set a value of 1.0.

On the other hand, the author used three meshes for modeling this reactor. Mesh 1 is the same as that of Figure 6.52. Mesh 2 is obtained by generating 2

U 4.2 35.0	U 4.2 0.15	U 4.2 22.5	U 4.5 0.15	U 4.5 37.5	M 4.3 17.5	U 4.5 0.15	U 4.2 32.5	REF
U 4.2 0.15	U 4.2 17.5	U 4.5 32.5	M 4.0 22.5	U 4.2 0.15	U 4.2 32.5	M 4.0 0.15	U 4.5 17.5	REF
U 4.2 22.5	U 4.5 32.5	U 4.2 22.5	U 4.2 0.15	U 4.2 22.5	M 4.3 17.5	U 4.5 0.15	M 4.3 35.0	REF
U 4.5 0.15	M 4.0 22.5	U 4.2 0.15	M 4.0 37.5	U 4.2 0.15	U 4.5 20.0	M 4.3 0.15	U 4.5 20.0	REF
U 4.5 37.5	U 4.2 0.15	U 4.2 22.5	U 4.2 0.15	U 4.2 37.5	U 4.5 0.15	U 4.2 17.5	REF	REF
M 4.3 17.5	U 4.2 32.5	M 4.3 17.5	U 4.5 20.0	U 4.5 0.15	M 4.3 0.15	U 4.5 32.5	REF	
U 4.5 0.15	M 4.0 0.15	U 4.5 0.15	M 4.3 0.15	U 4.2 17.5	U 4.5 32.5	REF	REF	
U 4.2 32.5	U 4.5 17.5	M 4.3 35.0	U 4.5 20.0	REF	REF	REF		
REF	REF	REF	REF	REF	Assembly Type Burnup (Gwd/t)			

Figure 6.51: A quarter of the core for PWR MOX/UO_2 reactor

x 2 x 2 identical hexahedra in each hexahedron of Mesh 1. Mesh 3 is obtained by generating 3 x 3 x 3 identical hexahedra in each hexahedron of Mesh 1.

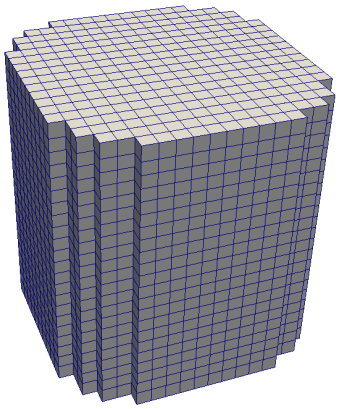


Figure 6.52: Mesh for PWR MOX/UO_2 reactor

As regards the polynomial expansion, the author tested the 120 possible 7-combinations of polynomial sets of order 2, but only one of them gave valid results: 1, x, y, z, x^2, y^2, z^2.

The boundary conditions for this case are zero flux. Moreover, the author calculated the reference results with TRIVAC code (Hébert and Sekki 2010). TRIVAC can solve the neutron diffusion equation with different methods, as

mentioned in Section 6.1. In this section, the author used the Nodal Collocation Method (Hébert 1987), with Legendre polynomials of order three. Actually, the author used a version of TRIVAC based on its version 5, which includes an eigensolver based on the SLEPc library (Bernal et al. 2017a). The mesh used in TRIVAC is that of Figure 6.52. The computational time for TRIVAC calculation was 1 min and 38 s and the five eigenvalues are: 1.149530, 1.137884, 1.135261, 1.135259 and 1.124268.

Regarding the results, Table 6.35 shows the computational time, eigenvalue errors and mean power errors for each mesh. In this table, one can see excellent results for all meshes, although the finer the mesh, the more accuracy. It is also important to highlight that the computational time for Mesh 2 is similar to the computational time of TRIVAC.

Table 6.35: Results for PWR MOX/UO_2 reactor: Computational time (min:s), eigenvalue errors (pcm) and mean power errors (%)

Mesh	Time	EE_1	EE_2	EE_3	EE_4	EE_5	MPE
1	0:8	39.85	42.25	34.63	34.80	36.81	1.55
2	1:18	4.45	0.45	8.63	8.55	17.80	1.25
3	5:30	2.89	1.57	2.35	2.23	5.70	0.43

One can find further details of the power results in Table 6.36 and Figures 6.53 and 6.54. On the one hand, Table 6.36 exhibits the axial power errors for each mesh. All meshes give accurate axial power results. Nonetheless, one can see in this table that Mesh 2 gives higher errors than Mesh 1 at axial levels 2 and 21, though these errors are two or three times lower at other axial levels. This might be due to round-off errors, because round-off errors have more effect on power errors for low power values than for high ones. In this case, the power results of TRIVAC in axial levels 2 and 11 are 0.2461 and 1.4849. A variation of the power of 0.005 in axial levels 2 and 11 corresponds to a percentage variation of 2.03% and 0.34% respectively. Therefore, this justifies the effect of round-off errors on axial levels 2 and 21.

On the other hand, Figures 6.53 and 6.54 show the radial power errors for each mesh. These figures might look a little bit asymmetrical due to the fact that results obtained with TRIVAC are a little asymmetrical, though they should not be asymmetrical. A possible reason for this little asymmetry could be round-off errors when one calculates cell averaged values. From these figures,

Table 6.36: Axial power errors (%) for PWR MOX/UO_2 reactor

Axial level	Mesh 1	Mesh 2	Mesh 3
21	1.66	2.44	0.86
20	1.43	0.20	0.06
19	0.87	0.09	0.02
18	0.34	0.01	0.00
17	0.06	0.04	0.01
16	0.12	0.06	0.02
15	0.24	0.09	0.03
14	0.32	0.11	0.04
13	0.36	0.12	0.04
12	0.39	0.12	0.04
11	0.39	0.13	0.04
10	0.36	0.12	0.04
9	0.31	0.11	0.04
8	0.23	0.10	0.04
7	0.11	0.07	0.03
6	0.07	0.04	0.01
5	0.35	0.01	0.01
4	0.87	0.09	0.01
3	1.43	0.19	0.06
2	1.67	2.43	0.85

one draws the following conclusion: only Mesh 3 gives accurate results, with power errors close to 1 %.

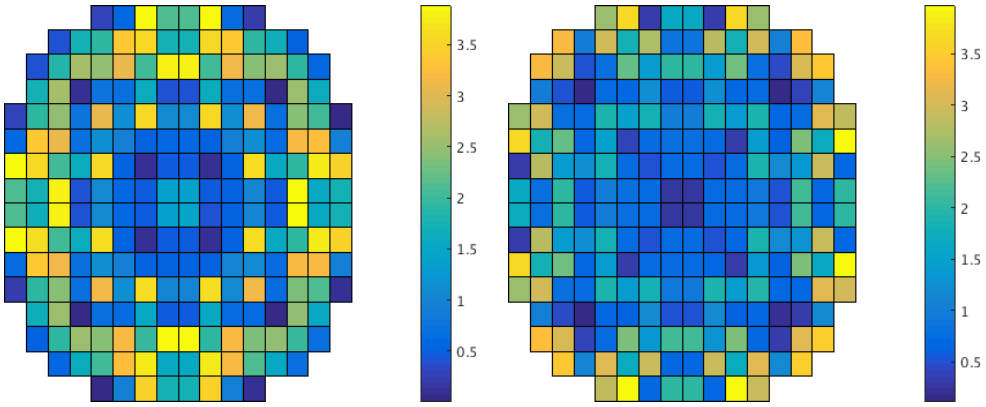


Figure 6.53: Radial power errors (%) of PWR MOX/UO_2 reactor for Meshes 1 and 2

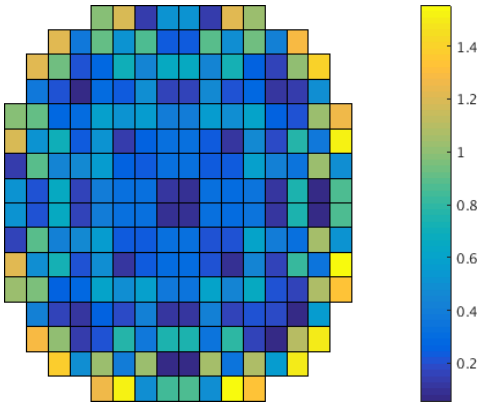
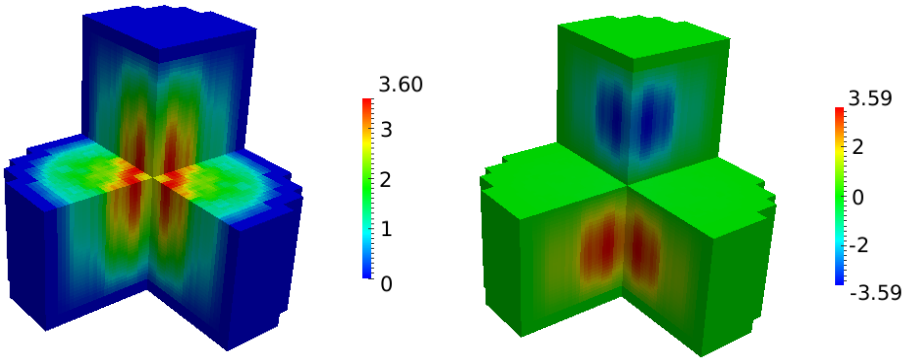


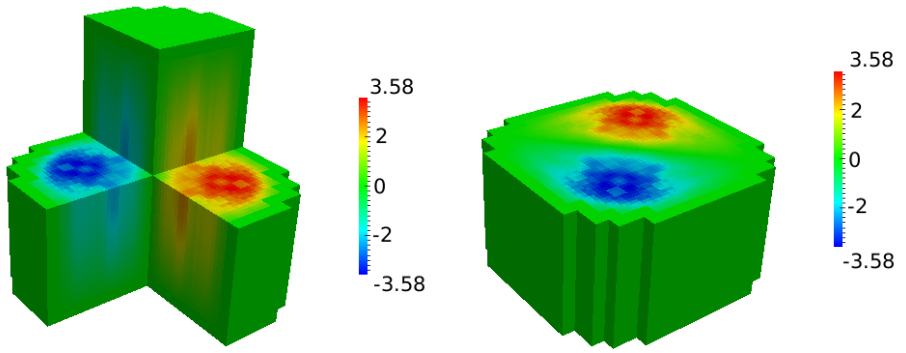
Figure 6.54: Radial power errors (%) of PWR MOX/UO_2 reactor for Mesh 3

Finally, Figures 6.55-6.57 show the power distribution for the five modes and Mesh 3.



First mode Second mode

Figure 6.55: Power (first and second modes) for PWR MOX/UO_2 reactor for Mesh 3



Third mode Fourth mode

Figure 6.56: Power (third and fourth modes) for PWR MOX/UO_2 reactor for Mesh 3

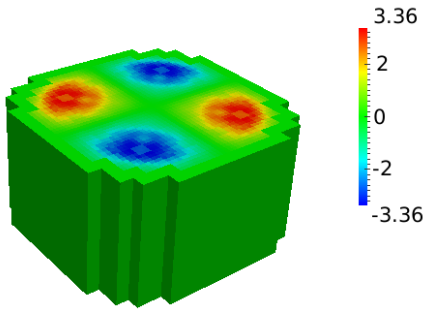


Figure 6.57: Power (fifth mode) for PWR MOX/UO_2 reactor for Mesh 3

6.6 Parallelization

The main objective of this section is to test the parallelization of the method. To do so, the author solves the Steady State Neutron Diffusion Equation applied to a VVER. In particular, this VVER reactor is modeled with two energy groups, without upscattering and with fission neutrons produced in the first energy group.

The author used the improved inter-cells polynomial expansion method of Section 3.2.3, because it is the method giving the most accurate results with the lowest computational time.

As VVERs have hexagonal geometry, the author also performs a sensitivity analysis of different meshes and polynomial sets. In addition, this case also includes a sensitivity analysis of the different linear system solvers and preconditioners of PETSc.

The results of this section were published in Bernal et al. 2018.

6.6.1 VV1K3D reactor

VV1K3D is a VVER mockup and is composed of 1690 hexagonal prisms, distributed in 10 axial levels of 20 cm in length. The hexagonal prisms are regular and their flat-to-flat distance is 23.6 cm. Figure 6.58 displays a cross section of the reactor, where each number represents an assembly type. Assemblies from 1 to 5 are composed of materials from 1 to 5, respectively. Composition of assembly 6 changes with the axial level: in the first five axial levels it is composed of material 4 and in the last ones it is composed of material 3. The cross sections of all the materials are shown in Table 6.37 (Chao and Shatilla 1995).

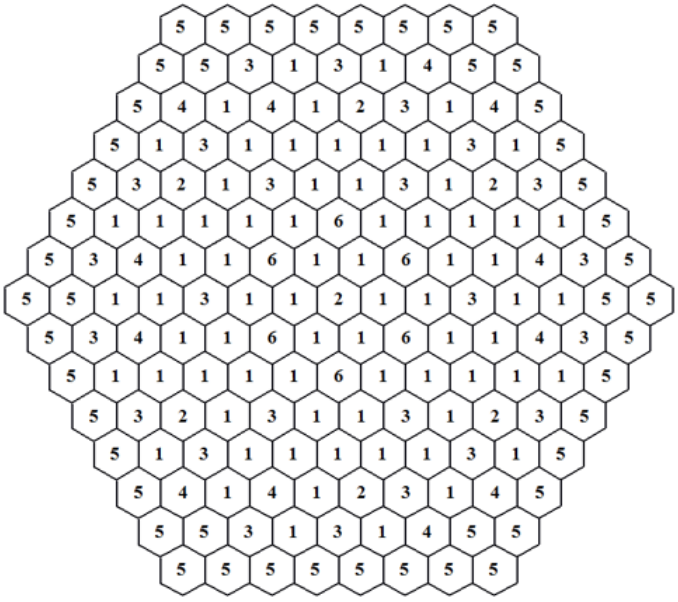


Figure 6.58: Assembly distribution in VV1K3D reactor

The author used three meshes. Mesh 1 is shown in Figure 6.59, which divides each hexagonal prism in 3 hexahedra as displayed in Figure 6.60 (left). Mesh 2 divides each hexahedron of Mesh 1 in 2 x 2 x 2 hexahedra and Mesh 3 divides each hexahedron of Mesh 1 in 3 x 3 x 3 hexahedra as exhibited in Figure 6.60.

As regards the polynomial sets, there are 120 possible 7-combinations of the set composed of 3D monomials of order 2. The author tested these 120 combinations, and only 2 of them gave valid results in this reactor. The first one is: 1, x, y, z, x^2, z^2 and xy. The second one is: 1, x, y, z, y^2, z^2 and xy.

Table 6.37: Cross section data for VV1K3D reactor

Material	Group	$D_g\ (cm)$	$\Sigma_{a,g}\ (cm^{-1})$	$\nu\Sigma_{f,g}\ (cm^{-1})$	$\Sigma_{s,g\to g+1}\ (cm^{-1})$
1	1	1.38320	$8.3859{\cdot}10^{-3}$	$4.81619{\cdot}10^{-3}$	$1.64977{\cdot}10^{-2}$
	2	$3.86277{\cdot}10^{-1}$	$6.73049{\cdot}10^{-2}$	$8.46154{\cdot}10^{-2}$	
2	1	1.38299	$1.15550{\cdot}10^{-2}$	$4.66953{\cdot}10^{-3}$	$1.47315{\cdot}10^{-2}$
	2	$3.89403{\cdot}10^{-1}$	$8.10328{\cdot}10^{-2}$	$8.52264{\cdot}10^{-2}$	
3	1	1.39522	$8.9443{\cdot}10^{-3}$	$6.04889{\cdot}10^{-3}$	$1.56219{\cdot}10^{-2}$
	2	$3.86225{\cdot}10^{-1}$	$8.44801{\cdot}10^{-2}$	$1.19428{\cdot}10^{-1}$	
4	1	1.39446	$1.19932{\cdot}10^{-2}$	$5.91507{\cdot}10^{-3}$	$1.40185{\cdot}10^{-2}$
	2	$3.87723{\cdot}10^{-1}$	$9.89671{\cdot}10^{-2}$	$1.20497{\cdot}10^{-1}$	
5	1	1.39506	$9.1160{\cdot}10^{-3}$	$6.40256{\cdot}10^{-3}$	$1.54981{\cdot}10^{-2}$
	2	$3.84492{\cdot}10^{-1}$	$8.93878{\cdot}10^{-2}$	$1.29281{\cdot}10^{-1}$	

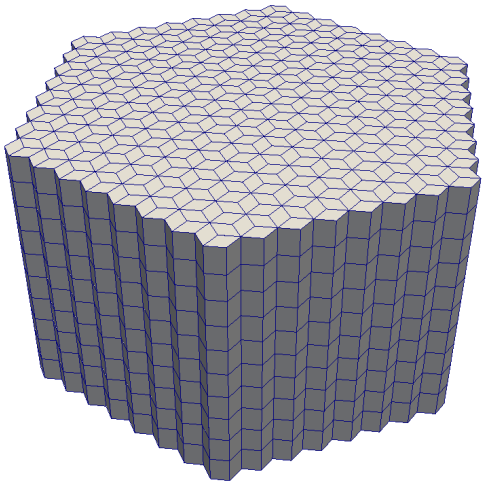


Figure 6.59: Mesh 1 of VV1K3D reactor

Boundary conditions are zero flux for all boundaries.

In the following paragraphs, the author will perform three different sensitivity analyses: polynomial set, mesh and linear system solver. For each analysis, the author fixes the other parameters. As the author already verified the method in the previous sections, in this section the author will not compare the results with those obtained with other reference codes.

First, the author performs the sensitivity analysis of the polynomial sets, Combinations 1 and 2. As regards the other parameters, the author used Mesh 3 and

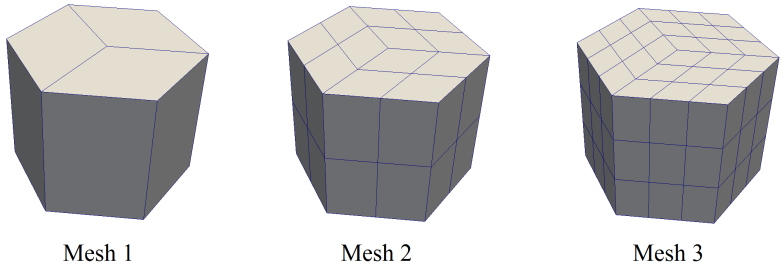


Figure 6.60: Subdivisions of hexagonal prism in Mesh 1, 2 and 3

the linear system solver was GMRES with Additive Schwarz preconditioner. This preconditioner uses Incomplete LU preconditioner as local preconditioner by default.

The computational time for each polynomial combination was: 4 min and 43 s for the first one; 4 min and 54 s for the second one. The results are evaluated considering that Combination 1 is the reference. On the one hand, Table 6.38 exhibits the eigenvalue results, which are accurate because $EE < 100$ pcm. On the other hand, Table 6.39 shows the axial power errors for the different axial levels, which are good since they are lower than 1 %. One can conclude that for fine meshes, the results are almost insensitive to the polynomial sets.

Table 6.38: Eigenvalue results for the sensitivity analysis of the polynomial set, for VV1K3D reactor

Eigenvalue	Combination 1	Combination 2	$EE(pcm)$
1	1.005460	1.005503	4.23
2	0.987339	0.987434	9.62
3	0.987319	0.987403	8.57
4	0.968399	0.968575	18.14
5	0.964224	0.964336	11.64

Table 6.39: Axial power results for the sensitivity analysis of the polynomial set, for VV1K3D reactor

Axial level	10	9	8	7	6	5	4	3	2	1
$PE(\%)$	0.09	0.08	0.08	0.05	0.02	0.04	0.08	0.11	0.16	0.12

Second, the author performs the sensitivity analysis of the mesh, for Meshes 1-3. In this case, one fixes the polynomial set to Combination 1: 1, x, y, z,

x^2, z^2 and xy. The linear system solver is also fixed to GMRES with Additive Schwarz preconditioner.

For each mesh, the number of rows of matrices $L_{g,g}$ is: 19323 for Mesh 1; 158412 for Mesh 2; 538947 for Mesh 3. In addition, the computational time for each mesh was: 5 s for Mesh 1; 58 s for Mesh 2; 4 min and 43 s for Mesh 3. The results are evaluated considering that Mesh 3 is the reference. Table 6.40 exhibits the eigenvalue results, whereas Table 6.41 shows the axial power results. One draws the following conclusion. Mesh 2 is more accurate than Mesh 1, but the results of Mesh 1 are good enough, because the maximum eigenvalue error is about 100 pcm and the maximum axial power error is about 1 %.

Table 6.40: Eigenvalue results for the sensitivity analysis of the mesh, for VV1K3D reactor

	Eigenvalue			$EE(pcm)$	
Eigenvalue	Mesh 1	Mesh 2	Mesh 3	Mesh 1	Mesh 2
1	1.005193	1.005389	1.005460	26.61	7.12
2	0.987292	0.987313	0.987339	4.78	2.64
3	0.986931	0.987234	0.987319	39.22	8.59
4	0.968356	0.968382	0.968399	4.46	1.78
5	0.962930	0.964002	0.964224	134.26	22.99

Table 6.41: Axial power results for the sensitivity analysis of the mesh, for VV1K3D reactor

	Axial level	10	9	8	7	6	5	4	3	2	1
PE	Mesh 1	0.97	0.88	0.72	0.44	0.02	0.44	0.72	0.93	1.08	1.10
(%)	Mesh 2	0.21	0.19	0.15	0.10	0.01	0.09	0.16	0.20	0.24	0.24

Third, the author performs the sensitivity analysis of the linear system solver. In this case, the mesh and the polynomial set are fixed: Mesh 3 and Combination 1 (1, x, y, z, x^2, z^2 and xy). The author tested the following linear system solvers of PETSc: BiConjugate Gradient (bicg), GMRES, Generalized Conjugate Residual (gcr), BiCGSTAB (bcgs) and Conjugate Gradient Squared (cgs). The author used these solvers because they can be applied to non-symmetric matrices. These solvers were used with the following preconditioners of PETSc: Jacobi, SOR and Additive Schwarz (asm), which is the same as Incomplete LU for one processor. The author used the default tolerances of PETSc.

Among all these combinations of solvers and preconditioners, only one of them is forbidden in PETSc (for non-symmetric matrices): BiConjugate Gradient

with SOR. For the rest, the results are the same, but there are differences in the computational time as shown in Figure 6.61. From this figure, one concludes that GMRES is the fastest method in combination with the Additive Schwarz preconditioner.

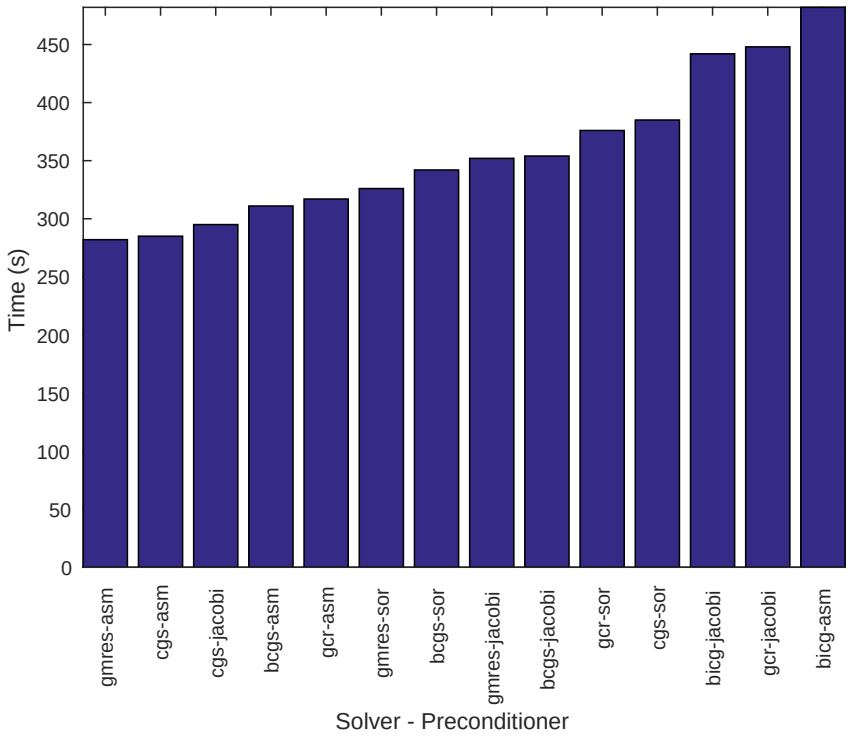


Figure 6.61: Time results for the linear system solvers, for VV1K3D reactor

Furthermore, the author calculated the condition number of matrices $L_{1,1}$ and $L_{2,2}$ by using the singular value decomposition solver of SLEPc. Table 6.42 shows the calculated condition number for each mesh and polynomial combinations. In this table, C.1 corresponds to Combination 1 and C.2 corresponds to Combination 2. One draws two conclusions from this table. First, the matrices are well-conditioned, since their condition number is very low. Second, the condition number of this discretization, applied to this reactor, is almost insensitive to the mesh and the polynomial terms.

Table 6.42: Condition number of $L_{1,1}$ and $L_{2,2}$, for VV1K3D reactor

$L_{1,1}$						$L_{2,2}$					
Mesh 1		Mesh 2		Mesh 3		Mesh 1		Mesh 2		Mesh 3	
C.1	C.2	C.1	C.2	C.1	C.2	C.1	C.2	C.1	C.2	C.1	C.2
56.1	56.1	56.3	56.3	62.9	56.3	74.4	74.6	74.6	74.6	74.6	74.6

For testing the parallel computation, the author ran the simulations with Mesh 3, Combination 1, GMRES solver and Additive Schwarz preconditioner. The size of the matrices $L_{1,1}$ and $L_{2,2}$ for this case is 538947. To evaluate the parallelization, one uses the speedup, as discussed in Section 6.1. Figure 6.62 displays the speedup of the parallelization. This figure exhibits the total simulation time. One can draw two conclusions from this figure. First, the performance is close to ideal up to 5 processors. Second, a reasonably good performance gain is seen up to 14 processors in the strong scaling sense. It should be pointed out that the efficacy of the Additive Schwarz preconditioner decreases with the number of processors, so this behavior is normal. Finally, the parallel computation runs the case of Mesh 3 in 35 s, with 16 processors.

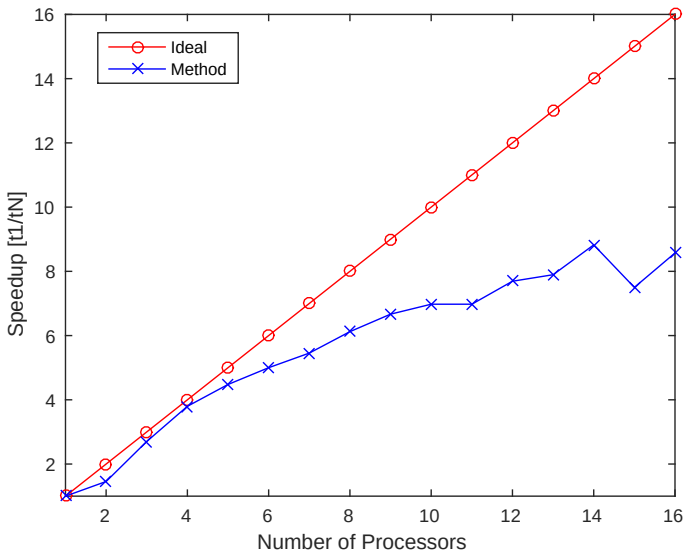


Figure 6.62: Speedup of the parallelization.

Finally, Figures 6.63-6.65 show the power distribution for the five modes, Mesh 3 and Combination 1.

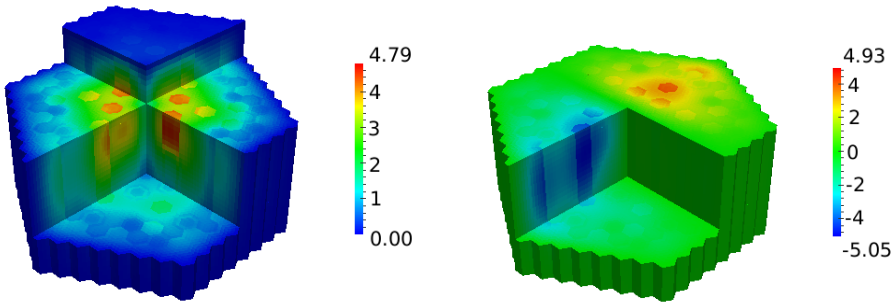


Figure 6.63: Power (first and second modes) for VV1K3D reactor, for Mesh 3 with Combination 1

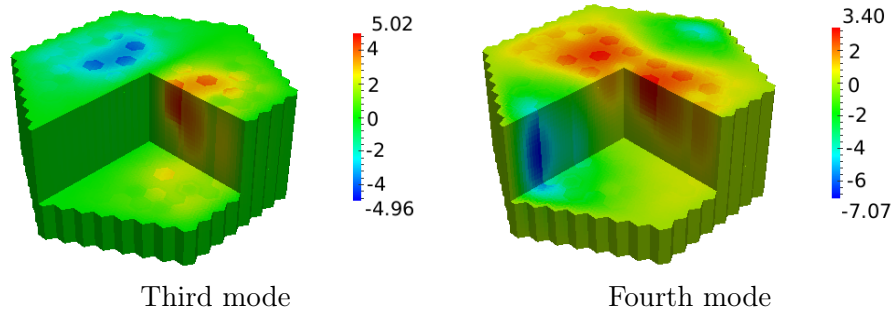


Figure 6.64: Power (third and fourth modes) for VV1K3D reactor, for Mesh 3 with Combination 1

6.7 Adjoint calculation

In this section, the author evaluates the methods for calculating the adjoint eigenvectors, which are proposed in Section 4.2. These methods are the one developed by the author and the other is that proposed by Döring and Kalkkuhl (Döring, Kalkkuhl, and Schröder 1993).

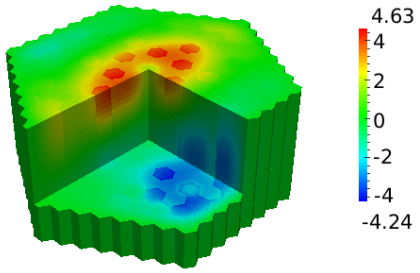


Figure 6.65: Power (fifth mode) for VV1K3D reactor, for Mesh 3 with Combination 1

The evaluation is based on the accomplishment of two equations. First, the biorthogonal property, which was shown in Equation 4.52. The author calculated 5 adjoint eigenpairs and checked the biorthogonal property with 10 forward eigenvectors. Second, the adjoint eigenvalue problem, which is evaluated with the error shown in Equation 6.7.

$$E_i = \left\langle L^T \Phi_i^* \mathbf{k}_i - M^T \Phi_i^*, L^T \Phi_i^* \mathbf{k}_i - M^T \Phi_i^* \right\rangle \tag{6.7}$$

The author tested the methods in two reactors: the 3D homogeneous reactor and Langenbuch reactor. These reactors are modeled with two energy groups, without upscattering and with fission neutrons produced in the first energy group.

6.7.1 3D homogeneous reactor

This reactor is the same as that of Section 6.2.3, so one can find the geometry and cross sections in the mentioned section.

The method developed by the author and that developed by Döring and Kalkkuhl provide the same results. Errors (E_i) are zero for the five eigenpairs. The biorthogonal property is also accomplished for the 10 forward eigenvectors, as shown in Table 6.43. One can conclude that the results are excellent.

Table 6.43: Biorthogonal property for the 3D homogeneous reactor

	$L\Phi_j$									
Φ_i^*	-995.20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	1003.58	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	-1017.22	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	1021.44	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	-1029.48	0.0	0.0	0.0	0.0	0.0

6.7.2 Langenbuch reactor

This reactor is the same as the one of Section 6.2.4. Thus, one can find in this section the geometry, materials and cross sections.

As regards the results obtained with the method of Döring and Kalkkuhl, Table 6.44 displays the errors (E_i) and Table 6.45 shows the biorthogonal property. One might draw two conclusions from these tables. First, E_i are not close to zero. Second, there are some values of the biorthogonal property which are clearly not zero, such as $i = 2, j = 6$ and $i = 5, j = 9$; thus, one cannot conclude that the biorthogonal property is accomplished.

Table 6.44: Errors for Langenbuch reactor and the method of Döring and Kalkkuhl

Eigenpair	1	2	3	4	5
E_i	320.81	305.05	323.80	256.74	274.24

Table 6.45: Biorthogonal property for Langenbuch reactor and the method of Döring and Kalkkuhl

	$L\Phi_j$									
Φ_i^*	-92.78	0.0	0.0	0.0	-0.31	0.0	-6.29	0.0	-0.36	0.0
	0.0	-86.17	0.0	0.0	0.0	-19.44	0.0	-0.39	-0.33	0.0
	0.0	0.0	97.52	0.0	0.0	0.0	-0.41	0.0	0.0	-0.16
	0.0	0.0	0.0	81.94	0.0	0.74	0.0	-17.69	0.33	0.0
	0.0	0.0	0.0	0.0	-88.25	0.07	0.0	0.20	-19.92	0.0

One can see the results for the method developed by the author in Tables 6.46 and 6.47. Table 6.46 displays the errors (E_i), which are close to zero. Table

6.47 shows the biorthogonal property, in which one can appreciate that there are some values slightly higher than zero. However, these values are almost zero and they are about three order of magnitude lower than non-zero values. Thus, one concludes that the biorthogonal property is accomplished. In conclusion, the method proposed by Döring and Kalkkuhl might provide wrong results for heterogeneous results, whereas the method of this thesis provides accurate results.

Table 6.46: Errors for Langenbuch reactor and the method developed by the author

Eigenpair	1	2	3	4	5
E_i	0.69	0.70	1.37	1.13	0.72

Table 6.47: Biorthogonal property for Langenbuch reactor and the method developed by the author

	$L\Phi_j$									
Φ_i^*	101.08	0.0	0.0	0.0	0.0	-0.00	0.0	-0.01	0.00	0.0
	0.0	102.88	0.0	0.0	0.0	-0.28	0.0	-0.00	0.00	0.0
	0.0	0.0	106.33	0.0	0.0	0.0	0.00	0.0	0.0	0.00
	0.0	0.0	0.0	99.32	0.0	-0.01	0.0	0.19	-0.01	0.0
	0.0	0.0	0.0	0.0	105.76	0.01	0.0	-0.00	-0.31	0.0

6.8 Modal method

In this section the author assesses the modal method for calculating the time-dependent Neutron Diffusion Equation for the multigroup formulation. For the spatial discretization, the author used the improved inter-cells polynomial expansion method of Section 3.2.3.

The modal method is tested in a simple transient of PWR MOX/UO2 reactor used in Section 6.5.2.

6.8.1 PWR MOX/UO2 reactor

This transient case is a withdrawal and insertion of a control rod of the reactor proposed in the PWR MOX/UO2 Core Transient Benchmark (Kozlowski and Downar 2007). The author used and defined this reactor in Section 6.5.2, particularly in Figure 6.51, which shows a quarter of this reactor, although the author simulated the whole geometry. As regards the cross sections, this benchmark gives the cross sections for different values of the thermal-hydraulic variables, different energy groups (2, 4 and 8) and for 6 precursors groups. For this case, the author used the 8-energy groups cross sections, which include upscattering terms from the fifth energy group and fission production in the first four energy groups.

In this section, the author did not use the thermal-hydraulic coupling; but he used the following fixed thermal-hydraulic variables: moderator density of 752.06 kg/m^3, fuel temperature of 560.0 K and boron concentration of 1000 ppm. With respect to the control rods, the author used the same position as defined in the benchmark.

In the initial state of this transient, all the control rods are inserted and then the control rod positioned in the middle of the reactor is fully withdrawn in 3 s. Then, this control rod is another time fully inserted in 3 s. Thus, the author performed a total transient of 6 s using a step time of 0.025 s.

The author modeled the reactor with two meshes composed of hexahedra. Figure 6.66 shows Mesh 1. Mesh 2 is obtained by subdividing each hexahedra of Mesh 1 in 2 x 2 x 2 identical hexahedra.

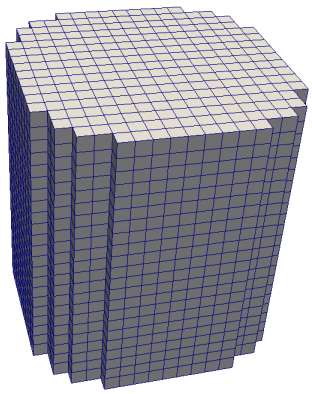


Figure 6.66: Mesh 1 for the transient of PWR MOX/UO_2 reactor

With respect to the polynomial expansion, the author stated in Section 6.5.2 that only one of polynomial set of order two gave valid results: 1, x, y, z, x^2, y^2, z^2.

The results are compared with those obtained by PARCS (Downar et al. 2006) using the NEMMG solver. The author modeled the reactor in PARCS with the same mesh as Mesh 1.

In this section, the author only shows three kind of results due to the extent of them. The first one is the total power at each time step. The second and third ones are the radial and axial power profile at the time step in which the power reaches its maximum value. The author also performed a sensitivity analysis of three parameters: the mesh, the time step for updating the eigenvectors or modes and the number of modes. For the updating of modes, the author used a tolerance of 10^{-6} for the convergence of the eigenvalue calculations.

Figures 6.67-6.69 display the total power results at each time step. In these figures, VALKIN-FVM is the name of the modal method developed in this thesis. Figure 6.67 shows the influence of the time step for updating the modes, for Mesh 2 and 5 modes. Figure 6.68 shows the sensitivity analysis of the mesh, for the calculation performed with 5 modes and updating these with a time step of 0.1 s. In Figure 6.69, one can see the results for different number of modes used in the modal method, for Mesh 2 and updating the modes with a time step of 0.1 s. One can draw three conclusions from these figures. First, one should update the modes with a time step of 0.1 s to obtain accurate results. Second, the calculation using Mesh 2 and updating the modes each 0.1 s produced excellent results. Third, the effect of the number of modes on the modal method for this case is not remarkable.

As regards the time results, Table 6.48 shows the computational time for PARCS and VALKIN-FVM.

Table 6.48: Time results(h:min:s) for the transient of PWR MOX/UO_2 reactor

	VALKIN-FVM		
PARCS	1 MODE	3 MODES	5 MODES
1:20:35	2:22:50	3:0:47	4:11:54

Moreover, the power reaches its maximum value at 3.45 s. At this time step and for the calculation with Mesh 2, 5 modes and 0.1 s of updating time step, the power results are shown in Table 6.49 and Figure 6.70. Table 6.49 shows the axial power errors, where one can see that the maximum value is 1.91%.

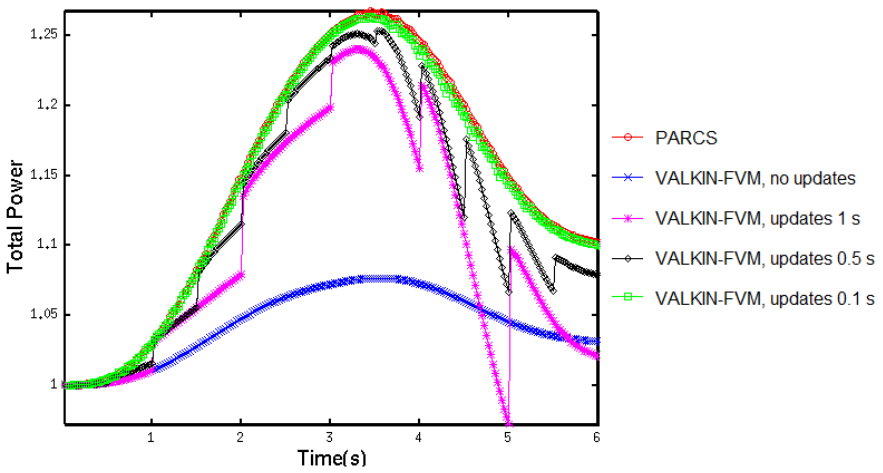


Figure 6.67: Analysis of the time step for updating the modes for the transient of PWR MOX/UO_2 reactor, with Mesh 2

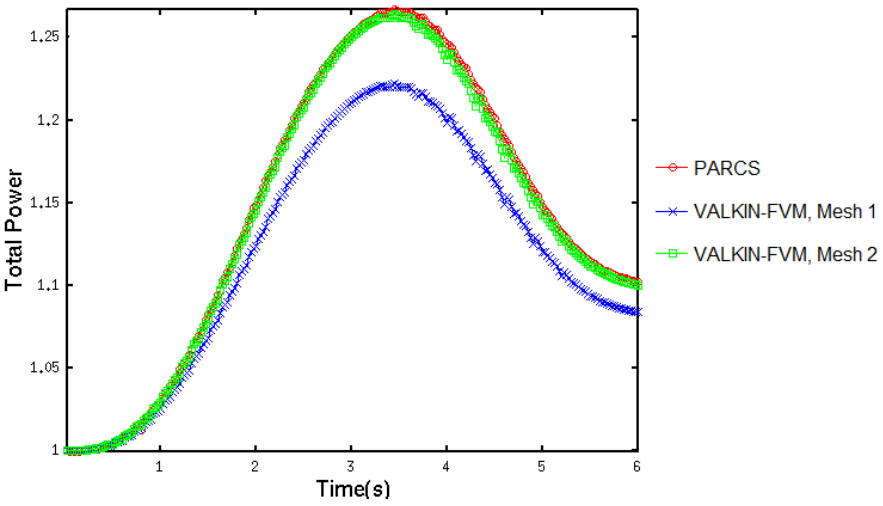


Figure 6.68: Analysis of the mesh for the transient of PWR MOX/UO_2 reactor

Figure 6.70 displays the radial power errors, whose maximum value is 5.44%. One can conclude that the results are good.

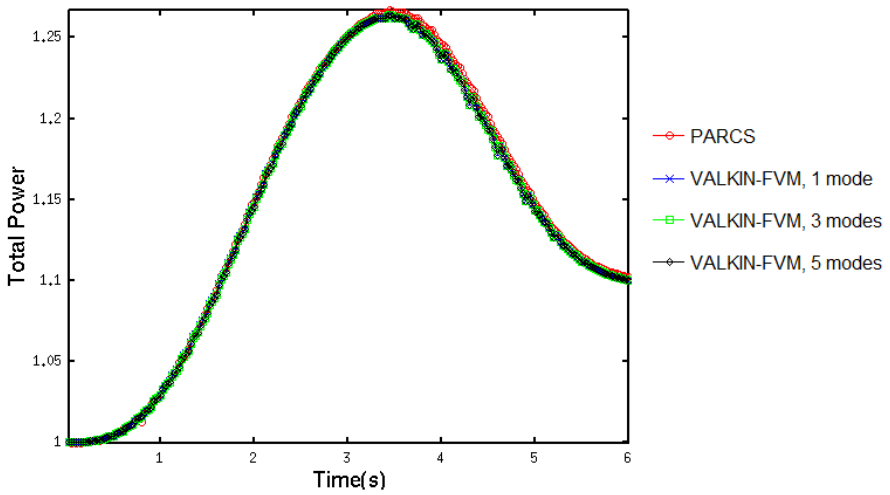


Figure 6.69: Analysis of the number of modes for the transient of PWR MOX/UO_2 reactor, with Mesh 2

Finally, the author ran this case with parallel computing with VALKIN-FVM. Figure 6.71 shows the parallel performance of VALKIN-FVM for several number of modes and number of processors. From this figure, one concludes that VALKIN-FVM has a good parallel computing performance, particularly up to 8 processors. From this parallel performance, one realizes that VALKIN-FVM would have similar computational time to that of PARCS with just 2 processors, but it could be faster with more processors. The author would like to highlight that the parallel performance increases with the number of modes calculated.

6.9 Neutron Transport Equation with the Discrete Ordinates and FVM

In this section, the author tests the methods of Chapter 5, for solving the Steady State of the Neutron Transport Equation, with the Discrete Ordinates formulation and the FVM.

In particular, the author applied these methods to two reactors: a 3D homogeneous reactor and the 2D reactor of the C5G7 MOX Benchmark (Lewis et al.

Table 6.49: Axial power errors (%) for the transient of PWR MOX/UO_2 reactor

Axial level	$PE(\%)$
21	0.69
20	0.73
19	0.40
18	0.37
17	0.24
16	0.22
15	0.19
14	0.12
13	0.10
12	0.05
11	0.02
10	0.09
9	0.08
8	0.12
7	0.23
6	0.23
5	0.26
4	0.34
3	0.18
2	1.91

2001). For both cases, the author used GMRES with the Additive Schwarz preconditioner for solving the linear systems.

6.9.1 3D homogeneous reactor

This reactor has the same geometry as that of Section 6.2.3, but the cross sections are different. The cross sections are defined for the two-energy group formulation, without upscatering and with fission neutrons produced in the first energy group. Table 6.50 shows these cross sections.

Table 6.50: Cross sections of the homogeneous reactor for the Neutron Transport Equation

Group	$\Sigma_{t,g}\ (cm^{-1})$	$\nu\Sigma_{f,g}\ (cm^{-1})$	$\Sigma_{s,g\to g}\ (cm^{-1})$	$\Sigma_{s,g\to g+1}\ (cm^{-1})$
1	$5.2096647{\cdot}10^{-1}$	$7.72686955{\cdot}10^{-3}$	$4.95171815{\cdot}10^{-1}$	$1.60585809{\cdot}10^{-2}$
2	1.31245720	$1.55083969{\cdot}10^{-1}$	1.20309806	

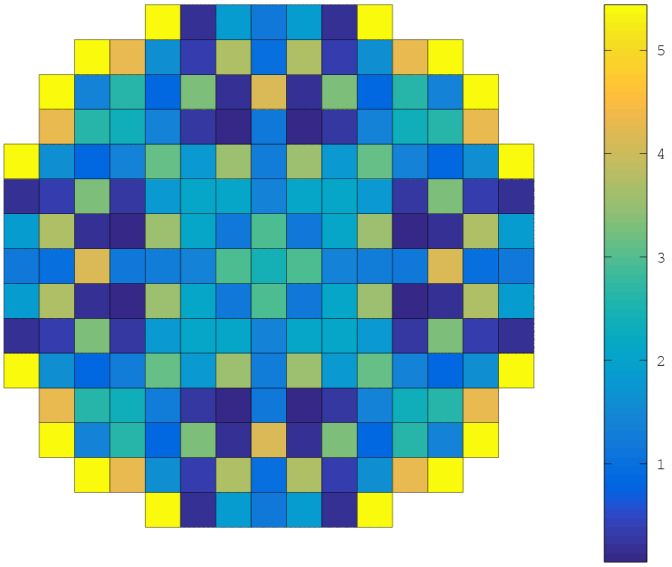


Figure 6.70: Radial power errors (%) for the transient of PWR MOX/UO_2 reactor

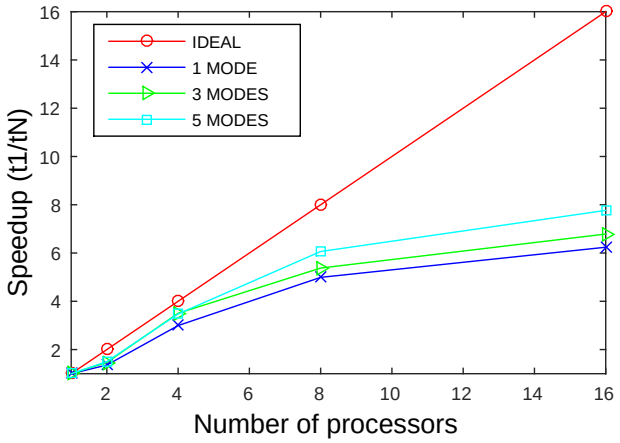


Figure 6.71: Speedup for the transient of PWR MOX/UO_2 reactor

The author used six meshes. Mesh 1 is composed of 10 x 6 x 18 identical hexahedra (1080 hexahedra). Mesh 2 is obtained by subdividing each hexahedron of

Mesh 1 in 2 x 2 x 2 identical hexahedra (8640 hexahedra). Mesh 3 is obtained by subdividing each hexahedron of Mesh 1 in 3 x 3 x 3 identical hexahedra (29160 hexahedra). Mesh 4 is obtained by subdividing each hexahedron of Mesh 1 in 4 x 4 x 4 identical hexahedra (69120 hexahedra). Mesh 5 is obtained by subdividing each hexahedron of Mesh 1 in 5 x 5 x 5 identical hexahedra (135000 hexahedra). Mesh 6 is obtained by subdividing each hexahedron of Mesh 1 in 24 tetrahedra (25920 tetrahedra). One can see Mesh 6 in Figure 6.72.

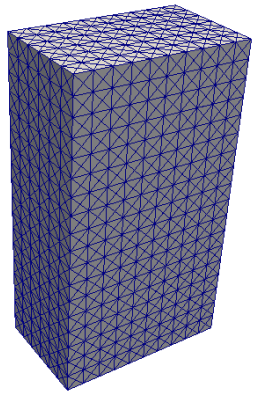


Figure 6.72: Mesh 6 for the 3D homogeneous reactor

Moreover, the author will perform a sensitivity analysis of the coefficient δ of the linear-step method, which appears in Equation 5.45. Particularly, the author used the following values: 0.0, 0.5 and 1.0. If $\delta = 0.0$, the linear-step method is equivalent to the linear method. If $\delta = 1.0$, the linear-step method is equivalent to the step method. The author will determine the best value of this coefficient δ.

With respect to the directional discretization, the author modeled this case with the level-symmetric quadrature, particularly with S_2. This case is equivalent to the product quadrature with 2 polar angles and 4 azimuthal angles, that is, $N_p = 2$ and $N_a = 1$ of Section 5.2.

The author calculated the reference results with TITAN (Yi 2009). The author used Mesh 4 to model this reactor in TITAN, but the results are evaluated in Mesh 1. In addition, the author ran TITAN with the Discrete Ordinates method and the Diamond Difference method for the spatial discretization. For the Discrete Ordinates, the author used the level-symmetric quadrature,

particularly S_2. For these conditions, the eigenvalue calculated with TITAN is 1.07352 and the computational time is 512 s.

The boundary conditions are vacuum in all boundaries. The author calculated five modes in each simulation, but only calculates one eigenvalue error, because TITAN only can calculate one eigenvalue. The author ran all the simulations, including the one performed with TITAN, on an Intel Core i7-3770 CPU (3.4 GHz), with the CentOS 6.8 operating system.

First, the author analyzes the effect of the coefficient δ for Mesh 1. Table 6.51 shows the computation time, eigenvalues and EE_1 for different values of δ. From this table, one draws three conclusions. First, the higher the value of δ, the higher the eigenvalue error. This might be due to the fact that the leakage terms increase with δ, and consequently the value of the eigenvalue decreases. Second, the lower the value of δ, the higher the computational time. Third, for $\delta = 0.0$, there are several eigenvalues with similar values. It seems that for values of δ close 0.0, the eigenvalues are closer, and therefore the rate of convergence is worse. As regards the results of $\delta = 0.018$, the author will comment on them afterwards.

Table 6.51: Results for the 3D homogeneous reactor with the linear-step method and Mesh 1: Computational time (s), eigenvalues and eigenvalue error of the first mode (pcm)

δ	Time	$\mathbf{k}_1$	$\mathbf{k}_2$	$\mathbf{k}_3$	$\mathbf{k}_4$	$\mathbf{k}_5$	EE_1
0.0	3	1.079883	1.078980	1.074792	1.073899	1.059962	592.71
0.5	1	0.914669	0.862032	0.786708	0.772808	0.733606	14797.25
1.0	1	0.831953	0.762853	0.667857	0.657651	0.611328	22502.36
0.018	2	1.071079	1.047655	1.013257	1.006292	0.985595	227.37

Besides the eigenvalues, the author plots the distribution of the power for the first two modes and $\delta = 0.0$, which is shown in Figure 6.73. In addition, Figure 6.74 displays the same distribution, but for $\delta = 1.0$. From these figures, one realizes that the power distribution of the second mode, for $\delta = 0.0$, does not make sense, although the power distribution of the first mode seems to be right. All in all, one concludes that one has to use the lowest value of δ which provides a power distribution of the modes with physical sense. The author tested a number of values of δ, and concluded that the most accurate δ for this case is 0.018, which is proved in Table 6.51 that it has the lowest eigenvalue error.

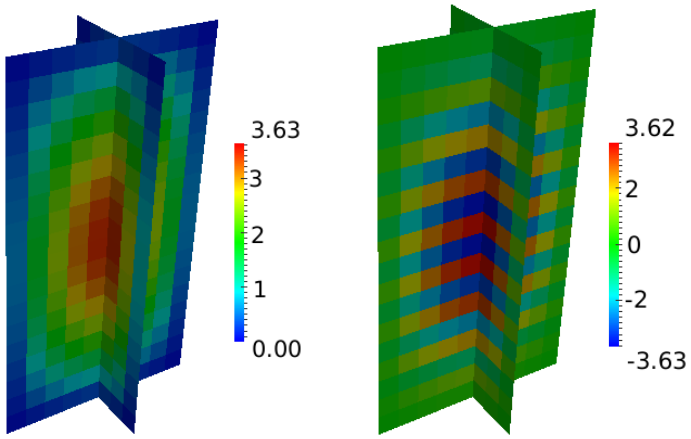


Figure 6.73: Power (first and second modes) for the 3D homogeneous reactor with the linear-step method, Mesh 1 and $\delta = 0.0$

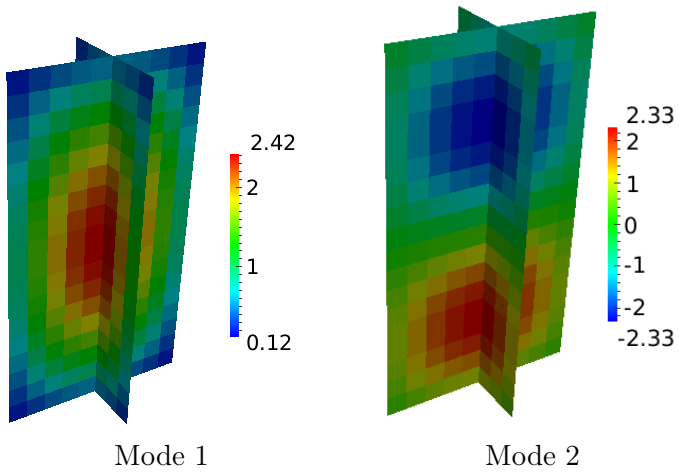


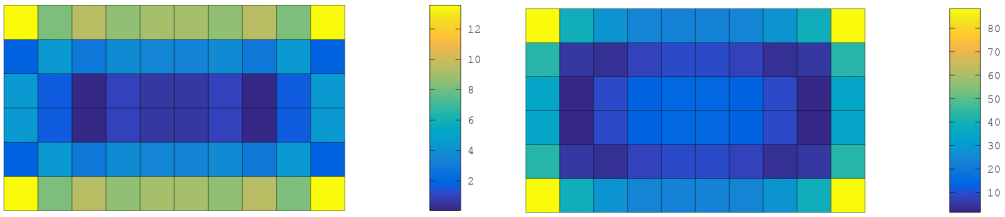
Figure 6.74: Power (first and second modes) for the 3D homogeneous reactor with the linear-step method, Mesh 1 and $\delta = 1.0$

With respect to the power results, Table 6.52 and Figures 6.75 and 6.76 show these results for the mentioned values of δ. Table 6.52 contains the axial power errors, whereas Figures 6.75 and 6.76 display the radial power errors. Finally, one can conclude that one has to use values of δ close to 0.0, but not 0.0, to obtain the most accurate results.

Table 6.52: Axial power errors (%) for the 3D homogeneous reactor with the linear-step method and Mesh 1

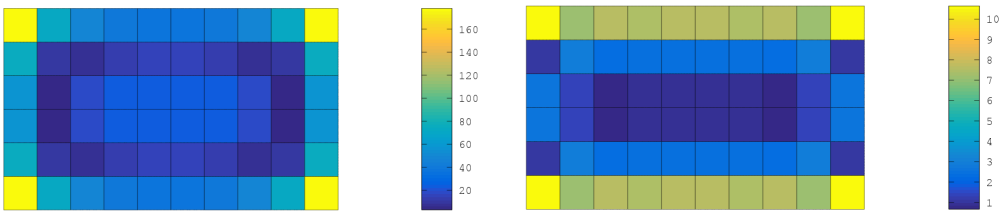
	$PE(\%)$			
Axial level	$\delta = 0.0$	$\delta = 0.5$	$\delta = 1.0$	$\delta = 0.018$
18	3.36	57.03	108.03	1.43
17	0.01	14.14	27.75	0.06
16	0.53	5.15	10.15	0.19
15	0.00	1.16	2.40	0.14
14	0.18	1.01	1.87	0.08
13	0.15	2.32	4.45	0.00
12	0.08	3.11	6.04	0.07
11	0.28	3.57	6.98	0.14
10	0.23	3.79	7.41	0.17
9	0.28	3.79	7.41	0.17
8	0.23	3.58	6.98	0.14
7	0.13	3.12	6.05	0.08
6	0.11	2.33	4.46	0.01
5	0.13	1.03	1.89	0.08
4	0.04	1.14	2.38	0.13
3	0.47	5.12	10.14	0.18
2	0.09	14.11	27.73	0.06
1	3.43	56.97	107.97	1.43

Now, the author performs a sensitivity analysis of the mesh. For each mesh, the author determines the best value of δ, which is written in Table 6.53. This table also includes the computational time, eigenvalues and eigenvalue errors. From this table, one draws three conclusions. First, the finer the mesh, the more accurate is the first eigenvalue. Second, one obtains lower values of δ with physical sense for finer meshes. Third, meshes composed of tetrahedra, such as Mesh 6, can use $\delta = 0.0$, although the eigenvalue results are less accurate than those obtained with meshes composed of hexahedra. On the other hand, one can see the power results for each mesh in Table 6.54 and Figures 6.77-6.79. Table 6.54 exhibits the axial power errors, while Figures



$\delta = 0.0$ $\delta = 0.5$

Figure 6.75: Radial power errors (%) for the 3D homogeneous reactor with the linear-step method and Mesh 1 (I)



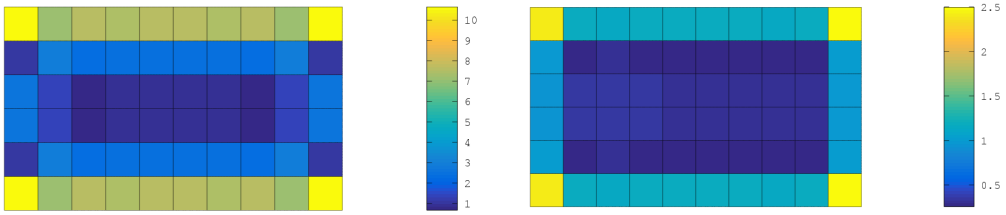
$\delta = 1.0$ $\delta = 0.018$

Figure 6.76: Radial power errors (%) for the 3D homogeneous reactor with the linear-step method and Mesh 1 (II)

6.77-6.79 display the radial power errors. One draws similar conclusions to those of the eigenvalues. First, the axial power errors are similar for all meshes, but one can see an improvement of the radial power errors for fine meshes. Second, meshes composed of hexahedra provides better results than the mesh composed of tetrahedra. Third, the finer the mesh, the more accurate the radial power results, though this improvement is slight.

Table 6.53: Results for the 3D homogeneous reactor with the linear-step method: Computational time (s), eigenvalues and eigenvalue error of the first mode (pcm)

Mesh	δ	Time	$\mathbf{k}_1$	$\mathbf{k}_2$	$\mathbf{k}_3$	$\mathbf{k}_4$	$\mathbf{k}_5$	EE_1
1	0.018	2	1.071079	1.047655	1.013257	1.006292	0.985595	227.37
2	0.010	17	1.072928	1.050082	1.014626	1.004789	0.984770	55.16
3	0.007	39	1.073339	1.050624	1.014994	1.004635	0.984759	16.84
4	0.005	100	1.073551	1.050896	1.015224	1.004686	0.984872	2.86
5	0.005	187	1.073530	1.050880	1.015155	1.004523	0.984720	0.89
6	0.0	91	1.068146	1.045405	1.009215	0.996339	0.976958	500.64



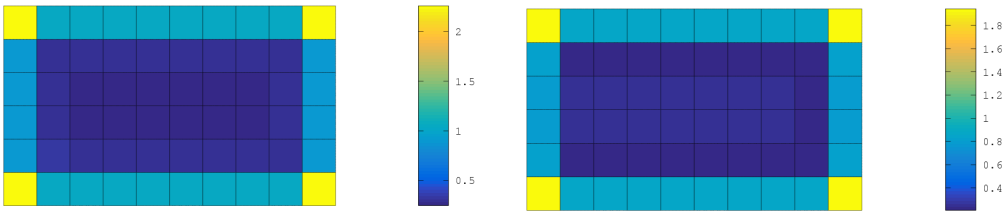
Mesh 1 Mesh 2

Figure 6.77: Radial power errors (%) for the 3D homogeneous reactor with the linear-step method, for Meshes 1 and 2

6.9.2 2D C5G7 reactor

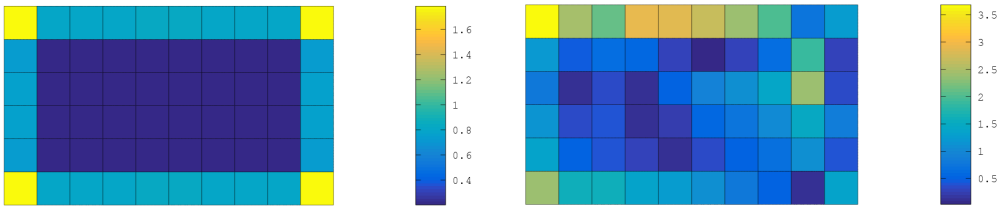
This reactor is the same as the one described in Section 6.5.1, so one can find in the mentioned section the geometry and material composition. The interested reader can look for the cross sections in the C5G7 MOX Benchmark (Lewis et al. 2001).

However, in this case, the author modeled this reactor with two meshes. Mesh 1 is the one described in Section 6.5.1. Mesh 2 is similar to Mesh 1, but the



Mesh 3 Mesh 4

Figure 6.78: Radial power errors (%) for the 3D homogeneous reactor with the linear-step method, for Meshes 3 and 4



Mesh 5 Mesh 6

Figure 6.79: Radial power errors (%) for the 3D homogeneous reactor with the linear-step method, for Meshes 5 and 6

Table 6.54: Axial power errors (%) for the 3D homogeneous reactor with the linear-step method

	$PE(\%)$					
Axial level	Mesh 1	Mesh 2	Mesh 3	Mesh 4	Mesh 5	Mesh 6
18	1.43	1.56	1.50	1.43	1.37	3.67
17	0.06	0.32	0.37	0.37	0.37	0.96
16	0.19	0.26	0.29	0.29	0.29	0.46
15	0.14	0.18	0.21	0.20	0.20	0.14
14	0.08	0.08	0.10	0.08	0.09	0.11
13	0.00	0.03	0.01	0.03	0.02	0.43
12	0.07	0.12	0.11	0.12	0.12	0.78
11	0.14	0.18	0.18	0.19	0.19	0.78
10	0.17	0.22	0.23	0.23	0.23	0.54
9	0.17	0.22	0.23	0.23	0.23	0.41
8	0.14	0.18	0.20	0.19	0.19	0.32
7	0.08	0.12	0.13	0.12	0.12	0.19
6	0.01	0.03	0.04	0.03	0.03	0.30
5	0.08	0.08	0.07	0.08	0.08	0.04
4	0.13	0.18	0.16	0.19	0.19	0.77
3	0.18	0.25	0.25	0.28	0.28	1.67
2	0.06	0.32	0.32	0.34	0.34	3.19
1	1.43	1.56	1.43	1.37	1.31	5.73

reflector is modeled with a progressive mesh as that of Figure 6.80. Likewise the previous section, the author determined the best value of δ for each mesh.

Regarding the quadrature sets, the author used the level-symmetric and the product quadrature developed in this thesis. Table 6.55 summarizes the quadrature sets that the author used. In this table, S_n is the approach used in the level-symmetric quadrature, N_p is the number of collocation points for the polar angle, N_a is the number of collocation points in each quadrant of the azimuthal angle, N_d is the total number of directions and N_d^{2D} is the number of directions simulated for 2D cases.

As mentioned in Section 6.5.1, the reference results for this case are given in the benchmark (Lewis et al. 2001), which were obtained with MCNP code (Briesmeister Judith 2000). Although there are only reference results for the first mode, the author calculated five modes, but for the quarter of the core.

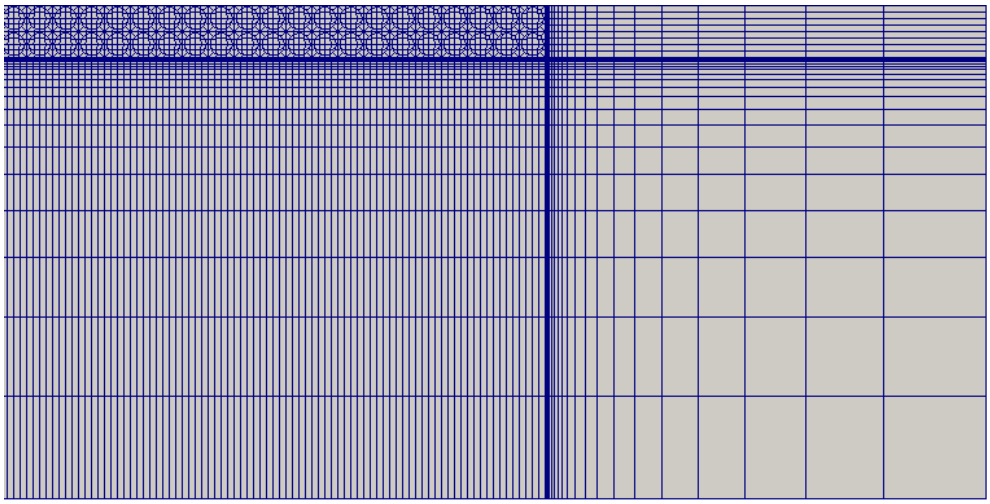


Figure 6.80: Mesh 2 for 2D C5G7 reactor

Table 6.55: Quadrature sets for 2D C5G7 reactor

Set	Type	S_n	$N_p/2$	N_a	N_d	$N_d^{2D} = N_d/2$
S_2	Level-symmetric	S_2			8	4
S_4	Level-symmetric	S_4			24	12
S_6	Level-symmetric	S_6			48	24
$PQ_{1,2}$	Product quadrature		1	2	16	8
$PQ_{1,3}$	Product quadrature		1	3	24	12
$PQ_{1,4}$	Product quadrature		1	4	32	16
$PQ_{2,2}$	Product quadrature		2	2	32	16
$PQ_{2,3}$	Product quadrature		2	3	48	24

As regards the linear system solver, the author used GMRES with Additive Schwarz preconditioner.

Regarding the results, Table 6.56 summarizes the eigenvalue and power results, for each mesh and quadrature set. One draws seven conclusions from this table. First, one obtains different minimum values of δ for each mesh. Particularly, one can use a lower value of δ with Mesh 1. Second, the values of δ used in this reactor are much higher than the values used in the previous section, in the 3D homogeneous reactor. The author proved in the previous section that the closer the value of δ to 0.0, the more accurate the results. However, if one uses values of δ lower than 0.24 or 0.26 in this reactor, the calculation does not converge. Third, one obtains more accurate results with lower values of δ, for any mesh and quadrature set. Fourth, Mesh 2 highly improves the power results with respect to Mesh 1, but increases slightly the eigenvalue errors, for any value of δ and quadrature set. Fifth, the higher the number of directions, the more accurate the results, for any δ and mesh. Sixth, the product quadrature highly improves the eigenvalue errors with respect to the level-symmetric quadrature, for the same number of directions. Seventh, the power results obtained with the level-symmetric quadrature seems to be slightly more accurate than those obtained with the product quadrature. For example, for Mesh 2 and $\delta = 0.26$, one obtains $EE_1 = 383.28 pcm$ and $MPE = 0.53\%$ for S_6, whereas one obtains $EE_1 = 254.97\, pcm$ and $MPE = 0.72\,\%$ for $PQ_{2,3}$.

In addition, Figures 6.81-6.96 show further details of the power errors. One draws the same conclusions as in the previous paragraph. Furthermore, one realizes that the maximum errors are located in those pins composed of MOX which are close to the reflector.

With respect to the computational time, the author gives only the computational time for one case calculating only one mode. This case is Mesh 1, $\delta = 0.24$ and $PQ_{2,2}$. For this case, one processor spent 41 min and 43 s in the calculation. In addition, the author ran this case with parallel computing, obtaining the speedups of Figure 6.97. This figure displays the total simulation time. One draws three conclusions from this figure. First, the performance is close to ideal up to 9 processors. Second, a reasonably good performance gain is seen up to 16 processors in the strong scaling sense. Third, it seems that the speedup is better than the ideal situation for 7 and 8 processors; however, this might be a round-off error in the calculation of the time. Finally, for 8 processors the computational time is 4 min and 53 s.

Although the author performed the whole sensitivity analysis with only one mode, he also calculated five modes with Mesh 2, $\delta = 0.26$ and $PQ_{2,3}$. These

Table 6.56: Results for 2D C5G7 reactor with transport

Mesh	δ	Set	$\mathbf{k}_1$	EE_1 (pcm)	MPE (%)
1	1.0	S_2	1.179704	576.93	2.01
		$PQ_{1,2}$	1.180297	527.01	1.90
		$PQ_{1,3}$	1.180783	485.99	1.90
		$PQ_{1,4}$	1.180744	489.30	1.89
		$PQ_{2,2}$	1.180791	485.33	1.87
		$PQ_{2,3}$	1.181243	447.29	1.86
		S_4	1.180217	533.74	1.67
		S_6	1.180474	512.07	1.77
	0.24	S_2	1.182529	338.91	1.94
		$PQ_{1,2}$	1.182575	334.99	0.99
		$PQ_{1,3}$	1.183769	234.36	1.08
		$PQ_{1,4}$	1.183730	237.66	1.07
		$PQ_{2,2}$	1.182871	310.06	1.03
		$PQ_{2,3}$	1.183967	217.66	1.08
		S_4	1.182089	375.99	0.83
		S_6	1.182405	349.36	0.90
2	1.0	S_2	1.178923	642.83	1.66
		$PQ_{1,2}$	1.179635	582.79	1.41
		$PQ_{1,3}$	1.180104	543.27	1.40
		$PQ_{1,4}$	1.180061	546.85	1.38
		$PQ_{2,2}$	1.180196	535.49	1.50
		$PQ_{2,3}$	1.180633	498.67	1.47
		S_4	1.179590	586.61	1.27
		S_6	1.179861	563.77	1.39
	0.26	S_2	1.181966	386.37	1.67
		$PQ_{1,2}$	1.182149	370.90	0.56
		$PQ_{1,3}$	1.183289	274.85	0.61
		$PQ_{1,4}$	1.183242	278.76	0.60
		$PQ_{2,2}$	1.182486	342.54	0.69
		$PQ_{2,3}$	1.183525	254.97	0.72
		S_4	1.181681	410.36	0.47
		S_6	1.182002	383.28	0.53

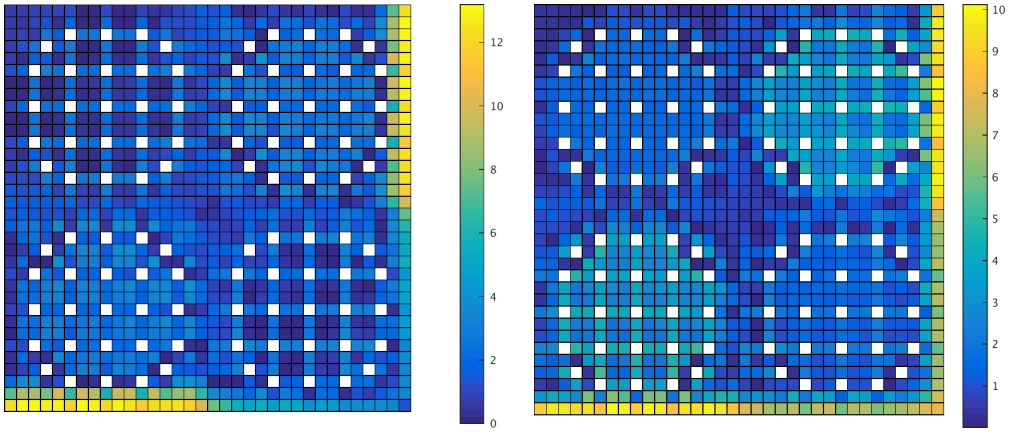


Figure 6.81: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 1 and S_2

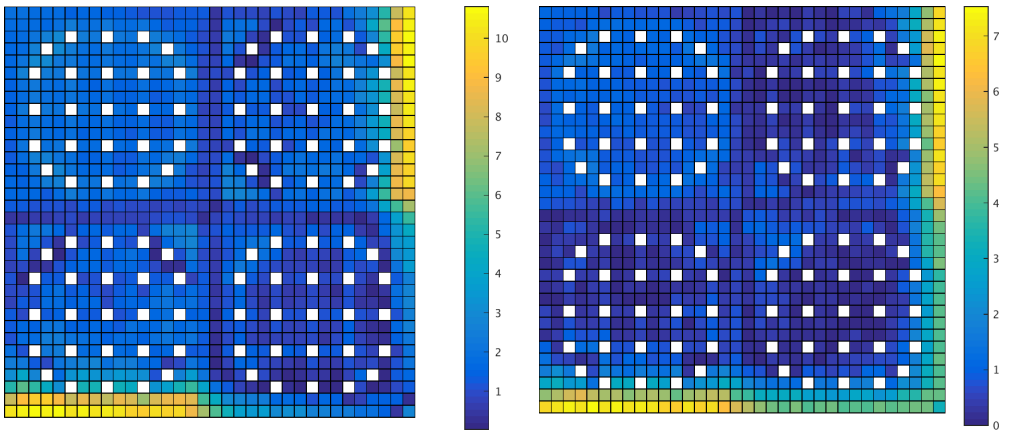
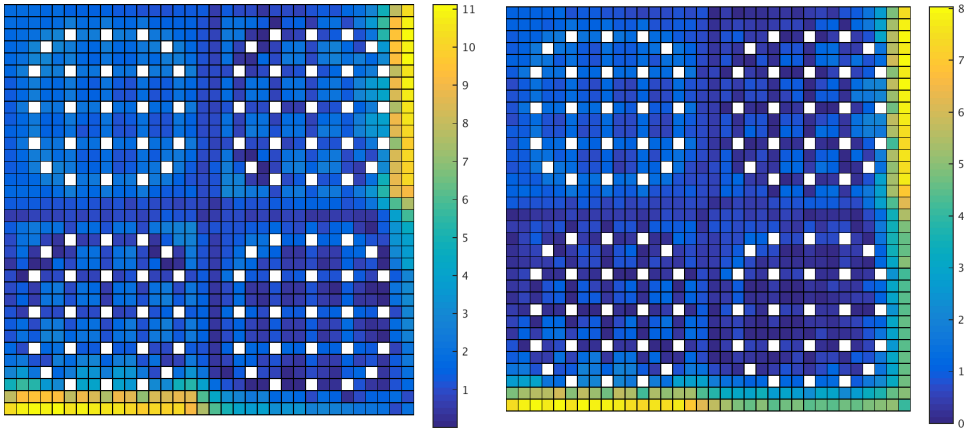
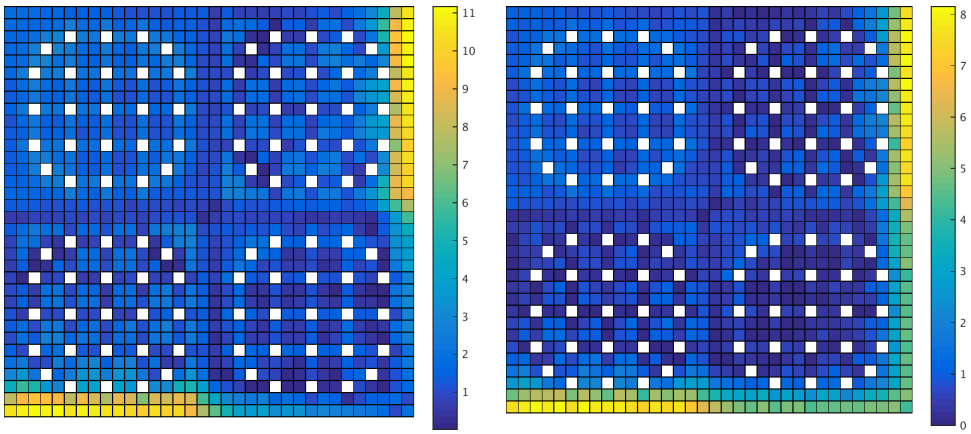


Figure 6.82: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 1 and $PQ_{1,2}$



$\delta = 1.0$ $\delta = 0.24$

Figure 6.83: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 1 and $PQ_{1,3}$



$\delta = 1.0$ $\delta = 0.24$

Figure 6.84: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 1 and $PQ_{1,4}$

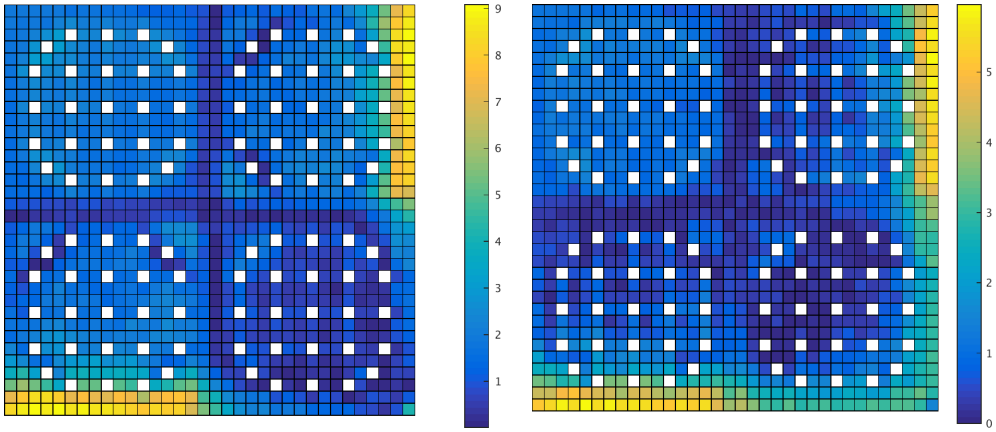


Figure 6.85: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 1 and $PQ_{2,2}$

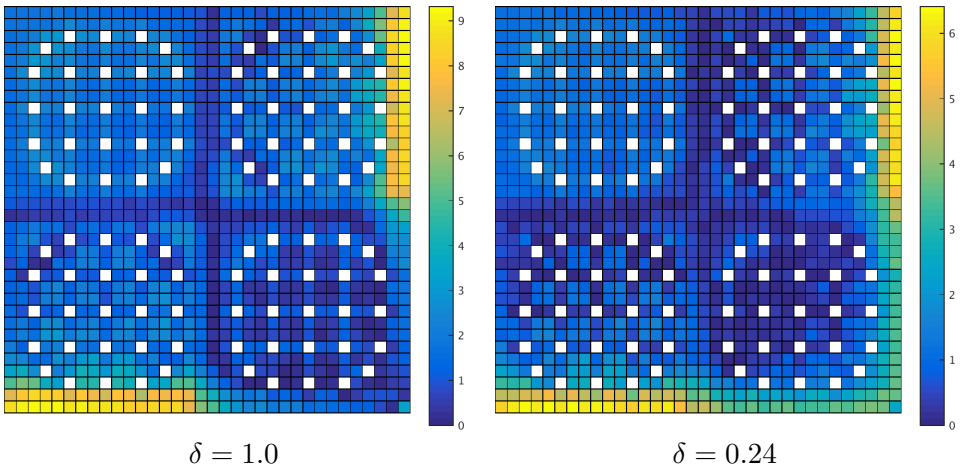
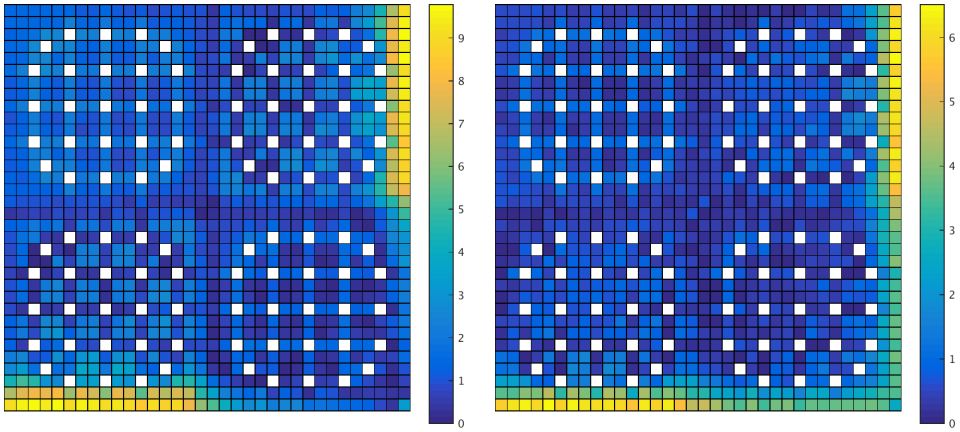
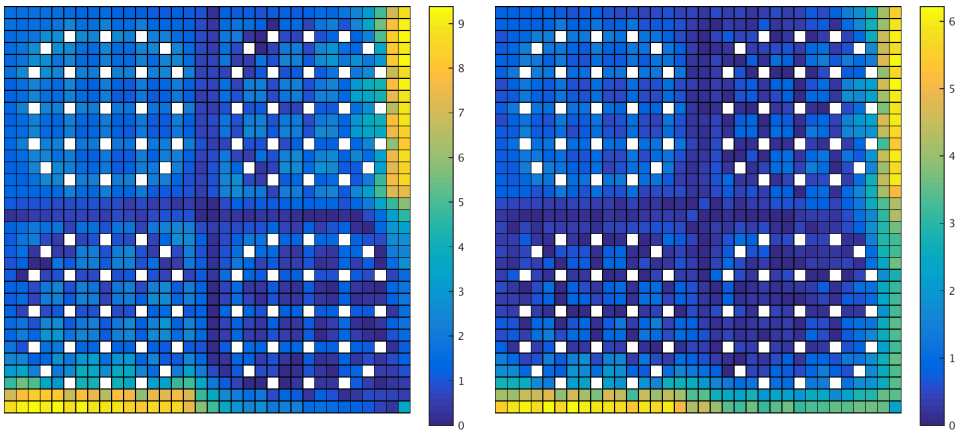


Figure 6.86: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 1 and $PQ_{2,3}$



$\delta = 1.0$ $\delta = 0.24$

Figure 6.87: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 1 and S_4



$\delta = 1.0$ $\delta = 0.24$

Figure 6.88: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 1 and S_6

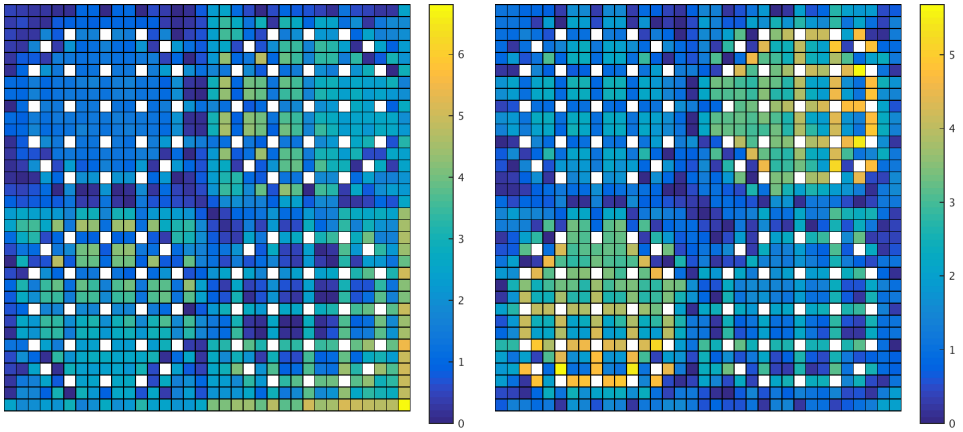


Figure 6.89: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 2 and S_2

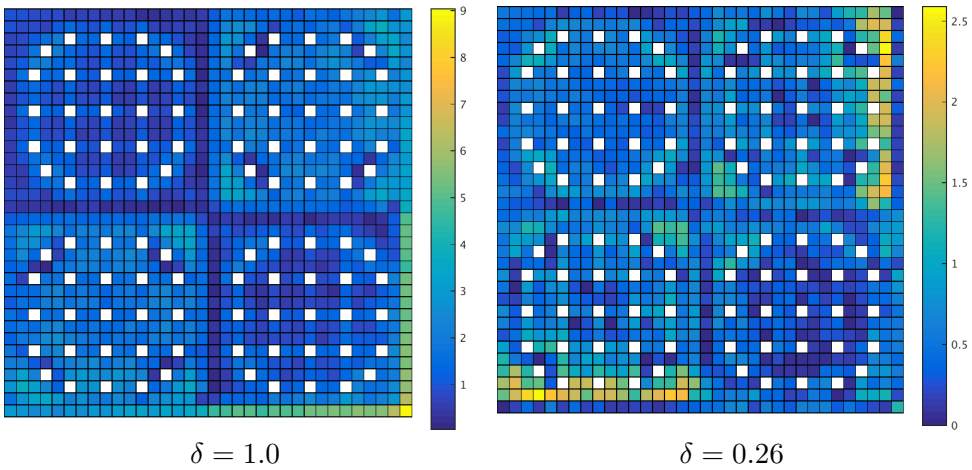
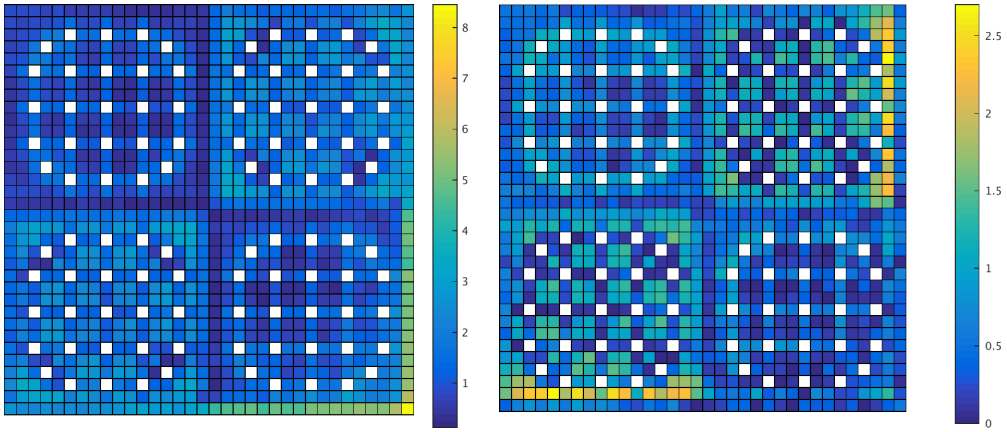
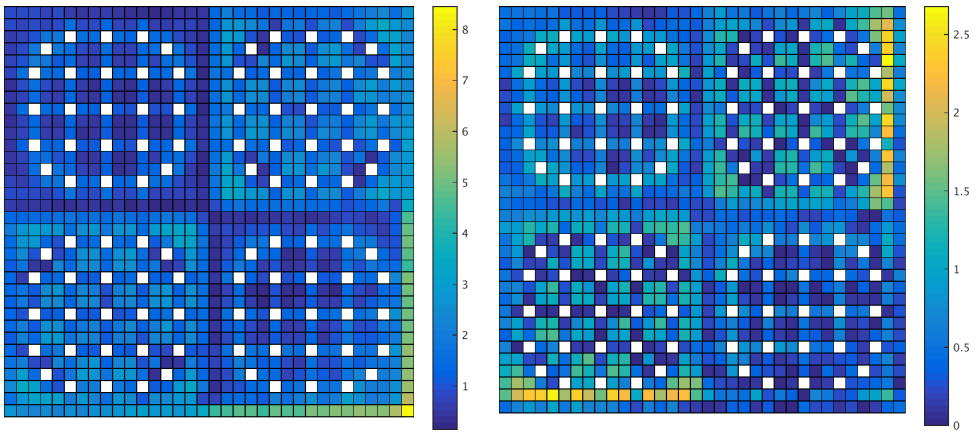


Figure 6.90: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 2 and $PQ_{1,2}$



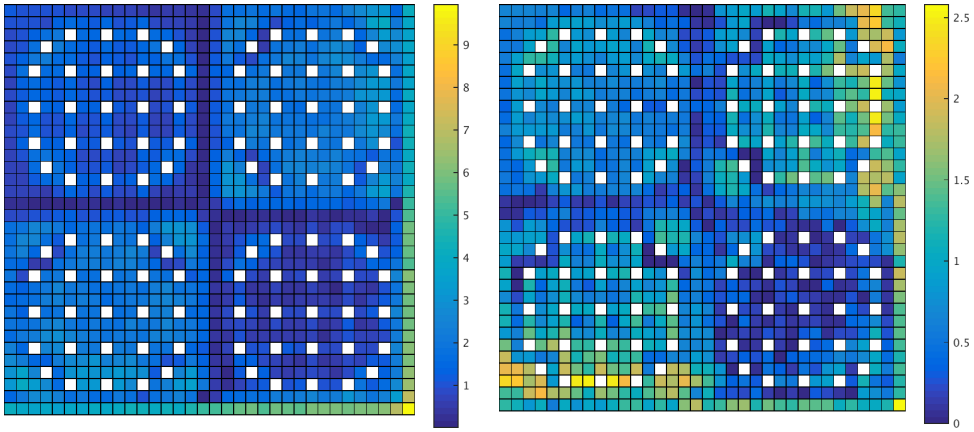
$\delta = 1.0$ $\delta = 0.26$

Figure 6.91: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 2 and $PQ_{1,3}$



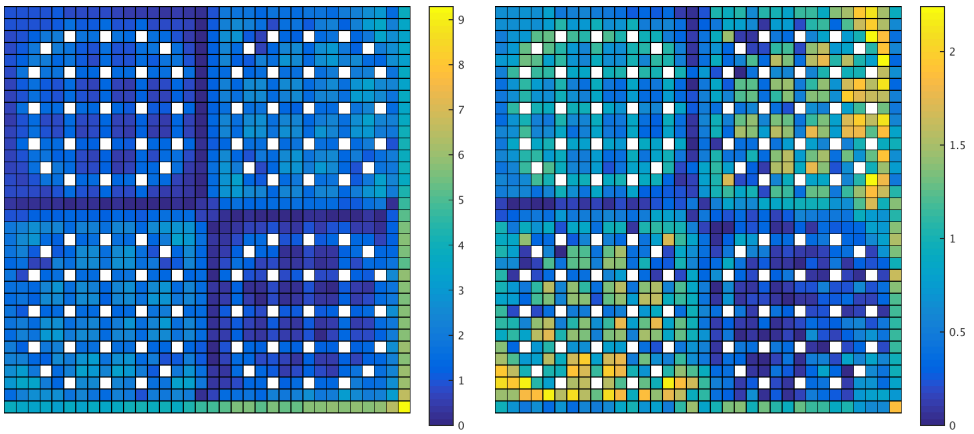
$\delta = 1.0$ $\delta = 0.26$

Figure 6.92: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 2 and $PQ_{1,4}$



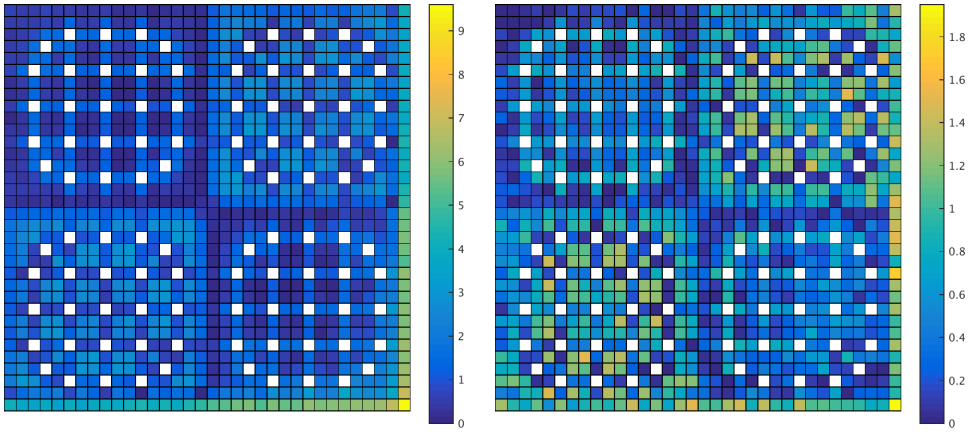
$\delta = 1.0$ $\delta = 0.26$

Figure 6.93: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 2 and $PQ_{2,2}$



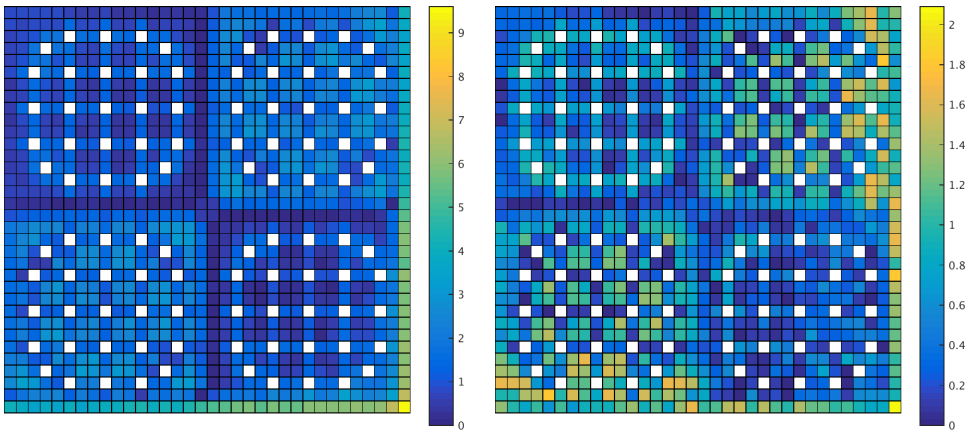
$\delta = 1.0$ $\delta = 0.26$

Figure 6.94: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 2 and $PQ_{2,3}$



$\delta = 1.0$ $\delta = 0.26$

Figure 6.95: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 2 and S_4



$\delta = 1.0$ $\delta = 0.26$

Figure 6.96: Power errors (%) for 2D C5G7 reactor with transport, for Mesh 2 and S_6

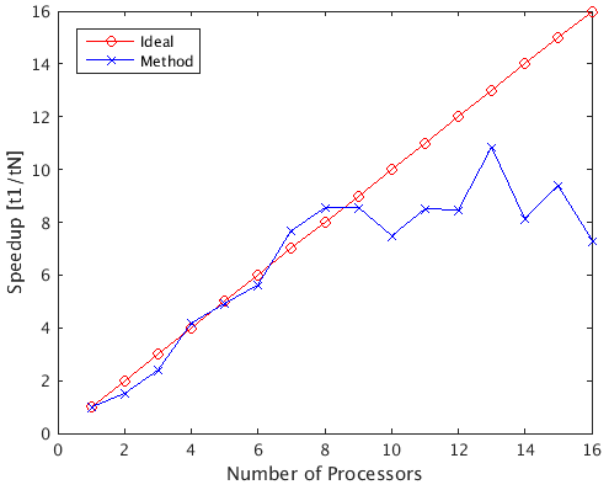


Figure 6.97: Speedup for 2D C5G7 reactor, for Mesh 1, $\delta = 0.24$, $PQ_{2,2}$ and the calculation of one mode

eigenvalues are: 1.183525, 0.909022, 0.870758, 0.716367 and 0.582865. These values correspond to the quarter of the core of this reactor.

Chapter 7

Conclusions

7.1 Conclusions

The author developed a code for solving the Neutron Diffusion and Transport Equations with the following features:

- Application to any geometry, 2D or 3D, discretized with structured and unstructured meshes.
- Use of the Finite Volume Method for discretizing the spatial derivatives terms.
- Use of the Discrete Ordinates method for discretizing the directional terms of the Neutron Transport Equation.
- General multigroup formulation including any upscattering and fission neutrons production.
- Calculation of multiple modes (eigenvalues and eigenvectors) of the Steady State of the Neutron Diffusion and Transport Equations.
- Use of a Modal Method for calculating the time-dependent Neutron Diffusion Equation.

- Parallel computation.
- Use of the state of the art algorithms for solving linear algebra operations.

The code uses a grid generator to model any kind of reactor by discretizing its geometry in a set of simple elements such as triangles, quadrangles, tetrahedra or hexahedra. In particular, the code uses Gmsh because of four reasons. First, it is fast and user-friendly. Second, it can generate meshes with different kind of elements. Third, it is free and open-source. Fourth, the developers keep it up to date.

On the one hand, the author developed three different FVMs for discretizing the Neutron Diffusion Equation: one based on the Moving Least Squares method, an inter-cells polynomial expansion method, and an improved inter-cells polynomial expansion method.

One summarizes the main features of the method based on the Moving Least Squares method as follows:

- Two types of equations for each energy group: cell equations for the Neutron Diffusion Equation and face equations for the boundary conditions.
- Use of the Moving Least Squares method for calculating the face averaged values of the neutron flux and the gradient of the neutron flux.
- Calculation of one face averaged value of the gradient of the neutron flux for each face.
- Enforcement of the current continuity condition by means of the diffusion coefficient for each inner face.
- Four approaches for calculating the diffusion coefficient for each inner face: Cell i, Cell l, Homogenized and Linear.

The main features of the inter-cells polynomial expansion method are the following:

- Expansion of the neutron flux, for each energy group and cell, with a finite series of 3D monomials. The expansion is limited to the number of faces of the cell plus one. In this expansion, the monomials are fixed and the coefficients are the unknowns.
- Use of four type of equations for each energy group: cell equations for the Neutron Diffusion Equation, boundary conditions for the boundary

faces, current continuity condition for the inner faces and flux continuity condition for the inner faces.

- Analytical calculation of the cell and face averaged values of the neutron flux and the gradient of the neutron flux.

The improved inter-cells polynomial expansion method uses the same polynomial expansion as the previous method, but it has the following differences:

- Two type of equations for each energy group: cell equations for the Neutron Diffusion Equation and face equations for the flux continuity condition.
- Implicit enforcement of the current continuity and the boundary conditions.
- The unknowns are the cell averaged values of the neutron flux and the face averaged values of the neutron current.
- The matrices have better condition number and lower size.

To solve the Steady State of the Neutron Diffusion Equation, the code solves the eigenvalue problem of the Neutron Diffusion Equation, which is discretized by any of the previous methods, and for a general multigroup formulation. The main characteristics of this calculation are the following:

- Use of the Krylov-Schur method implemented in the SLEPc library.
- The size of vectors used in the Krylov subspace does not depend on the number of energy groups.
- General multigroup formulation: any upscattering and fission neutrons production.
- Use of the linear algebra algorithms of PETSc to calculate vector and matrix operations, and to solve linear systems.
- Calculation of multiple eigenvalues and eigenvectors.

Furthermore, the author performed the parallelization of the whole code, which is summarized as follows:

- Based on MPI, and particularly on the parallel objects and algorithms of PETSc.
- Running the tasks in several CPUs: geometry pre-processing, equations discretization, eigenvalue and linear system solvers, linear algebra operations and post-processing.
- Storage of the data in several CPUs. These data are classified in six different types: cells, faces, nodes, materials, regions and inner faces. All these types of data are stored in all the CPUs, but each one with its local dimension.

As regards the Modal Method, the following points summarize the main features of the method:

- Solution of the general multigroup formulation.
- Expansion of the neutron flux as a sum of the product of the eigenvectors and time amplitudes for each eigenvector.
- Obtaining a reduced system of time-dependent Ordinary Differential Equations by means of the product of the adjoint eigenvectors.
- Calculation of the time amplitudes by solving the reduced system with the matrix exponential method of SLEPc.
- Fast adjoint calculation based on the product of the adjoint matrix and the forward eigenvectors, which includes a reorthogonalization to accomplish the biorthogonal property.
- Updating of modes and recalculation of the initial conditions, for transient with strong local variations of the cross sections.

On the other hand, the author applied the Discrete Ordinates method to the Neutron Transport Equation for discretizing the directional terms. In addition, the author applied the FVM for discretizing the spatial derivatives terms of the previous equation. The main features of this formulation are the following:

- It solves the Neutron Transport Equation, for each energy group, direction and cell. The unknowns are the cell averaged values of the angular flux.
- General multigroup formulation.

- Use of two type of quadrature sets: level-symmetric and product quadrature based on Gauss-Legendre.
- The author developed the product quadrature based on Gauss-Legendre.
- Simple formulation: all the angular terms are included in one term, which one can calculate a priori; the formulation includes a simplification of this term for 2D cases.
- Reflective boundary conditions, even for quadrature sets without symmetric directions.
- Two interpolation schemes for calculating the face averaged values of the angular flux:
  - Linear-step:
    - Interpolation scheme for the inner faces.
    - Combination of the upwind (step) and the central difference (linear) schemes.
    - Use of a factor δ to change the contribution of each scheme. This value ranges from 0 to 1. When $\delta = 1$, the method is equivalent to the step method.
  - Multislope second order upwind:
    - Interpolation scheme for the boundary faces.
    - It is a second order upwind, but considering the slopes of all the neighbor cells.
    - Calculation of a 1D slope as a weighted sum of the slopes connecting the adjacent cell to the boundary face, with the adjacent cells to this cell.
- Calculation of multiple eigenvalues and eigenvectors with the same algorithms as those used for the Neutron Diffusion Equation.

Finally, the author tested the algorithms in several 2D and 3D reactors, with different multigroup formulations and geometries. This evaluation included the use of computational resources and accuracy of the solution. The author evaluated the use of computational resources by means of the computational time and the speedup. The computational time evaluates the computational cost, whereas the speedup assesses the parallel performance. As regards the

accuracy of the solution, the author checked the accuracy of the crucial variables in Nuclear Safety Analyses: the multiplication factor (largest eigenvalue) and the power. In addition, since the code of this thesis can calculate multiple eigenvalues, the author tested the accuracy for the calculation of five eigenvalues. For evaluating these variables, the author used eigenvalue errors, power errors and the mean power error. These errors are calculated by using a reference solution, which is calculated with already validated codes. Furthermore, the author performed a sensitivity analysis of the following parameters: mesh, parameters of the FVM, linear system solvers and parameters of the Modal Method. These parameters of the Modal Method are the number of modes of the expansion and the time step for updating the modes. From the results, one draws the following conclusions:

- For the method based on the Moving Least Squares method:
  - It obtains accurate eigenvalue and power results for fine meshes.
  - The diffusion coefficient is almost insensitive for fine meshes, but one obtains the best results with Homogenized.
  - One has to use direct methods to solve the eigenvalue problem, because the matrices are not well-conditioned.
- For the inter-cells polynomial expansion method:
  - It obtains accurate eigenvalue and power results for fine meshes.
  - There are only few polynomial sets, containing monomials of second order, which give valid results.
  - The polynomial sets are almost insensitive for fine meshes. However, for coarse meshes there are some sets obtaining better results for certain eigenvalues in meshes composed of tetrahedra.
  - One has to use direct methods to solve the eigenvalue problem, because the matrices are not well-conditioned.
  - It gives more accurate results than the method based on the Moving Least Squares method, for the same meshes and in some cases for coarser meshes.
- For the improved inter-cells polynomial expansion method:

  - It gives the same results as those of the inter-cells polynomial expansion method, but with much lower computational time.
  - The matrices have better condition number and lower size than those of the inter-cells polynomial expansion method.
  - One can use iterative methods. The author tested all the iterative methods of PETSc, and he concluded that the fastest one is GMRES with Additive Schwarz preconditioner.
  - The general multigroup formulation provides accurate results with competitive computational times.
  - The parallel performance is close to ideal up to 5 processors and a reasonably good performance gain is seen up to 14 processors in the strong scaling sense.

- For the Modal Method:
  - The author applied this method to the equations discretized with the improved inter-cells polynomial expansion method.
  - The adjoint calculation accomplishes the biorthogonal property and the equation of the eigenvalue problem, even in heterogeneous reactors.
  - The author applied the method in a full size reactor of a benchmark multigroup transient. Accurate results were obtained.
  - One has to use fine meshes for obtaining accurate results.
  - One has to update the modes at least each 0.1 s to obtain accurate results in the studied case.
  - The effect of the number of modes was not remarkable on the analyzed transient.
  - The parallel computing performance was particularly good up to 8 processors. Actually, with more than 2 processors the code is faster than the NRC's PARCS code with the Nodal Expansion Multigroup solver.

- For the Neutron Transport Equation with the Discrete Ordinates and FVM:

- One obtains the most accurate results with the finer meshes and the lowest values of δ.
- This δ should be above certain value to not provide results with non physical sense.
- The method is simple and might be fast, but the main drawback is that one does not know a priori the best value of δ, so one has to find out this value by experimental testing.
- The product quadrature based on Gauss-Legendre gives better eigenvalue results than the level-symmetric, for the same number of directions. However, the power results might be slightly worse, but accurate enough.
- The method was applied to two reactors for its validation. The first reactor was validated by means of a code-to-code comparison with TITAN. The second reactor is the 2D reactor of the C5G7 benchmark; the reference results of this benchmark were obtained with the MCNP code.

7.2 Future work

As future work of this thesis, the author thinks that one should develop the following ideas:

- Application of the Modal Method to instability analyses.
- Development of a solver of eigenvalue problems with a restart capability of several modes.
- Add the moving-meshes capability for transients consisting in movement of control rods.
- Development of a Modal Method to solve the time-dependent Neutron Transport Equation.
- Coupling the code developed in this thesis with a thermal-hydraulic code.
- Application of the exponential matrix method to solve the whole system of time-dependent Ordinary Differential Equation, that is, without reducing the system.

- Development of new interpolation schemes for estimating the face averaged values of the angular flux:
  - Expansion of the angular flux with polynomials depending on space variables.
  - Use of High Order schemes of the FVM, such as MUSCL or WENO.
  - Use of Artificial Intelligence to get a matrix response for calculating the face averaged values from the cell averaged values. For example, Deep learning.
- Definition of circles, cylinders and spheres as new elements in the meshes to avoid the discretization of them.
- Application of Jacobian Free Newton Krylov methods.
- Expansion of the neutron flux with 4D monomials depending on space and time variables to solve at the same time the spatial and time distribution of the neutron population.
- Calculation of fixed-source problems of the Neutron Transport Equation.
- Development of a Photon Transport Code.
- Development of lattice calculation tools for cross sections processing, such as homogenization and collapse.
- High fidelity instability analyses based on three issues. First, solution of the initial adjoint eigenvalue problem to obtain accurately the adjoint eigenvectors. Second, calculation of the time and space distribution of the neutron flux with the matrix exponential method, applied to the whole system. Third, calculation of the time amplitudes for each mode by multiplying the adjoint eigenvectors and the neutron flux at certain time.
- Development of new methods for solving linear systems or calculating the inverse of a matrix, such us the use of rational matrix functions.

7.3 Scientific contribution

During the development of this thesis, the author contributed 7 publications to international scientific journals:

- Methodology to resolve the transport equation with the discrete ordinates code TORT into the IPEN/MB-01 reactor.
  - Authors: A. Bernal, A. Abarca, T. Barrachina and R. Miró.
  - Journal: International Journal of Computer Mathematics.
  - Year: 2014.
- Resolution of the Generalized Eigenvalue Problem in the Neutron Diffusion Equation Discretized by the Finite Volume Method.
  - Authors: Á. Bernal, R. Miró, D. Ginestar and G. Verdú.
  - Journal: Abstract and Applied Analysis.
  - Year: 2014.
- Development of a finite volume inter-cell polynomial expansion method for the neutron diffusion equation.
  - Authors: A. Bernal, J. E. Roman, R. Miró, D. Ginestar and G. Verdú.
  - Journal: Journal of Nuclear Science and Technology.
  - Year: 2016.
- Assembly Discontinuity Factors for the Neutron Diffusion Equation discretized with the Finite Volume Method. Application to BWR.
  - Authors: A. Bernal, J. E. Roman, R. Miró and G. Verdú.
  - Journal: Annals of Nuclear Energy.
  - Year: 2016.
- Multigroup neutron diffusion equation with the finite volume method in reactors using MOX fuels.

  - Authors: A. Bernal, J. E. Roman, R. Miró and G. Verdú.
  - Journal: Journal of Nuclear Science and Technology.
  - Year: 2017.

- A Krylov-Schur solution of the eigenvalue problem for the neutron diffusion equation discretized with the Raviart-Thomas method.
  - Authors: A. Bernal, A. Hébert, J. E. Roman, R. Miró and G. Verdú.
  - Journal: Journal of Nuclear Science and Technology.
  - Year: 2017.

- Calculation of multiple eigenvalues of the neutron diffusion equation discretized with a parallelized finite volume method.
  - Authors: A. Bernal, J. E. Roman, R. Miró and G. Verdú.
  - Journal: Progress in Nuclear Energy.
  - Year: 2018.

Furthermore, the author also presented 14 works in international conferences:

- Methodology to resolve the transport equation with the discrete ordinates code TORT into KRITZ reactor.
  - Authors: A. Bernal, A. Abarca, T. Barrachina, R. Miró and G. Verdú.
  - Conference: Mathematical Modelling in Engineering & Human Behaviour 2012.

- Resolution of the generalized eigenvalue problem in the neutron diffusion equation discretized by the Finite Volume Method.
  - Authors: A. Bernal, R. Miró, D. Ginestar and G. Verdú.
  - Conference: Mathematical Modelling in Engineering & Human Behaviour 2013.

- Determination of PWR core water level using ex-core detectors signals.

  - Authors: A. Bernal, A. Abarca, R. Miró and G. Verdú.
  - Conference: 6th International Nuclear Atlantic Conference (INAC 2013).

- An inter-cells polynomial expansion method for the steady-state 2 energy-group neutron diffusion equation discretized by the Finite Volume Method.
  - Authors: A. Bernal, J. E. Roman, R. Miró, D. Ginestar and G. Verdú.
  - Conference: Mathematical Modelling in Engineering & Human Behaviour 2014.

- Generalized and standard multigroup neutron diffusion equation eigenvalue problem with the finite volume method.
  - Authors: A. Bernal, R. Miró, D. Ginestar and G. Verdú.
  - Conference: ANS Reactor Physics Topical Meeting (PHYSOR 2014). The Role of Reactor Physics towards a Sustainable Future.

- A polynomial expansion method based on Helmholtz equation for the Neutron Diffusion Equation discretized by the Finite Volume Method.
  - Authors: A. Bernal, J. E. Roman, R. Miró and G. Verdú.
  - Conference: Mathematical Modelling in Engineering & Human Behaviour 2015.

- Eigenvalue problem of the neutron diffusion equation discretized with the finite volume method in a VVER.
  - Authors: A. Bernal, J. E. Roman, R. Miró and G. Verdú.
  - Conference: International Congress on Advances in Nuclear Power Plants (ICAPP 2016).

- Eigenvalue problem of the neutron diffusion equation with ADF and discretized with the FVM. Application to a PWR modelled with unstructured grid.

  - Authors: A. Bernal, J. E. Roman, R. Miró, G. Verdú and J. A. Bermejo.
  - Conference: Physics of Reactors conference. Unifying Theory and Experiments in the 21st Century (PHYSOR 2016).

- Dose rate analysis in a high capacity nuclear spent fuel storage system using the MAVRIC code.
  - Authors: A. Bernal, A. Abarca, R. Miró and G. Verdú.
  - Conference: Physics of Reactors conference. Unifying Theory and Experiments in the 21st Century (PHYSOR 2016).

- A spectral method for the steady state of the 2 energy-group neutron diffusion equation.
  - Authors: A. Bernal, R. Miró and G. Verdú.
  - Conference: Mathematical Modelling in Engineering & Human Behaviour 2016.

- Calculation of non-fundamental Modes in TRIVAC5 with SLEPc.
  - Authors: A. Bernal, A. Hébert, J. E. Roman, R. Miró and G. Verdú.
  - Conference: International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2017).

- Solution of the eigenvalue problem and linear systems in the neutron diffusion equation with high performance libraries.
  - Authors: A. Bernal, J. E. Roman, R. Miró and G. Verdú.
  - Conference: Congress on Numerical Methods in Engineering (CMN 2017).

- Calculation of the adjoint flux of the neutron diffusion equation.
  - Authors: A. Bernal, J. E. Roman, R. Miró and G. Verdú.

  - Conference: Mathematical Modelling in Engineering & Human Behaviour 2017.

- VALKIN-FVM: A Modal Finite Volume Method for solving the transient Neutron Diffusion Equation on unstructured meshes.

  - Authors: A. Bernal, J. E. Roman, R. Miró and G. Verdú.

  - Conference: PHYSOR 2018: Reactor Physics paving the way towards more efficient systems.

Bibliography

Ahrens, James et al. (2005). "36-paraview: An end-user tool for large-data visualization". In: *The visualization handbook* 717 (cit. on p. 77).

Alcouffe, Ray E et al. (1995). "DANTSYS: a diffusion accelerated neutral particle transport code system". In: *Los Alamos National Laboratory, LA-12969-M* (cit. on p. 24).

Alcouffe, Ray E et al. (2005). "PARTISN: A time-dependent, parallel neutral particle transport code system". In: *Los Alamos National Laboratory, LA-UR-05-3925 (May 2005)* (cit. on p. 24).

Amestoy, Patrick R, Iain S Duff, and J-Y L'excellent (2000). "Multifrontal parallel distributed symmetric and unsymmetric solvers". In: *Computer methods in applied mechanics and engineering* 184.2-4, pp. 501–520 (cit. on pp. 40, 75, 119, 135, 159).

Anderson, E. et al. (1999). *LAPACK Users' Guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics. ISBN: 0-89871-447-8 (paperback) (cit. on p. 40).

Anselone, PM and LB Rall (1968). "The solution of characteristic value-vector problems by Newton's method". In: *Numerische Mathematik* 11.1, pp. 38–45 (cit. on p. 37).

Aragonés, José M and Carol Ahnert (1986). "A linear discontinuous finite difference formulation for synthetic coarse-mesh few-group diffusion calculations". In: *Nuclear Science and Engineering* 94.4, pp. 309–322 (cit. on p. 13).

Aragonés, José M, Carol Ahnert, and Nuria García-Herranz (2007). "The analytic coarse-mesh finite difference method for multigroup and multidimensional diffusion calculations". In: *Nuclear Science and Engineering* 157.1, pp. 1–15 (cit. on p. 13).

Askew, JR (1972). *A characteristics formulation of the neutron transport equation in complicated geometries*. Tech. rep. United Kingdom Atomic Energy Authority (cit. on p. 23).

Balay, Satish et al. (2017). *PETSc Users Manual*. Tech. rep. ANL-95/11 - Revision 3.8. Argonne National Laboratory (cit. on pp. 40, 44, 75).

Bennewitz, F, H Finnemann, and H Moldaschl (1975). *Solution of the multidimensional neutron diffusion equation by nodal expansion*. Tech. rep. (cit. on p. 12).

Berkenbosch, AC, EF Kaasschieter, and JHM ten Thije Boonkkamp (1994). "Finite-difference methods for one-dimensional hyperbolic conservation laws". In: *Numerical Methods for Partial Differential Equations* 10.2, pp. 225–269 (cit. on p. 34).

Bernal, A et al. (2016a). "Assembly Discontinuity Factors for the Neutron Diffusion Equation discretized with the Finite Volume Method. Application to BWR". In: *Annals of Nuclear Energy* 97, pp. 76–85 (cit. on pp. 68, 143).

Bernal, Álvaro et al. (2014). "Resolution of the generalized eigenvalue problem in the neutron diffusion equation discretized by the finite volume method". In: *Abstract and Applied Analysis*. Vol. 2014. Hindawi (cit. on pp. 53, 119).

Bernal, Álvaro et al. (2016b). "Development of a finite volume inter-cell polynomial expansion method for the neutron diffusion equation". In: *Journal of Nuclear Science and Technology* 53.8, pp. 1212–1223 (cit. on pp. 64, 135).

Bernal, Álvaro et al. (2017a). "A Krylov–Schur solution of the eigenvalue problem for the neutron diffusion equation discretized with the Raviart–Thomas method". In: *Journal of Nuclear Science and Technology* 54.10, pp. 1085–1094 (cit. on pp. 38, 117, 166).

Bernal, Álvaro et al. (2017b). "Multigroup neutron diffusion equation with the finite volume method in reactors using MOX fuels". In: *Journal of Nuclear Science and Technology* 54.11, pp. 1251–1260 (cit. on pp. 72, 157).

Bernal, Alvaro et al. (2018). "Calculation of multiple eigenvalues of the neutron diffusion equation discretized with a parallelized finite volume method". In: *Progress in Nuclear Energy* 105, pp. 271–278 (cit. on pp. 39, 77, 170).

Blackford, L. S. et al. (1997). *ScaLAPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics. ISBN: 0-89871-397-8 (paperback) (cit. on p. 40).

Briesmeister Judith, F MCNPTM (2000). "A general Monte Carlo N-Particle transport code". In: *LA12625-M, Version B* 4, pp. 2–27 (cit. on pp. 159, 194).

Brown, Forrest B et al. (2003). "MCNP–A General Monte Carlo N-Particle Transport Code, Version 5". In: *Los Alamos National Laboratory, Oak Ridge, TN* (cit. on p. 24).

Butcher, J.C. (2008). *Numerical Methods for Ordinary Differential Equations*. Wiley. ISBN: 9780470753750 (cit. on pp. 42, 43).

Cacuci, D.G. (2010). *Handbook of Nuclear Engineering: Vol. 1: Nuclear Engineering Fundamentals; Vol. 2: Reactor Design; Vol. 3: Reactor Analysis; Vol. 4: Reactors of Generations III and IV; Vol. 5: Fuel Cycles, Decommissioning, Waste Disposal and Safeguards*. Handbook of Nuclear Engineering v. 3. Springer US. ISBN: 9780387981307 (cit. on pp. 5, 6, 15).

Carreño, A et al. (2017). "Spatial modes for the neutron diffusion equation and their computation". In: *Annals of Nuclear Energy* 110, pp. 1010–1022 (cit. on p. 39).

Chao, YA and YA Shatilla (1995). "Conformal mapping and hexagonal nodal methods II: implementation in the ANC-H code". In: *Nuclear Science and Engineering* 121.2, pp. 210–225 (cit. on p. 171).

Cho, JY et al. (2003). "Three-Dimensional Whole Core Transport Calculation Methodology of the DeCART Code". In: *KAERI0TR-236502003, Korea Atomic Energy Research Institute* (cit. on p. 24).

Cueto-Felgueroso, Luis et al. (2007). "Finite volume solvers and moving least-squares approximations for the compressible Navier–Stokes equations on unstructured grids". In: *Computer Methods in Applied Mechanics and Engineering* 196.45-48, pp. 4712–4736 (cit. on pp. 32, 51).

DeHart, MD and MD Jessee (2005). "NEWT: a new transport algorithm for two-dimensional discrete ordinates analysis in non-orthogonal geometries". In: *ORNL/TM-2005/39, Oak Ridge National Laboratory* (cit. on p. 24).

Derstine, KL (2011). "DIF3D 10.0: Code System Using Variational Nodal Methods and Finite Difference Methods to Solve Neutron Diffusion and Transport Theory Problems". In: *CCC-784* (cit. on p. 13).

DiGiovine, AS et al. (1995). "SIMULATE-3 User's Manual". In: *Studsvik0SOA-95015, Studsvik of America, Inc* (cit. on p. 13).

Döring, Matthias G, Jens Chr Kalkkuhl, and Wolfram Schröder (1993). "Subspace iteration for nonsymmetric eigenvalue problems applied to the λ-eigenvalue problem". In: *Nuclear science and engineering* 115.3, pp. 244–252 (cit. on pp. 38, 88–90, 177).

Downar, T et al. (2006). "PARCS v2. 7 US NRC Core Neutronics Simulator User Manual". In: *Purdue University* (cit. on pp. 13, 117, 182).

Dulla, Sandra, Ernest H Mund, and Piero Ravetto (2008). "The quasi-static method revisited". In: *Progress in Nuclear Energy* 50.8, pp. 908–920 (cit. on p. 45).

Edenius, Malte et al. (1995). "CASMO-4, a fuel assembly burnup program, user's manual". In: *Studsvik0SOA-9501, Studsvik of America, Inc* (cit. on p. 24).

Eisenstat, Stanley C, Howard C Elman, and Martin H Schultz (1983). "Variational iterative methods for nonsymmetric systems of linear equations". In: *SIAM Journal on Numerical Analysis* 20.2, pp. 345–357 (cit. on p. 40).

Evans, Thomas M et al. (2010). "Denovo: A new three-dimensional parallel discrete ordinates code in SCALE". In: *Nuclear technology* 171.2, pp. 171–200 (cit. on p. 24).

Fletcher, Roger (1976). "Conjugate gradient methods for indefinite systems". In: *Numerical analysis*. Springer, pp. 73–89 (cit. on p. 40).

Gaskell, PH and AKC Lau (1988). "Curvature-compensated convective transport: SMART, A new boundedness-preserving transport algorithm". In: *International journal for numerical methods in fluids* 8.6, pp. 617–641 (cit. on p. 32).

Geuzaine, Christophe and Jean-François Remacle (2009). "Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities". In: *International journal for numerical methods in engineering* 79.11, pp. 1309–1331 (cit. on pp. 25, 48).

Gill, Daniel F and Yousry Y Azmy (2011). "Newton's method for solving k-eigenvalue problems in neutron diffusion theory". In: *Nuclear Science and Engineering* 167.2, pp. 141–153 (cit. on p. 39).

Gill, Daniel F et al. (2011). "Newton's Method for the Computation of k-Eigenvalues in SN Transport Applications". In: *Nuclear Science and Engineering* 168.1, pp. 37–58 (cit. on p. 39).

Ginestar, D et al. (1998). "High order backward discretization of the neutron diffusion equation". In: *Annals of Nuclear Energy* 25.1-3, pp. 47–64 (cit. on pp. 13, 44).

Goluoglu, Sedat and HL Dodds (2001). "A time-dependent, three-dimensional neutron transport methodology". In: *Nuclear science and engineering* 139.3, pp. 248–261 (cit. on p. 45).

Goluoglu, Sedat et al. (2011). "Monte Carlo criticality methods and analysis capabilities in SCALE". In: *Nuclear Technology* 174.2, pp. 214–235 (cit. on p. 24).

González-Pintor, S, D Ginestar, and G Verdú (2013). "Modified Block Newton method for the lambda modes problem". In: *Nuclear Engineering and Design* 259, pp. 230–239 (cit. on p. 39).

González-Pintor, S, Damián Ginestar, and G Verdú (2011). "Updating the Lambda Modes of a nuclear power reactor". In: *Mathematical and Computer Modelling* 54.7-8, pp. 1796–1801 (cit. on p. 39).

Grundmann, Ulrich et al. (2005). "DYN3D version 3.2-code for calculation of transients in light water reactors (LWR) with hexagonal or quadratic fuel elements-description of models and methods". In: (cit. on p. 13).

Gupta, Anurag and RS Modak (2011). "Evaluation of dominant time-eigenvalues of neutron transport equation by Meyer's sub-space iterations". In: *Annals of Nuclear Energy* 38.7, pp. 1680–1686 (cit. on p. 38).

Harvie, Dalton James Eric (2012). "An implicit finite volume method for arbitrary transport equations". In: *ANZIAM Journal* 52, pp. 1126–1145 (cit. on pp. 50, 51).

Hébert, A (1993). "Application of a dual variational formulation to finite element reactor calculations". In: *Annals of Nuclear Energy* 20.12, pp. 823–845 (cit. on p. 13).

Hébert, A (2008). "A Raviart–Thomas–Schneider solution of the diffusion equation in hexagonal geometry". In: *Annals of Nuclear Energy* 35.3, pp. 363–376 (cit. on p. 13).

Hébert, A. (2009). *Applied Reactor Physics*. Presses internationales Polytechnique. ISBN: 9782553014369 (cit. on pp. 5, 6, 10, 12, 15, 20, 23, 28, 44, 45).

Hébert, A and A Kavenoky (1981). *Development of the SPH homogenization method*. Tech. rep. CEA Centre d'Etudes Nucleaires de Saclay (cit. on p. 28).

Hébert, A and D Sekki (2010). "A user guide for Trivac Version4". In: *Institut de Génie Nucléaire, Tech. Rep. IGE-293* (cit. on pp. 13, 117, 165).

Hébert, Alain (1987). "Development of the nodal collocation method for solving the neutron diffusion equation". In: *Annals of Nuclear Energy* 14.10, pp. 527–541 (cit. on pp. 12, 117, 166).

Henry, A.F. (1975). *Nuclear-reactor analysis*. MIT Press. ISBN: 9780262080811 (cit. on p. 18).

Hernandez, V., J. E. Roman, and V. Vidal (2003). "SLEPc: Scalable Library for Eigenvalue Problem Computations". In: *Lect. Notes Comput. Sci.* 2565, pp. 377–391 (cit. on p. 40).

Hernandez, V. et al. (2005a). *Single Vector Iteration Methods in SLEPc*. Tech. rep. STR-2. Available at http://slepc.upv.es. Universitat Politècnica de València (cit. on pp. 35, 36).

— (2005b). *Subspace Iteration in SLEPc*. Tech. rep. STR-3. Available at http://slepc.upv.es. Universitat Politècnica de València (cit. on p. 36).

— (2006). *Lanczos Methods in SLEPc*. Tech. rep. STR-5. Available at http://slepc.upv.es. Universitat Politècnica de València (cit. on p. 37).

— (2007). *Krylov-Schur Methods in SLEPc*. Tech. rep. STR-7. Available at http://slepc.upv.es. Universitat Politècnica de València (cit. on p. 37).

Hernandez, Vicente, Jose E. Roman, and Vicente Vidal (2005). "SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems". In: *ACM Trans. Math. Software* 31.3, pp. 351–362 (cit. on pp. 40, 44).

Heroux, Michael et al. (2003). *An overview of Trilinos*. Tech. rep. Citeseer (cit. on p. 44).

Higham, Nicholas J (2009). "The scaling and squaring method for the matrix exponential revisited". In: *SIAM review* 51.4, pp. 747–764 (cit. on p. 86).

Hindmarsh, Alan C et al. (2005). "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers". In: *ACM Transactions on Mathematical Software (TOMS)* 31.3, pp. 363–396 (cit. on p. 44).

Hoffmann, K.A. and S.T. Chiang (2000). *Computational Fluid Dynamics*. Computational Fluid Dynamics v. 2. Engineering Education System. ISBN: 9780962373138 (cit. on pp. 28, 29).

Jessee, Matthew Anderson et al. (2014). *Polaris: a new two-dimensional lattice physics analysis capability for the SCALE code system*. Tech. rep. Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States) (cit. on p. 24).

Kavenoky, A (1980). *The SPH homogenization method*. Tech. rep. IAEA-TECDOC–231. International Atomic Energy Agency (IAEA) (cit. on p. 28).

Knoll, Dana A and David E Keyes (2004). "Jacobian-free Newton–Krylov methods: a survey of approaches and applications". In: *Journal of Computational Physics* 193.2, pp. 357–397 (cit. on p. 37).

Kochunas, Brendan et al. (2013). *Overview of development and design of MPACT: Michigan parallel characteristics transport code*. Tech. rep. American Nuclear Society, 555 North Kensington Avenue, La Grange Park, IL 60526 (United States) (cit. on p. 24).

Kophazi, Jozsef and Danny Lathouwers (2012). "Three-dimensional transport calculation of multiple alpha modes in subcritical systems". In: *Annals of Nuclear Energy* 50, pp. 167–174 (cit. on p. 38).

Koren, Barry (1993). "A robust upwind discretization method for advection, diffusion and source terms". In: *Department of Numerical Mathematics [NM]* R 9308 (cit. on p. 32).

Kozlowski, Tomasz and Thomas J Downar (2007). "PWR MOX/UO2 Core Transient Benchmark Final Report". In: *Nuclear Energy Agency, Organization for Economic Co-operation and Development, US Nuclear Regulatory Commission* (cit. on pp. 157, 164, 181).

Lamarsh, J.R. and A.J. Baratta (2001). *Introduction to Nuclear Engineering*. Addison-Wesley series in nuclear science and engineering. Prentice Hall. ISBN: 9780201824988 (cit. on pp. 5, 6).

Lathouwers, D (2003). "Iterative computation of time-eigenvalues of the neutron transport equation". In: *Annals of Nuclear Energy* 30.17, pp. 1793–1806 (cit. on p. 38).

Lathrop, Kaye D (1968). "Ray effects in discrete ordinates equations". In: *Nuclear Science and Engineering* 32.3, pp. 357–369 (cit. on p. 23).

Lautard, JJ, S Loubiere, and C Fedon-Magnaud (1990). "CRONOS2: A modular computational system for neutronic core calculations, 1993". In: *IAEA specialists meeting on advanced calculational methods for power reactors and LWR core design parameters* (cit. on p. 24).

Lawson, Chuck L et al. (1979). "Basic linear algebra subprograms for Fortran usage". In: *ACM Transactions on Mathematical Software (TOMS)* 5.3, pp. 308–323 (cit. on p. 40).

Lehoucq, Richard B, Danny C Sorensen, and Chao Yang (1998). *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. Vol. 6. Siam (cit. on p. 40).

Leppänen, Jaakko (2013). "Serpent–a continuous-energy Monte Carlo reactor physics burnup calculation code". In: *VTT Technical Research Centre of Finland* 4 (cit. on p. 24).

Lewis, EE et al. (2001). "Benchmark specification for Deterministic 2-D/3-D MOX fuel assembly transport calculations without spatial homogenization (C5G7 MOX)". In: *NEA/NSC* (cit. on pp. 157–159, 184, 192, 194).

Macian-Juan, Rafael and John H Mahaffy (1998). "Numerical diffusion and the tracking of solute fields in system codes: Part I. One-dimensional flows". In: *Nuclear engineering and design* 179.3, pp. 297–319 (cit. on p. 34).

Mahadevan, Vijay and Jean Ragusa (2008). "Novel hybrid scheme to compute several dominant eigenmodes for reactor analysis problems". In: (cit. on p. 39).

Marleau, Guy, Alain Hébert, and Robert Roy (2011). "A user guide for DRAGON Version 4". In: *IGE-294, Ecole Polytechnique de Montréal, Institut de génie nucléaire Département de génie mécanique (Aug. 26, 2016)* (cit. on p. 24).

Miró, Rafael et al. (2002). "A nodal modal method for the neutron diffusion equation. Application to BWR instabilities analysis". In: *Annals of Nuclear Energy* 29.10, pp. 1171–1194 (cit. on pp. 13, 83, 88, 117).

Miró, Rafael et al. (2006). *Parameterization of nuclear cross-sections for coupled neutronic-thermalhydraulic codes*. Tech. rep. American Nuclear Society, 555 North Kensington Avenue, La Grange Park, IL 60526 (United States) (cit. on p. 143).

Modak, RS and VK Jain (1996). "Sub-space iteration scheme for the evaluation of λ-modes of finite-differenced multi-group neutron diffusion equations". In: *Annals of Nuclear Energy* 23.3, pp. 229–237 (cit. on p. 38).

Moukalled, F., L. Mangani, and M. Darwish (2015). *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab*. Fluid Mechanics and Its Applications. Springer International Publishing. ISBN: 9783319168746 (cit. on pp. 25, 28, 29).

Müller, EZ and ZJ Weiss (1991). "Benchmarking with the multigroup diffusion high-order response matrix method". In: *Annals of Nuclear Energy* 18.9, pp. 535–544 (cit. on p. 125).

Ott, Karl (1966). "Quasistatic treatment of spatial phenomena in reactor dynamics". In: *Nuclear Science and Engineering* 26.4, pp. 563–565 (cit. on p. 45).

Palmiotti, G, EE Lewis, and CB Carrico (1995). "VARIANT: VARIational Anisotropic Nodal Transport". In: *Proceedings of the international conference on mathematics and computations, reactor physics, and environmental analyses*. Vol. 1 (cit. on p. 24).

Rhoades, Wayne A and DB Simpson (1997). *The TORT three-dimensional discrete ordinates neutron/photon transport code (TORT version 3)*. Tech. rep. Oak Ridge National Lab., TN (United States) (cit. on p. 24).

Roe, Philip L (1986). "Characteristic-based schemes for the Euler equations". In: *Annual review of fluid mechanics* 18.1, pp. 337–365 (cit. on p. 32).

Roman, J. E. et al. (2017). *SLEPc Users Manual*. Tech. rep. DSIC-II/24/02 - Revision 3.8. D. Sistemes Informàtics i Computació, Universitat Politècnica de València (cit. on p. 40).

Saad, Youcef (1983). "Projection methods for solving large sparse eigenvalue problems". In: *Matrix Pencils*. Springer, pp. 121–144 (cit. on p. 35).

Saad, Youcef and Martin H Schultz (1986). "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems". In: *SIAM Journal on scientific and statistical computing* 7.3, pp. 856–869 (cit. on pp. 40, 75).

Sanchez, Richard (2009). "Assembly homogenization techniques for core calculations". In: *Progress in Nuclear Energy* 51.1, pp. 14–31 (cit. on p. 27).

Sanchez, Richard et al. (1988). "APOLLO II: A user-oriented, portable, modular code for multigroup transport assembly calculations". In: *Nuclear Science and Engineering* 100.3, pp. 352–362 (cit. on p. 24).

Shampine, Lawrence F and Mark W Reichelt (1997). "The matlab ode suite". In: *SIAM journal on scientific computing* 18.1, pp. 1–22 (cit. on pp. 40, 44).

Shu, Chi-Wang (1998). "Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws". In: *Advanced numerical approximation of nonlinear hyperbolic equations*. Springer, pp. 325–432 (cit. on p. 33).

Singh, KP et al. (2009). "Iterative schemes for obtaining dominant alpha-modes of the neutron diffusion equation". In: *Annals of Nuclear Energy* 36.8, pp. 1086–1092 (cit. on p. 38).

Smith, Kord S (1986). "Assembly homogenization techniques for light water reactor analysis". In: *Progress in Nuclear Energy* 17.3, pp. 303–335 (cit. on pp. 27, 49).

Smith, Kord Sterling (1979). "An analytic nodal method for solving the two-group, multidimensional, static and transient neutron diffusion equations". PhD thesis. Massachusetts Institute of Technology (cit. on p. 12).

Sonneveld, Peter (1989). "CGS, a fast Lanczos-type solver for nonsymmetric linear systems". In: *SIAM journal on scientific and statistical computing* 10.1, pp. 36–52 (cit. on p. 40).

Stacey, W.M. (2007). *Nuclear Reactor Physics*. John Wiley & Sons. ISBN: 9783527406791 (cit. on pp. 5, 6, 9, 44, 45).

Stewart, Gilbert W (2002). "A Krylov–Schur algorithm for large eigenproblems". In: *SIAM Journal on Matrix Analysis and Applications* 23.3, pp. 601–614 (cit. on pp. 37, 75).

Sweby, Peter K (1984). "High resolution schemes using flux limiters for hyperbolic conservation laws". In: *SIAM journal on numerical analysis* 21.5, pp. 995–1011 (cit. on p. 32).

Theler, German (2013). "Unstructured grids and the multigroup neutron diffusion equation". In: *Science and Technology of Nuclear Installations* 2013 (cit. on p. 39).

Turinsky, Paul J et al. (1994). *NESTLE: Few-group neutron diffusion equation solver utilizing the nodal expansion method for eigenvalue, adjoint, fixed-source steady-state and transient problems*. Tech. rep. EG and G Idaho, Inc., Idaho Falls, ID (United States); Los Alamos National Lab., NM (United States) (cit. on p. 13).

Van Leer, Bram (1997). "Towards the ultimate conservative difference scheme". In: *Journal of Computational Physics* 135.2, pp. 229–248 (cit. on p. 32).

Verdú, G et al. (1994). "3D λ-modes of the neutron-diffusion equation". In: *Annals of Nuclear Energy* 21.7, pp. 405–421 (cit. on pp. 13, 38, 117).

Verdú, G et al. (1999). "The implicit restarted Arnoldi method, an efficient alternative to solve the neutron diffusion equation". In: *Annals of nuclear energy* 26.7, pp. 579–593 (cit. on p. 38).

Verdu, Gumersindo et al. (2010). "3D alpha modes of a nuclear power reactor". In: *Journal of nuclear science and technology* 47.5, pp. 501–514 (cit. on p. 38).

Verdú, G. et al. (1995). "A consistent multidimensional nodal method for transient calculations". In: *Annals of Nuclear Energy* 22.6, pp. 395 –410. ISSN: 0306-4549. DOI: https://doi.org/10.1016/0306-4549(94)00067-0 (cit. on p. 45).

Vidal-Ferrandiz, A et al. (2014). "Solution of the Lambda modes problem of a nuclear power reactor using an h–p finite element method". In: *Annals of Nuclear Energy* 72, pp. 338–349 (cit. on pp. 13, 38).

Vorst, Henk A Van der (1992). "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems". In: *SIAM Journal on scientific and Statistical Computing* 13.2, pp. 631–644 (cit. on p. 40).

Wagner, MR and K Koebke (1983). "Progress in nodal reactor analysis". In: *Atomkernenergie Kerntechnik* 43.2, pp. 117–126 (cit. on p. 27).

Wareing, Todd A, John M McGhee, and Jim E Morel (1996). "ATTILA: A three-dimensional, unstructured tetrahedral mesh discrete ordinates transport code". In: *Transactions of the American Nuclear Society* 75.CONF-961103– (cit. on p. 24).

Warsa, James S et al. (2004). "Krylov subspace iterations for deterministic k-eigenvalue calculations". In: *Nuclear Science and Engineering* 147.1, pp. 26–42 (cit. on p. 38).

Waterson, NP and H Deconinck (1995). "A unified approach to the design and application of bounded higher-order convection schemes". In: *Numerical methods in laminar and turbulent flow.* 9, pp. 203–214 (cit. on p. 32).

Yi, Ce (2009). "TITAN: A 3-D Deterministic Radiation Transport Code, TITAN User Manual Version 1.05". In: *Univ. of Florida* (cit. on pp. 21, 24, 117, 187).

Zhou, Gang (1995). *Numerical simulations of physical discontinuities in single and multi-fluid flows for arbitrary Mach numbers*. Chalmers University of Technology (cit. on p. 32).

Zimin, Vyacheslav G (2002). "SKETCH-N 1.0, Solve Neutron Diffusion Equations of Steady-State and Kinetics Problems". In: (cit. on p. 13).