Acceleration and higher order schemes of a characteristic solver for the solution of the neutron transport equation in 3D axial geometries

Daniele Sciannandrone

UNIVERSITÉ PARIS-SUD

ECOLE DOCTORALE 534: MODÉLISATION ET INSTRUMENTATION EN PHYSIQUE, ENERGIES, GÉOSCIENCES ET ENVIRONNEMENT
LABORATOIRE DE TRANSPORT STOCHASTIQUE ET DÉTERMINISTE (CEA SACLAY)

DISCIPLINE: PHYSIQUE

THÈSE DE DOCTORAT

Soutenue le 14/09/2015 par

Daniele Sciannandrone

Acceleration and higher order schemes of a characteristic solver for the solution of the neutron transport equation in 3D axial geometries

Directeur de thèse :	M. Richard SANCHEZ	Directeur de recherche (CEA Saclay)
Composition du jury :		
Président du jury :	Pierre DESESQUELLES	Professeur (Université Paris Sud)
Rapporteurs :	Jean-Concetto RAGUSA	Professeur (Texas A&M University)
	Piero RAVETTO	Professeur (Politecnico di Torino)
Examinateurs :	Simone SANTANDREA	Docteur,Ingénieur (CEA Saclay)
	Jean-François VIDAL	Docteur,Ingénieur (CEA Cadarache)

Abstract

The topic of our research is the application of the Method of Long Characteristics (MOC) to solve the Neutron Transport Equation in three-dimensional axial geometries. The strength of the MOC is in its precision and versatility. As a drawback, it requires a large amount of computational resources. This problem is even more severe in three-dimensional geometries, for which unknowns reach the order of tens of billions for assembly-level calculations.

The first part of the research has dealt with the development of optimized tracking and reconstruction techniques which take advantage of the regularities of three-dimensional axial geometries. These methods have allowed a strong reduction of the memory requirements and a reduction of the execution time of the MOC calculation. The convergence of the iterative scheme has been accelerated with a lower-order transport operator (DP_N) which is used for the initialization of the solution and for solving the synthetic problem during MOC iterations. The algorithms for the construction and solution of the MOC and DP_N operators have been accelerated by using shared-memory parallel paradigms which are more suitable for standard desktop working stations. An important part of this research has been devoted to the implementation of scheduling techniques to improve the parallel efficiency. The convergence of the angular quadrature formula for three-dimensional cases is also studied. Some of these formulas take advantage of the reduced computational costs of the treatment of planar directions and the vertical direction to speed up the algorithm. The verification of the MOC solver has been done by comparing results with continuous-in-energy Monte Carlo calculations. For this purpose a coupling of the 3D MOC solver with the Subgroup method is proposed to take into account the effects of cross sections resonances. The full calculation of a FBR assembly requires about $1h30$ of execution time with differences of few pcm with respect to the reference results. We also propose a higher order scheme of the MOC solver based on an axial polynomial expansion of the unknown within each mesh. This method allows the reduction of the meshes (and unknowns) by keeping the same precision. All the methods developed in this thesis have been implemented in the APOLLO3 version of the neutron transport solver TDT.

Key words: Neutron transport, method of characteristics, 3D, APOLLO3, high-order methods, 3D tracking strategies, synthetic acceleration, parallel methods, quadrature formulas, multi-group equivalence.

Résumé

Le sujet de ce travail de thèse est l'application de la méthode des caractéristiques longues (MOC) pour résoudre l'équation du transport des neutrons pour des géométries à trois dimensions extrudées. Les avantages du MOC sont sa précision et son adaptabilité, le point faible étant la quantité de ressources de calcul requises. Ce problème est même plus important pour des géométries à trois dimensions ou le nombre d'inconnues du problème est de l'ordre de la centaine de millions pour des calculs d'assemblage. La première partie de la recherche a été dédiée au développement de techniques optimisées pour le traçage et la reconstruction à la volée des trajectoires. Ces méthodes profitent des régularités des géométries extrudées et ont permis une forte réduction de l'empreinte mémoire et une réduction des temps de calcul du MOC. La convergence du schéma itératif a été accélérée par un opérateur de transport dégradé (DP_N) qui est utilisé pour initialiser les inconnues de l'algorithme itératif et pour la solution du problème synthétique au cours des itérations MOC. Les algorithmes pour la construction et la solution des opérateurs MOC et DP_N ont été accélérés en utilisant des méthodes de parallélisation à mémoire partagée qui sont le plus adaptés pour des machines de bureau ou pour les clusters de calcul. Une partie importante de cette recherche a été dédiée à l'implémentation de méthodes d'équilibrage de charge pour améliorer l'efficacité du parallèlisme. La convergence des formules de quadrature pour des cas 3D extrudé a aussi été explorée. Certaines formules profitent de couts négligeables du traitement des directions azimutales et de la direction verticale pour accélérer l'algorithme. La validation de l'algorithme du MOC a été faite par des comparaisons avec une solution de référence calculée par un solveur Monte Carlo avec un traitement continu de l'énergie. Pour cette comparaison on propose un couplage entre le MOC et la méthode des Sous-Groupes pour prendre en compte les effets des résonances des sections efficaces. Le calcul complet d'un assemblage de réacteur rapide avec interface fertile/fissile nécessite de $1h30$ d'exécution avec des erreurs de quelque pcm par rapport la solution de référence. On propose aussi une approximation d'ordre supérieur du MOC basée sur une expansion axiale polynomiale du flux dans chaque maille. Cette méthode permet une réduction du nombre de mailles (et d'inconnues) tout en gardant la même précision. Toutes les méthodes développées dans ce travail de thèse ont été implémentées dans la version APOLLO3 du solveur de transport TDT.

Mots-clés: Transport des neutrons, méthode des caractéristiques, 3D, APOLLO3, schémas d'ordre supérieur, traçage en 3D, accélération synthétique, méthodes parallèles, formules de quadrature, équivalence multi-groupe.

Acknowledgments

This Ph.D Thesis is the result of three years of researches done at the Laboratoire de Transport Stochastique et Déterministe (LTSD) of the Service d'Études de Réacteurs et des Mathématiques Appliquées (SERMA) of the CEA of Saclay. This work would not have been possible without the support of the people which have shared with me this experience.

First of all, I want to thank Simone Santandrea and Richard Sanchez for having given me the possibility of learning from them, for all the discussions and the exchange of ideas, for the opportunities they have opened to me, and for their continuous support. I also want to thank Piero Ravetto and Jean Ragusa for having accepted to be referees of this Thesis, and Patrick Blanc-Tranchant and Frank Gabriel, heads of SERMA and LTSD, for making this three-years experience actually possible.

My gratitude also goes to all the people of the SERMA for their kindness and for their precious help, in particular to Emiliano, Andrea, Anthime, Igor, Li, Fabien, Pietro and Remi. I want to thank Jean-François Vidal, Pascal Archier et Jean-Marc Palau of the SPRC of the CEA Cadarache for having shown great interest in this work and for their collaboration.

To my parents which have supported all my choices, and which have given me the possibility of following my ambitions.

To all my Parisian friends for their sustain and for making my staying in Paris happier, especially to Quentin and Mathilde for having been my personal French teachers, and for having shared with me 'la vie de coloc'.

Finally I am deeply thankful to Enrica for having turned on that shiny light in a future often so unclear. Thank you for having been always beside me in the most difficult moments, and thank you for having shared with me the most important ones.

Contents

List of Figures

List of Tables

Introduction

The knowledge of the physical phenomena that govern the behavior of nuclear power plants is a key step for their design and optimization. The *nuclear reactor* constitutes the component of a nuclear power plant where neutrons collide with heavy isotopes contained in the fuel and split them in lighter products through a *fission* reaction. Each fission releases about $200MeV$ of energy which is removed by the coolant that transfers it to the traditional part of the power plant, generally constituted by a steam generator, a system of turbine and electrical generator and a condenser.

Among the light products of a fission reaction there are newborn neutrons that start traveling through the nuclear reactor with a given probability to trigger new fissions, and initiate a *chain reaction*. Other fission products are generally unstable charged particles that are immediately blocked by surrounding materials because of Coulombic forces, and contribute to the chain reaction only through neutron emission.

Not the sole fission governs the evolution of the chain reaction: the wide set of neutron-induced nuclear reactions includes for example the *absorption*, which determines the neutron removal without any newborn neutron, or the *scattering*, whose sole effect is a change in the neutron direction and kinetic energy; other reactions of interest generally determine the production of more than one newborn neutron contributing to the chain reaction like a fission. The balance between the neutrons produced by the ensemble of the nuclear reactions and those removed by absorption or by leakage from the nuclear reactor determines the evolution of the chain reaction, and must be known with accurate precision to guarantee safety and operability of the nuclear reactor.

The accurate knowledge of the spatial distribution of the neutrons inside the reactor provides important information that can be exploited to improve the efficiency of the power plants. One example may be the power distribution, which constitutes the key parameter of the thermal-hydraulics design of the reactor. These quantities can be obtained by direct measure in exper-

imental or power reactors, or computed by modeling the physics governing the nuclear reactor and solving the equations with the help of numerical methods. This manuscript will consider only this second approach.

The behavior of neutrons inside a nuclear reactor is faithfully described by the Neutron Transport Equation which is based on the mechanical/statistical approach firstly introduced by Ludwig Boltzmann in 1872 in his kinetic theory of gases. The use of an accurate solution of this equation for modeling the whole reactor is still a visionary objective because of the prohibitive costs of the calculation for such complex geometries. In addition, the coupling between the *neutronics*[1] and physical phenomena of different nature (thermal-hydraulics, mechanics) requires these calculations to be realized for several working conditions of the reactor. The problem is circumvented by analyzing the core with two levels of resolution: at the *lattice level*, detailed transport calculations are used for problems of reduced size, these generally corresponding to an assembly or a cluster of assemblies. At the *core level*, assemblies are considered as homogeneous regions and diffusive-like approximations of the neutron transport equation are used to represent the neutron behavior. The physical constants of the diffusive laws are computed through equivalence between the detailed transport solution and the coarser representation at the core level. These constants are computed for different states of the assemblies (e.g., temperature, burn-up, surroundings) and constitute the *parametric libraries* that are actually used to perform core calculations.

Although the two-level approach, transport calculations are still computationally intensive and further approximations are used to lighten their costs. In particular, current industrial schemes rely on a two-dimensional approximation of the lattice geometry, whereas the third dimension is generally accounted for only at core level with the diffusion approximation. Such approach is justifiable if the composition of the geometry varies slowly with one of the directions, which is the case of common nuclear reactor designs. However, the need of more detailed analysis, as well as the need of studying new reactor designs for improving efficiency and safety, has growth the interest in three-dimensional transport calculations.

One of the most successful methods of resolution of the Neutron Transport Equation is the Method of Characteristics (MOC), firstly proposed by Askew [1], and later adopted for nuclear reactor analysis in several codes [2, 3, 4, 5, 6]. The MOC uses a trajectory-based discretization of the geometry to correctly account for transport effects. The strength of

[1] *Neutronics:* the ensemble of physical phenomena governing the interactions between neutrons and the matter.

this method is in its applicability to any geometry, whereas the amount of computational resources which it requires constitutes its major weakness.

Two/Three-Dimensional-Transport (TDT) is a solver of the neutron transport equation initially conceived for two-dimensional geometries based on the Method of Characteristics which has been firstly introduced in the code for nuclear reactor analysis APOLLO2 developed at the Laboratoire de Transport Stochastique et Déterministe in CEA Saclay [**?**]. A modernized version of the solver is provided with the APOLLO3® code which is in development in the same laboratory, and allows the treatment of three-dimensional geometries. The treatment is limited to (right)[2] cylindrical geometries, generated by the translation of a plane surface along the normal to the surface itself. Although cylindrical geometries can not describe a general three-dimensional problem, they are sufficient to model the majority of cases of practical use.

The extension of the MOC to three-dimensional geometries is limited by its computational requirements (a realistic three-dimensional calculation may require hundreds of time the resources needed for a similar two-dimensional calculation). In our research we concentrate on the development of numerical methods especially conceived for a MOC solver for three-dimensional cylindrical geometries, and implemented in the APOLLO3® version of TDT.

This manuscript is divided in three parts. In the first part we recall the basic concepts of the physics of interaction between neutrons and matter by introducing the Boltzmann equation and the concept of nuclear cross sections. In this part we will also recall the classical approximations and models used to obtain a numerical solution of the transport equation. The second part is dedicated to the new developments done in the MOC solver of TDT. Several topics will be discussed with the common objective of reducing its computational costs. The last part of this manuscript is dedicated to the actual application of the MOC algorithm to a heterogeneous assembly of an innovative reactor with the aim of verifying its correct implementation and to show its applicability to realistic problems.

[2] In broad terms, the direction of the translation in a cylindrical geometry does not need to be normal to the plane surface.

Part I

Background

Chapter 1

Neutron transport equation

The behavior of neutrons inside the reactor is faithfully described by the *Neutron transport equation* (NTE). This is an integro-differential equation obtained by imposing the balance between production and removal of neutrons inside an infinitesimal volume of the phase space defined by the neutron position $\mathbf{r}$ in the spatial domain $\mathcal{D}$, and velocity $\mathbf{v} \in \mathbb{R}^3$. In Reactor Physics it is common use to indicate the latter by the set of coordinates $\mathbf{\Omega} \in S^2$ (S^2 the unit sphere) and $E \in \mathbb{R}^+$, indicating, respectively, the direction of the neutrons and their kinetic energy. The physics of the interaction between neutrons and the surrounding materials is described by *cross sections*. This topic will be discussed in next section, while in Sec. 1.2 we will show the mathematical form of the steady-state NTE. Section 1.3 is dedicated to the common multi-group approximation, while in the last section we will provide details about the iterative scheme used to solve the NTE in the steady-state approximation.

1.1 Cross sections

The interaction between a neutron and an isotope X can be described by the reaction:

$$n + X \to P + \nu n,$$

where P includes all the reaction products except neutrons, while the symbol ν represents the number of *secondary* neutrons produced by the reaction. Among reaction products there are unstable isotopes called *precursors* of *delayed* neutrons which eventually decay by neutron emission. In steady-state calculations the concentration of precursors and the emission of delayed

neutrons are at equilibrium, and are taken into account with a modified value of secondary neutrons, ν.

1.1.1 Microscopic cross sections

The *microscopic partial* cross section of the reaction ρ between a neutron and an isotope x is a measure of the likelihood of that reaction to happen. It is measured in *barns*[1] and it will be indicated with the symbol $\sigma_{x,\rho}$. It is a nuclear property of the isotope, and it depends on the relative speed of the incident neutron with respect to the speed of target nuclei. Since these latter are generally assumed to be at rest, the dependence of the cross sections can be equivalently be written as a function of the neutron kinetic energy, E. The *total* microscopic cross section of an isotope is the sum of the partial cross sections of all possible reactions, and is indicated with $\sigma_{x,t}(E)$.

1.1.2 Resonances

The dependence of the cross section on the energy of the incident neutron is strongly influenced by the *resonance* phenomenon: when a neutron interacts with a target isotope, it may be captured to form an intermediate compound nucleus. If the total available energy in the reaction (the kinetic energy of the neutron if the target isotope is assumed at rest) exactly corresponds to the energy required to form the compound nucleus in one of its excited states, the probability of interaction becomes very large, and may change by orders of magnitude for small variations of the energy of the neutron [7]. Resonances appear only for values of the energy larger than a given threshold, which generally corresponds to the ground state of the compound nucleus. In Fig. 1.1 we show the total microscopic cross section of ${}^{238}_{92}U$. We can identify three characteristic regions of the energy domain:

Thermal domain (up to a few eV) where there are no resonances and the cross section varies regularly following the $1/v$ law, with v the relative speed of the incident neutron;

Resonance domain the first resonance of ${}^{238}_{92}U$ appears at $6,67eV$ and causes an abrupt increase of the cross section from tens of *barns* to tens of thousands of *barns*. Resonances appear more frequently for increasing values of the energy with decreasing value of the peak cross section;

[1] $1b = 1 \times 10^{-24} cm^2$

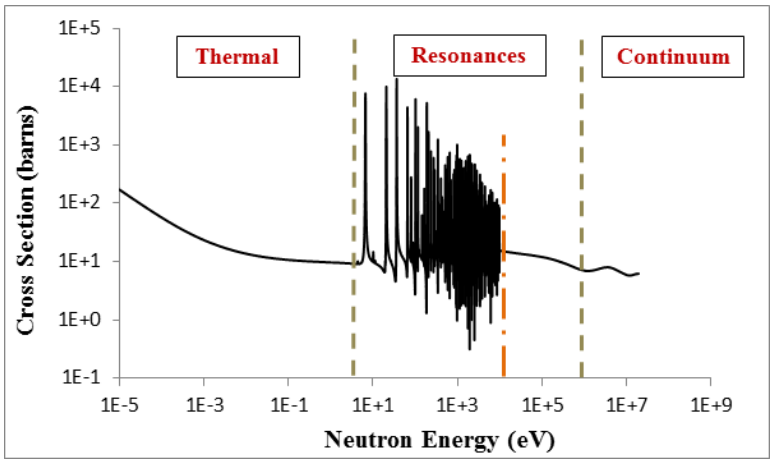


Figure 1.1: Total microscopic cross section of $^{238}_{92}U$(JEFF-3.1). The gray dashed lines determines the limits of resonances and continuum domains. The orange dashed line shows the limit between resolved and unresolved resonances.

Continuum where resonances overlap to form a continuous slowly varying function.

We further split the resonance domain into the domain of *resolved* resonances and the domain of *unresolved* ones. The limit between the two domains is determined by technological limits of experimental measures: starting from a certain energy, the spacing between two resonances becomes smaller than the maximal resolution of the measure instruments. While for the domain of resolved resonance we can represent the cross section as a function of the energy, the representation of the cross section in the domain of *unresolved* resonances can be expressed only by statistical laws [7].

1.1.3 Secondary neutron distributions

The angular and energy distributions of the neutrons produced by the reaction is also described by nuclear data. The probability for a neutron that has undergone a *scattering* reaction to be scattered from the volume $d\mathbf{\Omega}'dE'$ around $(\mathbf{\Omega}, E)'$ to volume $d\mathbf{\Omega}dE$ around $(\mathbf{\Omega}, E)$ of velocity space is indicated with $p_{x,s}(E' \to E, \mathbf{\Omega}' \cdot \mathbf{\Omega})$. Remark that the isotropy of the material allows to indicate the angular dependence of the scattering probability in terms of the cosine of the angle between the entering and exiting directions. The

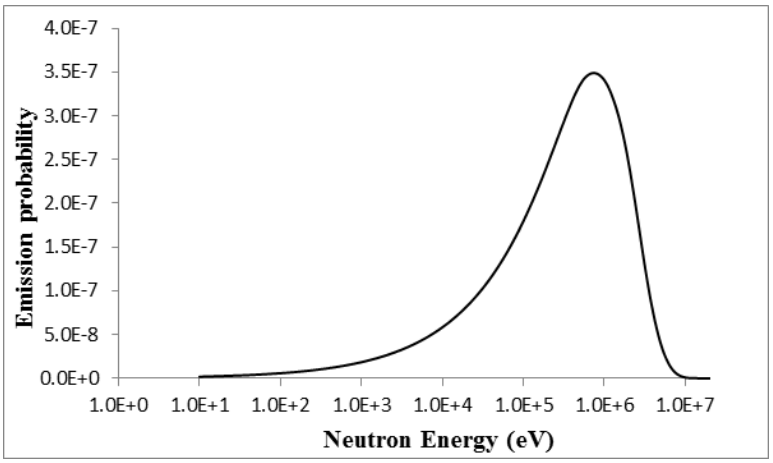


Figure 1.2: Fission neutrons emission spectrum for $^{235}_{92}U$ (JEFF 3.1).

scattering probability follows this normalization:

$$\int_{S^2} d\mathbf{\Omega} \int_{\mathbb{R}^+} dE p_{x,s}(E' \to E, \mathbf{\Omega}' \cdot \mathbf{\Omega}) = 1. \tag{1.1}$$

This relation imposes the probability of the incident neutron to be scattered in any other direction and energy to be a certain event. In a similar way we can define the energy and angular distribution of secondary neutrons for any reaction, $p_{x,\rho}$. The normalization in Eq. (1.1) is also applied to the $p_{x,\rho}$, while the multiplicity of the secondary neutrons is taken into account by the quantity $\nu_{x,\rho}$.

Concerning *fission* reaction, it is common to assume an isotropic angular distribution of the secondary neutrons, while their energy distribution is represented by the *fission spectrum*, $\chi_{x,f}(E)$, which provides the probability of a neutron to be emitted with energy E (Fig. 1.2).

Legendre expansion of the transfer probability

It is customary to represent the angular dependence of the transfer probability $p_{x,\rho}$ with an expansion on a basis of Legendre Polynomials [8] of the scattering angle, $P_k(\mathbf{\Omega}' \cdot \mathbf{\Omega})$ for $0 \leq k \leq K$. By using this representation we can write:

$$p_{x,s}(E' \to E, \mathbf{\Omega}' \cdot \mathbf{\Omega}) \approx \sum_k p_{x,k}(E' \to E) P_k(\mathbf{\Omega}' \cdot \mathbf{\Omega}). \tag{1.2}$$

The addition theorem of the Spherical Harmonics allows to recast this expression into [9]:

$$p_{x,\rho}(E' \to E, \mathbf{\Omega}' \cdot \mathbf{\Omega}) \approx \sum_n \frac{1}{2k(n)+1} p_{x,\rho}^{k(n)}(E' \to E) A_n(\mathbf{\Omega}') A_n(\mathbf{\Omega}), \quad (1.3)$$

where the symbol A_n represents the *Real Spherical Harmonic* [8] whose order $0 \leq k \leq K$ and degree $|l| \leq k$ are indicated with a single subscript $n = 1, \ldots, (K+1)^2$. Remark that in Eq. (1.3) we have used the function $k(n)$ to represent the correspondence between the numbering n of the harmonics and their degree k. In the following we will omit this complex notation and we will indicate only the symbol k. In the following developments we will use spherical harmonics normalized such that:

$$\int_{S^2} d\mathbf{\Omega} A_n(\mathbf{\Omega}) A_m(\mathbf{\Omega}) = 4\pi\delta_{mn}. \quad (1.4)$$

By using Eq. (1.3) to represent the emission probability, the transfer operator (Eq. (1.13)) can be rewritten in the following form:

$$\mathcal{H}\psi(\mathbf{r}, \mathbf{\Omega}, E) = \sum_n A_n(\mathbf{\Omega}) \int_{\mathbb{R}^+} dE' \Sigma_{s,k}(\mathbf{r}, E' \to E) \Phi_n(\mathbf{r}, E), \quad (1.5)$$

where $\Phi_n(\mathbf{r}, E)$ represents the nth *angular moment* of the flux:

$$\Phi_n(\mathbf{r}, E) = \frac{1}{4\pi} \int_{S^2} d\mathbf{\Omega} A_n(\mathbf{\Omega}) \psi(\mathbf{r}, \mathbf{\Omega}, E), \quad (1.6)$$

and the quantity $\Sigma_{s,k}$ is defined as:

$$\Sigma_{s,k}(\mathbf{r}, E' \to E) = \frac{1}{2k+1} \sum_x \sum_{\rho \neq f} n_x(\mathbf{r}) \sigma_{x,\rho}(E') p_{x,\rho}^k(E' \to E). \quad (1.7)$$

1.1.4 Macroscopic cross sections

The *macroscopic* cross section is computed by multiplying the microscopic cross section by the atomic density n_x of the target isotope and physically represents the probability of triggering the reaction per unit path traveled by the neutron. For reaction ρ we define the *macroscopic* cross section:

$$\Sigma_{x,\rho} = n_x \sigma_{x,\rho}. \quad (1.8)$$

For a heterogeneous medium with an isotopic composition $\mathcal{N}(\mathbf{r}) = \{n_x(\mathbf{r})\}_{x=1}^{N_x}$, the macroscopic cross section is the sum of the macroscopic cross sections of all N_x isotopes. In particular we define:

- macroscopic total cross section:

$$\Sigma_t(\mathbf{r}, E) = \sum_x \sum_\rho n_x(\mathbf{r})\sigma_{x,\rho}(E);$$

- macroscopic transfer cross section:

$$\Sigma_s(\mathbf{r}, E' \to E, \boldsymbol{\Omega}' \cdot \boldsymbol{\Omega}) = \sum_x \sum_{\rho \neq f} n_x(\mathbf{r})\sigma_{x,\rho}(E')\nu_{x,\rho}p_{x,\rho}(E' \to E, \boldsymbol{\Omega}' \cdot \boldsymbol{\Omega});$$

- macroscopic fission production cross section:

$$\chi\nu\Sigma_f(\mathbf{r}, E' \to E) = \sum_x \chi_{x,f}(E)n_x(\mathbf{r})\nu_f\sigma_{x,f}(E').$$

1.2 Steady-state Neutron Transport Equation

The mathematical form of the steady-state Neutron Transport Equation reads:

$$(\boldsymbol{\Omega} \cdot \nabla + \Sigma_t)\,\psi = (\mathcal{H}\psi) + \frac{1}{k_{eff}}\,(\mathcal{F}\Phi) \tag{1.9}$$

where $\psi(\mathbf{r}, \boldsymbol{\Omega}, E) = vn_{\mathrm{N}}(\mathbf{r}, \boldsymbol{\Omega}, E)$ is the neutron *angular flux*, which represents the number of neutrons, $n_{\mathrm{N}}(\mathbf{r}, \boldsymbol{\Omega}, v)$, crossing the unit surface orthogonal to direction of motion. The quantity $\Phi(\mathbf{r}, E)$ is the neutron *scalar flux*, defined as the integral of the angular flux over the set of angular directions:

$$\Phi(\mathbf{r}, E) = \int_{S^2} d\boldsymbol{\Omega}\psi(\mathbf{r}, \boldsymbol{\Omega}, E). \tag{1.10}$$

The product of the scalar flux and a macroscopic cross section is called *reaction rate*, and it identifies the total number of events triggered by the neutrons of energy E in a given position of the space $\mathbf{r}$:

$$R_{x,\rho}(\mathbf{r}, E) = \Sigma_{x,\rho}(\mathbf{r}, E)\Phi(\mathbf{r}, E). \tag{1.11}$$

In Eq. (1.9) the quantity $(\boldsymbol{\Omega}\cdot\nabla+\Sigma_t)$ is the so-called *transport* operator which describes the neutrons motion and collisions, and it will be denoted with the symbol $\mathcal{L}$. It is composed of the steady-state advection operator, $(\boldsymbol{\Omega}\cdot\nabla)$, and the total removal due to interaction of neutrons with the surrounding materials, represented by the total cross section. Another important quantity is the *angular current*, defined as:

$$\mathbf{J}(\mathbf{r}, \boldsymbol{\Omega}, E) = \boldsymbol{\Omega}\psi(\mathbf{r}, \boldsymbol{\Omega}, E). \tag{1.12}$$

The quantity $\mathcal{H}$ is the *transfer* operator which models the change of direction and energy of neutrons due to any reaction except fission. It represents the number of neutrons that are scattered in direction $\mathbf{\Omega}$ and energy E from any position of the velocity space $(\mathbf{\Omega}, E)'$:

$$\mathcal{H}\psi(\mathbf{r}, \mathbf{\Omega}, E) = \int_{S^2} d\mathbf{\Omega}' \int_{\mathbb{R}^+} dE' \Sigma_s(\mathbf{r}, E' \to E, \mathbf{\Omega}' \cdot \mathbf{\Omega}) \psi(\mathbf{r}, E', \mathbf{\Omega}'), \quad (1.13)$$

The $\mathcal{F}$ operator is the *production* operator, which takes into account the total neutrons emitted by the fission reaction:

$$\mathcal{F}\Phi(\mathbf{r}, \mathbf{\Omega}, E) = \frac{1}{4\pi} \int_{\mathbb{R}^+} dE' \chi \nu \Sigma_f(\mathbf{r}, E, E') \Phi(\mathbf{r}, E'). \quad (1.14)$$

The k_{eff} parameter which divides the fission operator in Eq. (1.9) is the eigenvalue introduced to impose the existence of a non-trivial solution of the steady-state NTE. It is called *effective multiplicative factor* and represents the ratio between the total production due to fission and the removal by absorption and leakage out of the system:

$$k_{eff} = \frac{\int_{\mathcal{D}} d\mathbf{r} \int_{S^2} d\mathbf{\Omega} \int_{\mathbb{R}^+} dE \mathcal{F}\Phi(\mathbf{r}, \mathbf{\Omega}, E)}{\int_{\mathcal{D}} d\mathbf{r} \int_{S^2} d\mathbf{\Omega} \int_{\mathbb{R}^+} dE \left(\mathcal{L} - \mathcal{H}\right) \psi(\mathbf{r}, \mathbf{\Omega}, E)} \quad (1.15)$$

A system is said to be *critical* if the neutron population remains stable, which corresponds to $k_{eff} = 1$. For $k_{eff} > 1$ the system is said to be *super-critical* and it identifies a divergent chain reaction, while for *sub-critical* systems ($k_{eff} < 1$) the chain reaction is not self-sustained[2].

1.2.1 Boundary Conditions

The NTE requires the definition of boundary conditions (BC) which are expressed in terms of the flux entering the boundary Γ of the domain $\mathcal{D}$. By denoting with $\hat{\mathbf{n}}$ the outgoing normal vector to the boundary, we write the BC as:

$$\psi(\mathbf{r}, \mathbf{\Omega}, E) = \psi_{in}, \quad \mathbf{r} \in \mathcal{D}, \mathbf{\Omega} \cdot \hat{\mathbf{n}} < 0. \quad (1.16)$$

Depending on the nature of the problem, the entering flux may assume different forms. For *vacuum* BCs, for example, we impose $\psi_{in} = 0$.

For systems characterized by geometrical symmetries such as *reflection, rotation or translation*, the solution is computed only in a portion of the domain, called *basic* domain, which represents the smallest geometrical motif

[2] A coherent definition of the k_{eff} should also take into account the excess of neutrons produced by reactions other than fission. This provides a small contribution and, for simplicity of equations, it will be not included in throughout this manuscript.

that reproduces the complete domain by repeated application of the symmetries. The basic domain is practically obtained by cutting the geometry along its symmetry axes as shown in Fig. 1.3. The boundaries generated by this geometry reduction are the so-called *closed* boundaries. The symmetries of the geometry are taken into account with appropriate BCs applied on closed boundaries. These BCs are called *geometrical* boundary conditions, and assume the form:

$$\psi(g\mathbf{r}, g\mathbf{\Omega}, E) = \psi(\mathbf{r}, \mathbf{\Omega}, E),\ \mathbf{r} \in \Gamma, \mathbf{\Omega} \cdot \hat{\mathbf{n}} > 0. \tag{1.17}$$

In this equation $g(\mathbf{r}, \mathbf{\Omega}) = (g\mathbf{r}, g\mathbf{\Omega})$ is the *geometrical motion* representing the one-to-one mapping between the flux exiting the geometry in position $(\mathbf{r}, \mathbf{\Omega})$ and the entering flux in position $(g\mathbf{r}, g\mathbf{\Omega})$. The explicit form of $\mathbf{g}$ depends on the kind of symmetry:

Reflectional with respect to the surface with outgoing normal $\hat{\mathbf{n}}$:

$$g(\mathbf{r}, \mathbf{\Omega}) = (\mathbf{r}, \mathbf{\Omega} - 2(\mathbf{\Omega} \cdot \hat{\mathbf{n}})\hat{\mathbf{n}});$$

Translational with period $\mathbf{T}$:

$$g(\mathbf{r}, \mathbf{\Omega}) = (\mathbf{r} - \mathbf{T}, \mathbf{\Omega});$$

Rotational with rotation angle θ:

$$g(\mathbf{r}, \mathbf{\Omega}) = (g(\mathbf{r}), g(\mathbf{\Omega})).$$

A special mention is due to *infinite lattice* geometries which are constituted of an infinite repetition of an elementary geometry which coincides with the basic domain (see Fig. 1.4).

1.3 Multi-group formalism

In the multi-group formalism the energy domain is partitioned into G energy groups, with each group representing the values of energy E contained in the range $\Delta E_g = [E_g, E_{g-1}]$. It is common use to count groups for decreasing values of the energy and to identify three main regions of the energy domain:

Fast groups: include groups where fission neutrons are emitted, which corresponds to values of energy within about $20MeV$ and a few keV.

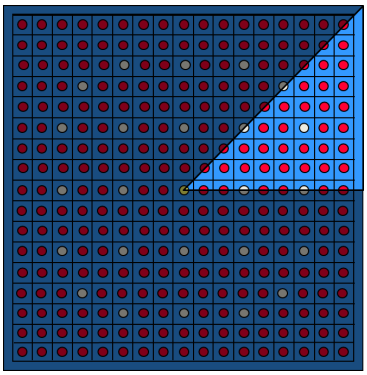


Figure 1.3: Square geometry with 1/8 symmetry. The basic domain (lightened) is an isosceles right triangle. The lower and the diagonal sides of the triangle are symmetry axes.

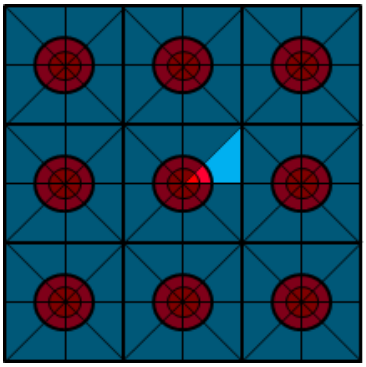


Figure 1.4: Example of infinite lattice with 1/8 symmetry. The sides of the basic domain are all symmetry axes. The infinite lattice is generated by repeated applications of the symmetries.

Epithermal groups: include groups where the fission emission is negligible but neutrons are still highly energetic with respect to the surrounding medium;

Thermal groups: include the energy groups where the energy of the neutrons is comparable with the energy of the surrounding medium.

For each energy group we define the *group flux* as the integral of the neutron flux within the energy group:

$$\psi^g(\mathbf{r}, \mathbf{\Omega}) = \int_{\Delta E_g} dE \psi^g(\mathbf{r}, \mathbf{\Omega}, E). \tag{1.18}$$

This quantity constitutes the unknown function of the multi-group problem. The equations governing the exchange of particles between energy groups are obtained by *equivalence* with respect to the continuous-in-energy NTE. In practice, this is done by defining *multi-group* cross sections that preserve the reaction rate over the energy group. In particular we define:

$$\Sigma_t^g(\mathbf{r}, \mathbf{\Omega}) = \frac{\int_{\Delta E_g} dE \Sigma_t(\mathbf{r}, E) \psi(\mathbf{r}, \mathbf{\Omega}, E)}{\psi^g(\mathbf{r}, \mathbf{\Omega})}, \tag{1.19}$$

$$\Sigma_s^{gg'}(\mathbf{r},\mathbf{\Omega},\mathbf{\Omega'}) = \frac{\int_{\Delta E_g} dE \int_{\Delta E_{g'}} dE' \Sigma_s(\mathbf{r},E' \rightarrow E, \mathbf{\Omega'}\cdot\mathbf{\Omega})\psi(\mathbf{r},\mathbf{\Omega'},E')}{\psi^{g'}(\mathbf{r},\mathbf{\Omega})}, \tag{1.20}$$

$$\chi\nu\Sigma_f^{gg'}(\mathbf{r}) = \frac{\int_{\Delta E_{g'}} dE' \int_{\Delta E_g} dE \chi\nu\Sigma_f(\mathbf{r},E,E')\Phi(\mathbf{r},E')}{\Phi^{g'}(\mathbf{r})}, \tag{1.21}$$

From these definitions one can see that formally the equivalent multi-group total and transfer cross sections also depend on the angular variable, which is undesired because it increases the complexity of the treatment of cross sections, and also increases the memory required for their storage. This angular dependence does not appear in the multi-group macroscopic fission cross section since the fission reaction rate is an isotropic quantity and is naturally weighted by the scalar flux. The same property holds for the isotropic component of the scattering if we assume the Legendre expansion discussed in Sec. 1.1.3. Also remark that the definition of the equivalent cross sections requires the knowledge of the flux inside the energy group, which is unfortunately the unknown function we ought to compute. However, the group flux in Eqs. (1.19)-(1.21) acts as a weight function, which implies that for a sufficiently fine energy mesh it can be substituted by a lower order approximation without incurring in large errors. The multi-group equivalence is usually performed using a reference scalar flux as weighting function (details about the calculation of the group flux are provided in Ch. 2). By making these assumptions the multi-group version of the NTE reads:

$$\left(\mathbf{\Omega}\cdot\nabla + \Sigma_t^g\right)\psi^g(\mathbf{r},\mathbf{\Omega}) = \sum_{g'}\int_{S^2} d\mathbf{\Omega}\Sigma_s^{gg'}(\mathbf{r},\mathbf{\Omega'}\cdot\mathbf{\Omega})\psi^{g'}(\mathbf{r},\mathbf{\Omega'}) + \\ + \frac{1}{k_{eff}}\sum_{g'}\frac{\chi\nu\Sigma_f^{gg'}(\mathbf{r})}{4\pi}\Phi^{g'}(\mathbf{r}). \tag{1.22}$$

Remark from this equation that the flux in each energy group is coupled to the fluxes of other groups through the transfer and the fission operators, while the transport operator only depends on the group considered (i.e., the multi-group transport operator is diagonal). The coupling between the equations can be studied by analyzing the profile of the transfer and fission matrices $\mathbf{\Sigma}_s = mat\{\Sigma_s^{gg'}\}$ and $\mathbf{\Sigma}_f = mat\{\chi\nu\Sigma_f^{gg'}\}$. The fission matrix constitutes a strong coupling between thermal and fast groups since the fission

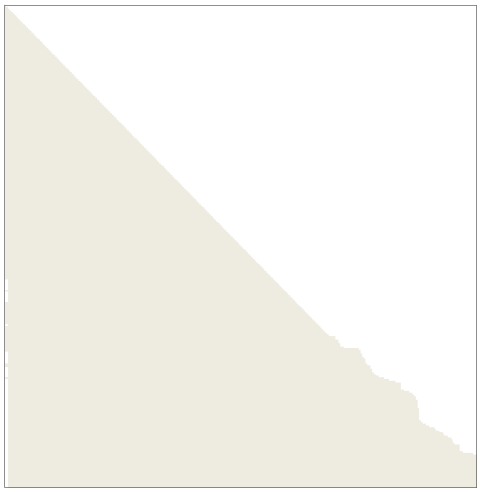


Figure 1.5: Non-zero profile of a typical transfer matrix for a 281 groups energy discretization.

neutrons are emitted in the fast groups (Fig. 1.2) while fission reactions may happen in the whole energy spectrum. Concerning the transfer matrix, reactions involving a neutron with large kinetic energy force the neutron to slow down while a gain in energy is unlikely. It follows that for fast energy groups the matrix is lower triangular. The triangular profile however is progressively lost for decreasing values of energy as it is shown in Fig. 1.5. This happens because at low energies neutrons form an equilibrium with the surrounding media, and they can as well loose or gain energy after a scattering reaction. The probability for the neutron to gain energy is usually referred to as the *upscattering* probability.

1.4 Numerical solution of the steady-state neutron transport equation

The numerical solution of the NTE is obtained using a nested iterative algorithm to solve simultaneously the criticality problem and the multi-group equations. The outermost part of the algorithm uses the *power* iteration method [10] to obtain the solution and the eigenvalue of the criticality problem. By denoting with superscript e the index of the iteration, and with $\vec{\psi}$ and $\vec{\Phi}$ the vectors of the multi-group fluxes, the iterative version of Eq. (1.22)

is:

$$(\mathcal{L}-\mathcal{H})\,\vec{\psi}^{e+1}(\mathbf{r},\mathbf{\Omega}) = \frac{1}{k^e_{eff}}\mathcal{F}\vec{\Phi}^e(\mathbf{r}). \tag{1.23}$$

The iterative procedure starts with an initial guess of $\Phi^0(\mathbf{r})$ and k^0_{eff}, and consists in repeatedly solving Eq. (1.23) until convergence is reached for both unknowns. An estimation of the k_{eff} at each iteration can be derived from equation Eq. (1.15) by substituting Eq. (1.23) in the expression for the total removals:

$$k^{e+1}_{eff} = k^e_{eff}\frac{\sum_g \int_{\mathcal{D}} d\mathbf{r}\mathcal{F}\Phi^{e+1}_g(\mathbf{r})}{\sum_g \int_{\mathcal{D}} d\mathbf{r}\mathcal{F}\Phi^{e}_g(\mathbf{r})}. \tag{1.24}$$

Each step of the iterative procedure requires the solution of the slowing down problem (Eq. (1.22)) for the fixed fission source computed at the previous iteration. The triangular profile of the transfer matrix for fast energy groups allows a direct inversion of the multi-group equations by subsequently solving mono-group problems, while a Gauss-Seidel method is used to invert the non triangular range of the matrix. The iterations for the calculation of the k_{eff} are commonly called *external* (or outer) iterations, while the iterations of the multi-group problem are generally referred to as *thermal* iterations.

The solution of the one-group problem requires to iterate over the in-group flux, which appears in the transport term and in the transfer integral. The iterative formulation of the in-group transport equation reads:

$$(\mathbf{\Omega}\cdot\nabla+\Sigma^g_t)\,\psi^{it+1}(\mathbf{r},\mathbf{\Omega}) = \int_{S^2} d\mathbf{\Omega}\Sigma^{gg}_s(\mathbf{r},\mathbf{\Omega}'\cdot\mathbf{\Omega})\psi^{it}(\mathbf{r},\mathbf{\Omega}')+S^g(\mathbf{r},\mathbf{\Omega}), \tag{1.25}$$

where it is the index of the so-called *internal* iteration (to lighten the notation the g superscript has been omitted in the definition of the in-group flux). The external source S^g includes the transfer contributions from other energy groups and the fission term at the respective thermal (index th) and external (index e) iterations:

$$\begin{aligned} S^g(\mathbf{r},\mathbf{\Omega}) = &\sum_{g'<g}\int_{S^2} d\mathbf{\Omega}\Sigma^{gg'}_s(\mathbf{r},\mathbf{\Omega}'\cdot\mathbf{\Omega})\psi^{th+1}(\mathbf{r},\mathbf{\Omega}')+ \\ &+\sum_{g'>g}\int_{S^2} d\mathbf{\Omega}\Sigma^{gg'}_s(\mathbf{r},\mathbf{\Omega}'\cdot\mathbf{\Omega})\psi^{th}(\mathbf{r},\mathbf{\Omega}')+ \\ &+\frac{1}{k^e_{eff}}\sum_{g'}\frac{\chi\nu\Sigma^{gg'}_f(\mathbf{r})}{4\pi}\Phi^e(\mathbf{r}). \end{aligned} \tag{1.26}$$

Each internal iteration requires the inversion of the mono-energetic transport operator to obtain a new guess for the in-group angular flux. In Ch.3 we will see how this is done using the Method of Characteristics.

1.4.1 The SN and PN approximations

The common representations of the angular variable used to obtain a numerical solution of the NTE are the SN and PN approximations. To better understand some properties of the two approximations, in this paragraph a matrix notation is used to indicate the angular discretization. In this context it is useful to rewrite Eq. (1.5) in terms of the vectors[3] $\vec{A}(\mathbf{\Omega}) = \{A_n(\mathbf{\Omega}), k(n) \leq K)\}$ and $\vec{\Phi}(\mathbf{r}, E) = \{\Phi_n(\mathbf{r}, E), k(n) \leq K)\}$, representing, respectively, the set of $(K+1)^2$ Spherical Harmonics and $(K+1)^2$ moments of the flux (Eq. (1.6)) required for a Kth order expansion of the transfer probability (see Sec. 1.1.3):

$$\mathcal{H}\psi = \vec{A}^{\intercal}\, \mathbf{\Sigma}_s\, \vec{\Phi}, \tag{1.27}$$

with

$$\mathbf{\Sigma}_s(\mathbf{r}, E) = diag\left(\int_{\mathbb{R}^+} dE \Sigma_{s,k}(\mathbf{r}, E' \rightarrow E)\star\right). \tag{1.28}$$

The fission operator (Eq. (1.14)) can be written in a similar fashion:

$$\mathcal{F}\Phi = \vec{A}^{\intercal}\, \mathbf{\Sigma_f}\, \vec{\Phi}, \tag{1.29}$$

where this time $\mathbf{\Sigma_f}(\mathbf{r}, E)$ is a matrix whose sole non-zero value corresponds to the first term of the diagonal.

The PN approximation is a projective method where the angular flux is represented on the basis $\vec{A}(\mathbf{\Omega})$ of the $(N+1)^2$ Spherical Harmonics up to the Nth degree:

$$\psi(\mathbf{r}, E, \mathbf{\Omega}) \approx \vec{\Phi}^{\intercal}(\mathbf{r}, E)\vec{A}(\mathbf{\Omega}). \tag{1.30}$$

By defining the projector:

$$\vec{\mathcal{P}} = \frac{1}{4\pi}\int_{S^2} d\mathbf{\Omega}\vec{A}(\mathbf{\Omega})\star, \tag{1.31}$$

the PN approximation proceeds by applying $\mathcal{P}_n$ to the NTE (Eq. (1.9)), obtaining

$$\mathbf{L}_{\mathrm{PN}}\vec{\Phi}(\mathbf{r}, E) = \mathbf{\Sigma}_s\vec{\Phi}(\mathbf{r}, E) + \frac{1}{k_{eff}}\mathbf{\Sigma}_f\vec{\Phi}(\mathbf{r}, E). \tag{1.32}$$

[3] Column vectors.

Because of the orthogonality of the Spherical Harmonics (Eq. (1.4)), this procedure diagonalizes both transfer and fission operators, whereas the operator $\mathbf{L}_{\mathrm{PN}}$ couples each moment of the flux to other six moments [11].

In the SN approximation, differently, the NTE is evaluated for a set of discrete directions $\{\mathbf{\Omega}_j\}_{j=1}^{N_\Omega}$, while a *quadrature formula* $S_N = \{(\omega_j, \mathbf{\Omega}_j)\}_{j=1}^{N_\Omega}$ is used to approximate angular integrals. In particular the moments of the flux read:

$$\vec{\Phi}(\mathbf{r}, E) = \frac{1}{4\pi} \int_{S^2} d\mathbf{\Omega} \vec{A}(\mathbf{\Omega}) \psi(\mathbf{r}, \mathbf{\Omega}, E) \approx \sum_{j=1}^{N_\Omega} \omega_j \vec{A}(\mathbf{\Omega}_j) \psi(\mathbf{r}, \mathbf{\Omega}_j, E) \equiv \mathbf{A}\mathbf{W}\vec{\psi}, \tag{1.33}$$

where $\vec{\psi} = \{\psi(\mathbf{\Omega}_j)\}_{j=1}^{N_\Omega}$ is the angular flux evaluated at the discrete directions, $\mathbf{W}$ is the $N_\Omega \times N_\Omega$ diagonal matrix containing the weights of the quadrature formula, and $\mathbf{A}$ is the $M \times N_\Omega$ matrix whose columns are the spherical harmonics evaluated at the discrete directions. Here we have indicated with M the total number of moments of the flux considered for the expansion. A system of equations similar to Eq. (1.32) can be written for the SN method:

$$\mathbf{L}_{\mathrm{SN}} \vec{\psi}(\mathbf{r}, E) = \mathbf{A}^\intercal \mathbf{\Sigma}_s \mathbf{A}\mathbf{W}\vec{\psi}(\mathbf{r}, E) + \frac{1}{k_{eff}} \mathbf{A}^\intercal \mathbf{\Sigma}_f \mathbf{A}\mathbf{W}\vec{\psi}(\mathbf{r}, E). \tag{1.34}$$

Differently from the PN approximation, the transport operator in the SN approximation is diagonalized using the basis associated to the N_Ω directions of the quadrature formula:

$$\mathbf{L}_{\mathrm{SN}} = diag(\mathbf{\Omega}_j \cdot \nabla + \Sigma_t). \tag{1.35}$$

Instead, both transfer and fission operators couple all the values of the angular flux through the angular quadrature formula and the Spherical Harmonics. Because of the diagonality of the transport operator, the SN approximation is often used for the inversion of the mono-group problem in Eq. (1.25). However, it is interesting to note that the SN approximation suffers of some problems, the most known is the *ray-effect* which appears in highly absorbing media with localized sources, and is due to the fact that the SN approximation constraints the streaming of neutrons in fixed directions [12]. In the literature we can find several attempts to mitigate the ray-effect deriving SN-like equations [13, 14], and it is still an actual topic. The ray-effect may introduce errors in cases such as shielding problems, but it mildly concerns nuclear reactor physics because sources are well distributed [15].

Chapter 2

Multi-group cross sections

This chapter is dedicated to the generation of multi-group equivalent cross sections, a fundamental step for correctly take into account the effects of selfshielding which derive from the presence of resonances in the cross sections. In particular, in 2.2 we describe the Subgroups method [16], which has shown to be a robust method for computing selfshielded cross sections. This method relies on *Probability tables* to approximate the coupling between the energy and spatial variables. For completeness, in Sec. 2.1 we recall the basic concepts of Probability tables and how these are computed in the CALENDF code [17].

2.1 Probability tables

The probability tables are used to approximate energy averages of the form:

$$\langle f \rangle_g = \frac{1}{\Delta E_g} \int_{\Delta E_g} dE f\left[\sigma_t(E)\right], \tag{2.1}$$

where f is a function which depends on the energy only through of the microscopic total cross section σ_t.

By indicating with $\mathcal{S}_g$ the set of values of the total cross section in the group g, $\mathcal{S}_g = \{\sigma_t(E), E \in \Delta E_g\}$, the Riemann integral over the energy in Eq. (2.1) is recast in terms of Lebesgue integral over $\mathcal{S}_g$:

$$\begin{aligned}\langle f \rangle_g &= \frac{1}{\Delta E_g} \int_{\Delta E_g} dE f\left[\sigma_t(E)\right] \int_{\mathcal{S}_g} d\sigma_t \delta(\sigma_t - \sigma_t(E)) = \\ &= \int_{\mathcal{S}_g} d\sigma_t f(\sigma_t) \int_{\Delta E_g} \frac{dE \delta(\sigma_t - \sigma_t(E))}{\Delta E_g} \equiv \int_{\mathcal{S}_g} d\sigma_t f(\sigma_t) p(\sigma_t), \end{aligned} \tag{2.2}$$

where δ is the Dirac function and the quantity $p(\sigma_t)d\sigma_t$ is a probability density function which provides a measure of the likelihood of a given cross section to be found in the energy range ΔE_g.

A *probability table* is a set of weights and abscissas $\mathcal{Q}_K = \{p_k, \sigma_{t,k}\}_k^K$ which approximate the Lebesgue integral in Eq. (2.2) with a finite sum:

$$\int_{\mathcal{S}_g} d\sigma_t f(\sigma_t) p(\sigma_t) \approx \sum_k p_k f(\sigma_{t,k}). \tag{2.3}$$

In the classical approach [18], the domain $\mathcal{S}_g$ is partitioned in K subdomains $\Delta\mathcal{S}_k$ and the following definition for the weights is used:

$$p_k = \frac{1}{\Delta E_g} \int_{\Delta E} dE \Theta_k \left[\sigma_t(E)\right], \tag{2.4}$$

with Θ_k the indication function of the total cross section within the kth subdomain:

$$\Theta_k \left[\sigma_t(E)\right] = \begin{cases} 1, & \sigma_t(E) \in \Delta\mathcal{S}_k \\ 0, & \text{elsewhere} \end{cases} \tag{2.5}$$

A comparison with Eq. (2.2) shows that the weights in Eq. (2.4) can be obtained by assuming a piecewise approximation of $f(\sigma_t)$ within the energy group. The abscissas in this case are generally chosen using a mid-point rule, or by using the average abscissa within the kth subdomain:

$$\sigma_{t,k} = \frac{1}{\Delta E_g} \int_{\Delta E} dE \Theta_k \left[\sigma_t(E)\right] \sigma_t(E). \tag{2.6}$$

A more general approach has been introduced by Pierre Ribon [18], and it is the one used in the CALENDF code [17]. In this approach Gaussian quadratures are used to approximate the energy integrals. These are obtained by imposing the set of weights and abscissas to exactly integrate a set of $2K$ polynomials of the total cross section, which entails:

$$\mathcal{M}_h = \frac{1}{\omega_g} \int_{\Delta E_g} dE \sigma_t^h(E) = \sum_k \sigma_{t,k}^h \, p_k, \tag{2.7}$$

where h is the degree of the polynomial such that $1 - K \leq h \leq K$ (see [18] for details about the choice of the degree of polynomials), and $k = 1, \ldots, K$ indicates the kth couple of weight and abscissa of the probability table. The quantity $\mathcal{M}_h$ is the hth moment of the total cross section, and is computed using the point-wise representation provided by nuclear data libraries [19]. The order K of the probability table is chosen by imposing a convergence

criteria on the calculation of higher order moments of the total cross section. In other words, for a probability table of order K, the moments of order $K+1$ and $-K$ are computed using the point-wise representation of the total cross section, and are compared with the approximated values computed with the probability table. If the error is larger than a given tolerance, the order of the probability table is increased to $K+1$, and so on until convergence.

The concept of probability tables can be extended in order to approximate averages of functions which depend on the energy through multiple cross sections. This is the case, for example, of the calculation of reaction rates for partial reactions, or rather the case of a mixture of isotopes. Let us limit the analysis to the calculation of reactions rates while we refer the reader to [18, 17, 7] for an in-depth discussion about the generation of probability tables. The average reaction rate over an energy group has the form:

$$\langle \sigma_\rho f \rangle_g = \frac{1}{\Delta E_g} \int_{\Delta E_g} dE\, \sigma_\rho(E) f\left[\sigma_t(E)\right] = \\ = \int_{\mathcal{S}_g} d\sigma_t f(\sigma_t) \int_{\mathcal{S}_{g,\rho}} d\sigma_\rho \sigma_\rho p(\sigma_\rho, \sigma_t), \quad (2.8)$$

where the definition of $p(\sigma_\rho, \sigma_t) d\sigma_\rho d\sigma_t$ naturally follows from the one in Eq. (2.2):

$$p(\sigma_\rho, \sigma_t) = \frac{1}{\Delta E_g} \int_{\Delta E_g} dE \delta(\sigma_t - \sigma_t(E)) \delta(\sigma_\rho - \sigma_\rho(E)), \quad (2.9)$$

and it represents the density of probability for having simultaneously $\sigma_t = \sigma_t(E)$ and $\sigma_\rho = \sigma_\rho(E)$. The integral in Eq. (2.8) can be recast into:

$$\langle \sigma_\rho f \rangle_g = \int_{\mathcal{S}_g} d\sigma_t f(\sigma_t) p(\sigma_t) \int_{\mathcal{S}_{g,\rho}} d\sigma_\rho \sigma_\rho p_\rho(\sigma_\rho, \sigma_t). \quad (2.10)$$

where we have used the quantity $p_\rho(\sigma_\rho, \sigma_t) = p(\sigma_\rho, \sigma_t)/p(\sigma_t)$, representing the conditional probability of σ_ρ for a given σ_t. By using the $\mathcal{Q}_K$ quadrature formula to approximate Eq. (2.10) we write:

$$\langle \sigma_\rho f \rangle_g \approx \sum_k p_k f(\sigma_{t,k}) \sigma_{\rho,k}, \quad (2.11)$$

with $\sigma_{\rho,k}$ the partial cross section averaged with respect to the conditional probability with respect to $\sigma_{t,k}$:

$$\sigma_{\rho,k} = \int_{\mathcal{S}_{g,\rho}} d\sigma_\rho \sigma_\rho p_\rho(\sigma_\rho, \sigma_{t,k}). \quad (2.12)$$

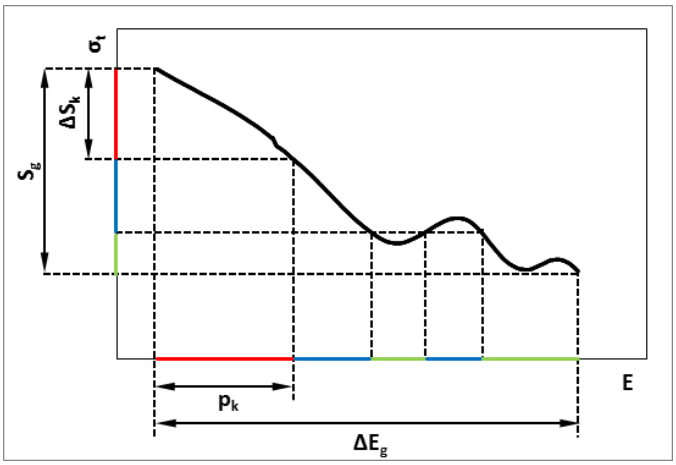


Figure 2.1: Graphical representation of Probability Tables. The function represents the total cross section σ_t as a function of the energy E. The symbol $\mathcal{S}_g$ represents the image of the energy group ΔE_g through the total cross section $\sigma_t(E)$. In the standard approach (Eq. (2.4)), $\mathcal{S}_g$ is partitioned in sub-domains ΔS_k (different colors), with associated probability p_k corresponding to the inverse image of ΔS_k. The CALENDF approach uses Gaussian formulas to exactly integrate polynomials of the total cross section.

In this form, the problem reduces to seek for the K values of $\sigma_{\rho,k}$ which better approximate the reaction rate integral. The CALENDF approach is to impose the exact integration of K co-moments of the total and partial cross sections:

$$\mathcal{P}_h = \frac{1}{\Delta E_g}\int_{\Delta E_g} dE\sigma_\rho(E)\sigma_t^h(E) = \sum_k \sigma_{\rho,k}\sigma_{t,k}^h\, p_k, \quad \text{for } -\frac{K}{2} \leq h \leq \frac{K}{2} - 1. \tag{2.13}$$

The quadrature points for the partial cross sections are found for all reactions of interest for neutronic calculations. A probability table of order K of a given isotope takes the form of a the set of K n-tuple containing the weights and the nodes for the calculation of energy averages of all reactions:

$$\mathcal{PT}_K = \{\{\sigma_{\rho,k}\}_\rho, p_k, \sigma_{t,k}\}_{k=1}^K. \tag{2.14}$$

2.2 Subgroups method

The multi-group approximation used for the solution of the transport equation (see Sec. 1.3) requires a correct definition of multi-group cross sections. An exact equivalence can be obtained by imposing the conservation of the macroscopic reaction rates for each energy group, as described by Eqs. (1.19)-(1.21). However, this approach requires the knowledge of the neutron flux inside the energy group, which is the unknown function that we want to compute. The problem is circumvented by substituting the real flux with an approximated solution, and by using a sufficiently refined energy mesh. The author in [7] analyzes the discretization error introduced by the multi-group equivalence for several approximations of the transport equation. Here, we briefly discuss the Subgroup method [20] which is one of the methods used in APOLLO3® for the calculation of multi-group cross sections.

In the Subgroup method the multi-group cross section of a given reaction ρ and isotope x for group g is computed imposing the following equivalence relation:

$$\sigma^g_{\rho,x}(\mathbf{r}) \approx \frac{< \sigma_{\rho,x}(E)\Phi(\mathbf{r},E) >_g}{< \Phi(\mathbf{r},E) >_g}, \tag{2.15}$$

where the scalar flux, Φ, is the solution of a slowing-down problem under some simplification hypotheses, and the brackets indicate the average operation over the energy group g:

$$< f(E) >_g = \frac{1}{\Delta E_g} \int_{\Delta E_g} dE f(E). \tag{2.16}$$

The first approximation used by the Subgroup consists in treating the fine structure of the cross sections only for one isotope at time, while group-averaged cross sections are used for other isotopes. The isotopes are processed iteratively to update the values of the multi-group cross sections until a consistent set is obtained. This approach introduces errors on the estimation of the multi-group cross sections whenever resonances of different isotopes overlap [21]. However, for sufficiently fine energy meshes the resonance overlapping is reduced and the error introduced is small [7]. An additional approximation is applied on the treatment of the transfer probabilities and the emission spectra. In the Subgroup method, these are assumed to be constant within each energy group, while the detailed treatment of the energy variable is reserved to the sole cross sections. This assumption is acceptable for sufficiently refined energy meshes, since the major contribution to the in-group emission is due to the transfer from other groups. Under

these hypotheses, and by assuming an isotropic law for the scattering, the slowing-down equation of group g for the unknown in-group flux ψ is written as follows:

$$[\mathbf{\Omega}\cdot\nabla+\Sigma_g(\mathbf{r},E)]\,\psi(\mathbf{r},\mathbf{\Omega},E)=N_y(\mathbf{r})p_{0,y}^{gg}<\sigma_{s,y}(E)\Phi(\mathbf{r},E)>_g\Delta E_g+ \\ +\sum_{x\neq y}N_x(\mathbf{r})p_{0,x}^{gg}\sigma_{s,x}^g(\mathbf{r})<\Phi(\mathbf{r},E)>_g\Delta E_g+Q^g(\mathbf{r}). \quad (2.17)$$

In the last equation we used Eq. (2.16) to rewrite the reaction rates in terms of group-averaged values. The total cross section, $\Sigma_g(\mathbf{r},E)$, takes into account the isotope which is being considered (subscript y) and *background* isotopes:

$$\Sigma_g(\mathbf{r},E)=N_y(\mathbf{r})\sigma_{t,y}(E)+\sum_{x\neq y}N_x(\mathbf{r})\sigma_{t,x}^g(\mathbf{r}); \quad (2.18)$$

while the quantity Q^g includes the contributions of the transfer from other energy groups and the fission source:

$$Q^g(\mathbf{r})=\sum_{\substack{g'\neq g\\ x}}p_{0,x}^{gg'}N_x(\mathbf{r})\sigma_x^{g'}(\mathbf{r})\Phi^{g'}(\mathbf{r})+\frac{1}{k_{eff}}\sum_{g',x}\chi_x^g\nu N_x(\mathbf{r})\sigma_{f,x}^{g'}(\mathbf{r})\Phi^{g'}(\mathbf{r}). \quad (2.19)$$

Remark that in the definition of Q^g we have assumed the multi-group cross sections for other energy groups and for the fission reaction to be known, which allows rewriting the reaction rates in terms of multi-group fluxes, defined as:

$$\Phi^g(\mathbf{r})=\int_{\Delta E_g}dE\Phi(\mathbf{r},E). \quad (2.20)$$

The values of the transfer probability, $p_{0,x}^{gg'}$, and of the fission emission spectrum, χ_x^g, are those provided by nuclear data libraries [19].

By denoting by $\mathcal{K}$ the inverse transport operator integrated over the angular variable:

$$\mathcal{K}(\mathbf{r},E)=\int_{S^2}d\mathbf{\Omega}\,[\mathbf{\Omega}\cdot\nabla+\Sigma_g(\mathbf{r},E)]^{-1}, \quad (2.21)$$

and by q^g the total emission density:

$$q^g(\mathbf{r})=N_y(\mathbf{r})p_{0,y}^{gg}<\sigma_{s,y}(E)\Phi(\mathbf{r},E)>_g\Delta E_g+ \\ +\sum_{x\neq y}N_x(\mathbf{r})p_{0,x}^{gg}\sigma_{s,x}^g(\mathbf{r})<\Phi(\mathbf{r},E)>_g\Delta E_g+Q^g(\mathbf{r}), \quad (2.22)$$

the in-group scalar flux used for the equivalence is written as:

$$\Phi(\mathbf{r},E)=\mathcal{K}(\mathbf{r},E)q^g(\mathbf{r}). \quad (2.23)$$

2.2.1 Use of probability tables in the Subgroup method

In the previous section we have seen that, under some simplifications, the scalar flux inside the gth energy group can be written as the product of the continuous-in-energy inverse transport operator $\mathcal{K}$ and a constant emission density q^g. From the definition of the total cross section (Eq. (2.18)) we can see that $\mathcal{K}$ depends on the energy only through the microscopic total cross section of the isotope which is being treated, $\sigma_{t,y}$. Such property allows the use of Probability Tables (see Sec. 2.1) to compute the averaged values of the flux and of the reaction rates. By using Eq. (2.3) and Eq. (2.11)[1] together with Eq. (2.23) we can write:

$$< \Phi(\mathbf{r}, E) >_g = \sum_k p_k \mathcal{K}(\mathbf{r}, \sigma_{t,k}) q^g(\mathbf{r}), \tag{2.24}$$

$$< \sigma_{\rho,y}(E)\Phi(\mathbf{r}, E) >_g = \sum_k \sigma_{\rho,k} p_k \mathcal{K}(\mathbf{r}, \sigma_{t,k}) q^g(\mathbf{r}). \tag{2.25}$$

By substituting these expression in Eq. (2.22) we obtain:

$$\begin{aligned} q^g(\mathbf{r}) = N_y(\mathbf{r}) p_{0,y}^{gg} \Delta E_g \sum_k \sigma_{\rho,k} p_k \mathcal{K}(\mathbf{r}, \sigma_{t,k}) q^g(\mathbf{r}) + \\ + \sum_{x \neq y} N_x(\mathbf{r}) p_{0,x}^{gg} \sigma_{s,x}^g(\mathbf{r}) \Delta E_g \sum_k p_k \mathcal{K}(\mathbf{r}, \sigma_{t,k}) q^g(\mathbf{r}) + Q^g(\mathbf{r}). \end{aligned} \tag{2.26}$$

Provided the values of the external source, Q^g, and an expression for the inverse transport operator, Eq. (2.26) can be solved for the unknown total emission density q^g. Finally, this quantity is used to compute the multi-group cross sections of the isotope which is being treated:

$$\sigma_{\rho,x}^g(\mathbf{r}) \approx \frac{\sum_k \sigma_{\rho,k} p_k \mathcal{K}(\mathbf{r}, \sigma_{t,k}) q^g(\mathbf{r})}{\sum_k p_k \mathcal{K}(\mathbf{r}, \sigma_{t,k}) q^g(\mathbf{r})}, \tag{2.27}$$

where again Eq. (2.3) and Eq. (2.11) have been used in Eq. (2.15) to approximate the energy averages. Remark that this definition is a non-linear function of the external source, which in turn depends on the values of the background selfshielded isotopes. This non-linearity requires an iterative solution to converge on both selfshielded cross sections and the external source. Remark also that, although the continuous-in-energy microscopic cross section is a function of the sole isotope, the multi-group equivalence introduces a spatial dependence of the cross sections which has to be correctly taken into account in actual calculations.

[1] For simplicity of notation the symbol $\sim$ is omitted.

Chapter 3

The Method of Characteristics for three-dimensional axial geometries

The Method of Characteristics (MOC) is one of the most successful methods for solving the Neutron Transport Equation. It makes use of the solution of the NTE along characteristic lines to compute the angular and spatial variation of the angular flux within the problem geometry.

The MOC has been firstly proposed by Askew [1], and later adopted for nuclear reactor analysis in several codes [2, 3, 4, 5, 6]. Although MOC can be applied to arbitrary geometries, it requires a large amount of computational resources to solve the transport equation along each characteristic line, a fact that has historically limited its application to two-dimensional problems. Nonetheless, the growing interest for accurate transport solutions for improved reactor safety and fuel-cycle, as well as the increasing computational power of modern architectures, have raised the interest in three-dimensional transport calculations based on MOC. The literature about this topic is vast and includes several solutions which differ from each other mainly for the representation basis used for the angular flux, and for the type of three-dimensional geometries to which these methods apply.

A first class of methods are the so-called *fusion* methods which are applied to treat the special case of *axial geometries*[1]. Fusion methods solve

[1]Recall that for axial geometries we intend those systems generated by the finite trans-

a system of coupled two-dimensional problems representing different axial nodes of the geometry. In these methods, the two-dimensional solution is often obtained through MOC calculations, whereas different approximations are proposed for the axial coupling. The Korean code DeCART firstly used a low-order Diffusion (CMFD) approximation for coupling the axial nodes [22] and then moved to a higher-order approximation (SP_N) to improve its accuracy [23]. In the French code MICADO the MOC is used either for solving the two-dimensional problems and the one-dimensional problems coupling the axial nodes [24].

Another class of methods are those based on the Short Characteristics (SC) formulation. In these methods the problem geometry is divided in sub-domains, and a surface representation is introduced for the angular flux crossing the sub-domain boundaries. In the SC formulation, the characteristic lines are used to compute the coefficients coupling the surface fluxes and the volume sources inside each sub-domain. The advantage of this approach is that the coefficients are computed only once and for a relatively small set of unknowns, making this method computationally attractive. However, for highly heterogeneous problems, a good spatial convergence is reached only for a highly refined surface representation of the angular flux. The French code IDT of APOLLO3® uses the SC approach for three-dimensional geometries composed of heterogeneous prismatic sub-domains [25, 26].

In the last class of methods for the solution of three-dimensional problems with MOC we include those based on the Long Characteristics (LC) approach. In this approach, the angular flux is represented along a finite number of characteristics lines crossing the whole problem geometry and no further approximation is done on the shape of the angular flux. A first example of application of LC MOC to 3D geometries is found in the code MCCG3D developed in 1998 by I. Suslov [27]. In this code, the geometry description is limited to *axial* geometries. Differently, the MCI module of the code DRAGON has been developed by Wu and Roy in 2003 [28] for solving the NTE with MOC in arbitrary three-dimensional geometries. The price to pay for such a general geometry description is the increased memory and time requirements of the solver. A similar approach is also used in the code MPACT [29]. The approach used in the MMOC solver developed by Liu and Wu is based on a discretization of the geometry in cubic heterogeneous pixels. Modular tracking strategies are used to generate global trajectories from the trajectories tracked in each pixel [30]. The code MOCK-3D applies LC MOC to Cartesian meshes with homogeneous

lation of a plane surface along an axis perpendicular to the surface itself

regions [31].

The methods developed in our research have been used to extend the functionalities of the APOLLO3® version of TDT. This is a numerical code based on the LC Method of Characteristics (MOC) and/or Collision Probability (CP) designed to solve the mono-group transport equation for 2D dimensional unstructured meshes. The new methods have been developed to allow TDT to solve the transport equation for three-dimensional axial geometries using LC MOC. In the following section we will introduce the MOC as a numerical method to solve the mono-group transport equation, while Sec. 3.2 is dedicated to the geometrical description of the problems we want to solve.

3.1 Method of Characteristics

In Sec. 1.4 we have seen that the multigroup problem is solved using an algorithm with nested iterations. The innermost level of iteration requires the solution of a fixed source mono-group transport problem:

$$\begin{cases} [\mathbf{\Omega} \cdot \nabla + \Sigma(\mathbf{r})]\, \psi(\mathbf{r}, \mathbf{\Omega}) = q(\mathbf{r}, \mathbf{\Omega}), & \mathbf{r} \in \mathcal{D}, \mathbf{\Omega} \in S_2 \\ \psi(\mathbf{r}, \mathbf{\Omega}) = \psi_{in} & \mathbf{r} \in \Gamma, \mathbf{\Omega} \cdot \hat{\mathbf{n}} < 0 \end{cases} \tag{3.1}$$

where $\hat{\mathbf{n}}$ is the outgoing normal to the boundary (Γ) of the geometrical domain ($\mathcal{D}$), and q is the fixed source representing the in-group self-scattering, the transfer from other groups, and the fission source (see Eq. (1.25)).

The Method of Characteristics is a method for solving first order partial differential equations and is applied to solve Eq. (3.1). The method uses parametric curves called *characteristics* to rewrite the partial differential equation as an ordinary differential equation. In the case of the neutron transport equation these characteristics are straight lines with direction parallel to $\mathbf{\Omega}$, with $\mathbf{\Omega}$ any angle in S_2. The parametric form of the characteristic can be written as follow:

$$\mathbf{t}(t) = \mathbf{t}_b + t\mathbf{\Omega},\ t \in [0, L], \tag{3.2}$$

where $\mathbf{t}_b$ indicates a generic position on the boundary with outgoing normal $\hat{\mathbf{n}}$ for which $\mathbf{\Omega} \cdot \hat{\mathbf{n}} < 0$. This indicates the entering point of the trajectory in the domain, while $\mathbf{t}(L)$ indicates its exiting point. The ordinary form of Eq. (3.1) reads:

$$\begin{cases} \dfrac{d}{dt}\psi(\mathbf{t}) + \Sigma(\mathbf{t})\psi(\mathbf{t}) = q(\mathbf{t}), \\ \psi(\mathbf{t}_b) = \psi_{in} \end{cases} \tag{3.3}$$

and its solution can be written in the following closed form:

$$\psi(\mathbf{t}) = \psi(\mathbf{t}_b)T(\mathbf{t}_b, \mathbf{t}) + \int_0^t dt' q(\mathbf{t}')T(\mathbf{t}', \mathbf{t}). \tag{3.4}$$

In this equation for simplicity of notation we have used the symbol $\mathbf{t}'$ instead of $\mathbf{t}(t')$ to represent the coordinate on the trajectory with parametric dependence on the integration variable t'. The symbol $\psi(\mathbf{t}_b)$ indicates the value of the *entering flux* provided by boundary conditions, while the symbol $T(\mathbf{t}, \mathbf{t}')$ indicates the *transmission* coefficient:

$$T(\mathbf{t}', \mathbf{t}) = \exp\left[-\tau(\mathbf{t}, \mathbf{t}')\right]. \tag{3.5}$$

Here we have used the symbol τ to indicate the *optical length* between the points $\mathbf{t}'$ and $\mathbf{t}$ on the trajectory:

$$\tau(\mathbf{t}', \mathbf{t}) = \int_{t'}^t dt'' \Sigma(\mathbf{t}''). \tag{3.6}$$

The transmission coefficient indicates the probability of a neutron emitted in position $\mathbf{t}$ to reach point $\mathbf{t}'$ without interacting with the surrounding medium, and it decreases for increasing values of the optical length.

The solution of the fixed source transport problem with the Method of Characteristics is obtained by evaluating Eq. (3.4) for all the characteristics lines spanning the phase space.

Source representation

In the Method of Characteristics the flux in the source term is represented on a support which differs from that of trajectories. For what concerns the angular variable, the Legendre expansion of the scattering term (see Sec. 1.1.3) naturally leads to a Spherical Harmonic expansion of the flux. The calculation of the flux moments is done by applying the SN approximation: the angular flux is evaluated for discrete directions, and a quadrature formula is used to update the flux moments. The solution with MOC therefore requires the evaluation of the angular flux along characteristic lines parallel to the directions contained in the SN quadrature formula.

The spatial dependence of the flux moments is generally represented by partitioning the geometry in non-overlapping regions $\mathcal{D}_r$ such that $\mathcal{D} = \cup_r \mathcal{D}_r$, and in surfaces Γ_α delimiting these regions (see fig. 3.1(a)). A local basis is then used to represent the flux moments as function of region and surface values. The common *step* approximation, for example, assumes constant

values of the flux moments inside each region. The region-averaged flux is computed by integrating the angular flux along the characteristic lines that cross the region. This topic will be discussed in detail in Chapter 4.

Trajectory-based spatial discretization

A numerical version of the MOC requires a suitable representation of the angular flux which allows the solution of the transmission equation along the characteristics lines. Different approaches exist: in the *short* characteristics approach the angular flux is represented on the surfaces delimiting macro-regions, and the transmission equation is finely integrated to compute coefficients which couple the volume and surface values. This approximation may be affected by numerical diffusion problems for optically-thin macro-regions. The *long* characteristics approach conversely partitions the geometry in parallel *bands* that cover the system from boundary to boundary – this is the approach used by TDT and the only considered throughout this manuscript. In his work [32] Fevotte explores different representations of the angular flux within each band, pointing out the advantages and disadvantages of each approximation. In the present work we will use a *rectangular* integration rule for the spatial variable: the geometry is divided in 'tubes' whose axes are the characteristic lines. The angular flux is assumed to evolve as Eq. (3.4) along the tube axis, while it is assumed to be constant on its cross sectional area Δ_t (see fig. 3.1(b)). Under this assumption, the spatial integral over one region $\mathcal{D}_r$ is approximated with the following expression:

$$\int_{\mathcal{D}_r} d\mathbf{r} w(\mathbf{r})\psi(\mathbf{r},\mathbf{\Omega}) \approx \sum_{\substack{t\|\mathbf{\Omega} \\ t\cap\mathcal{D}_r}} \Delta_t \int_{t_{in}}^{t_{out}} dt w(t)\psi(t), \tag{3.7}$$

where $t \cap \mathcal{D}_r$ is the set of trajectories that cross the region, and w is a weight function. Such method makes a zeroth order approximation of the region boundaries and therefore is not appropriate for surface integrals [33]. However, it has the advantage of precisely estimating integrals over the component of the surface perpendicular to the trajectory:

$$\int_{\Gamma_\alpha} d\mathbf{r}(\mathbf{\Omega}\cdot\hat{\mathbf{n}}) w(\mathbf{r})\psi(\mathbf{r},\mathbf{\Omega}) \approx \sum_{\substack{t\|\mathbf{\Omega} \\ t\cap\Gamma_\alpha}} \Delta_t \frac{(\mathbf{\Omega}\cdot\hat{\mathbf{n}})}{|\mathbf{\Omega}\cdot\hat{\mathbf{n}}|} w(t_{\alpha_t})\psi(t_{\alpha_t}), \tag{3.8}$$

where $t \cap \Gamma_\alpha$ is the set of trajectories crossing the surface and t_{α_t} is the point at which trajectory t intersects the surface. Such property is desirable

for the correct integration of the boundary angular currents as we will see in Chapter 4. This approach has also the advantage that the integration weight Δ_t only depends on the trajectory, while for more accurate methods it requires to be dependent on the single intersection [32, 33].

It is important to remark that the trajectory-based discretization of the geometry introduces a parametric dependence of spatial integrals with respect to the tracking angle. This dependence is found, for example, in the calculation of the region volumes. We talk about *angular volumes* when we refer to the numerical volumes computed using the trajectories with a given direction:

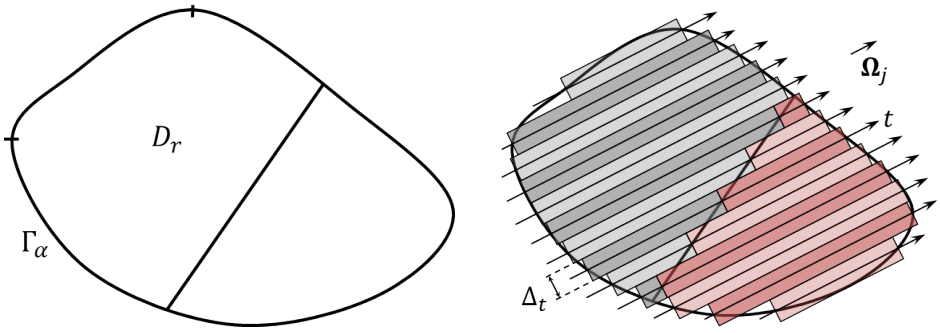
$$V_r(\mathbf{\Omega}) = \sum_{\substack{t \| \mathbf{\Omega} \\ t \cap \mathcal{D}_r}} \Delta_t l_t. \tag{3.9}$$

In this equation we have indicated with l_t the length of the intersection of the trajectory t with region $\mathcal{D}_r$. The dependence on the angle here is introduced by the fact that a finite number of trajectories is used to estimate the spatial integral. Clearly, this dependence disappears for sufficiently small trajectory spacing. Angle-independent estimates of spatial integrals can be obtained by averaging over the angular variable.

3.2 Geometrical description of the problem

A detailed description of the reactor geometry and of the material composition is required to obtain an accurate solution of the neutron transport equation. However, reactor designs are often characterized by complex heterogeneous geometries whose description is rather complicated (see Fig. 3.2). The methods that have been developed in our research are intended to solve the transport equation in what we call axial geometries. These are three-dimensional right cylinders generated by the finite movement of an arbitrary plane surface (2D section) along an axis perpendicular to the surface itself (z-axis) . We simplify the analysis by assuming a partitioning of the geometry with Cartesian axial meshes. These are meshes generated by the Cartesian product of an unstructured two-dimensional mesh, $\mathcal{D}_{2D} = \{\mathcal{D}_{r_{2D}}\}$, used for the 2D section, and a mono-dimensional mesh, $\vec{H} = \{\Delta h_{r_Z}\}$, for the z-axis. Under these assumptions, each three-dimensional region is univocally identified by the Cartesian coordinates (r_{2D}, r_Z).

Although these geometries are not general, they allow the treatment of the majority of cases encountered in reactor physics (e.g., reactor assembly) without the burden deriving from the generalization to arbitrary geometries.



(a) Partitioning in surfaces and regions. (b) Trajectory-based discretization.

Figure 3.1: Graphical representation of the different supports used for the spatial representation of the unknown flux. Fig. (a) shows an arbitrary partitioning of the geometry in sets of regions and surfaces used to represent the flux moments. Fig. (b) shows the trajectory-based discretization used by the Method of Characteristics. The space is partitioned in 'tubes' whose axes are the characteristic lines parallel to $\mathbf{\Omega}_j$. The angular flux along the characteristic lines follows the transmission equation, while an uniform value is assumed on the cross sectional area of each tube. The discretization is applied for all the directions in the SN quadrature formula.

Material composition

The variation of isotopic concentration during the reactor life caused by the transmutation of nuclei (e.g., depletion of ${}^{235}_{92}U$ and increase of fission products due to fission reactions), or the 'artificial' variation of the cross sections introduced by the multigroup equivalence (see 2.27) are some of the effects that may induce a spatial variation of the cross sections. The common approximation applied in neutronics calculation is to partition the problem geometry into homogeneous regions, that is regions with constant value of the cross section. The partitioning must be chosen in order to correctly take into account the effects described above.

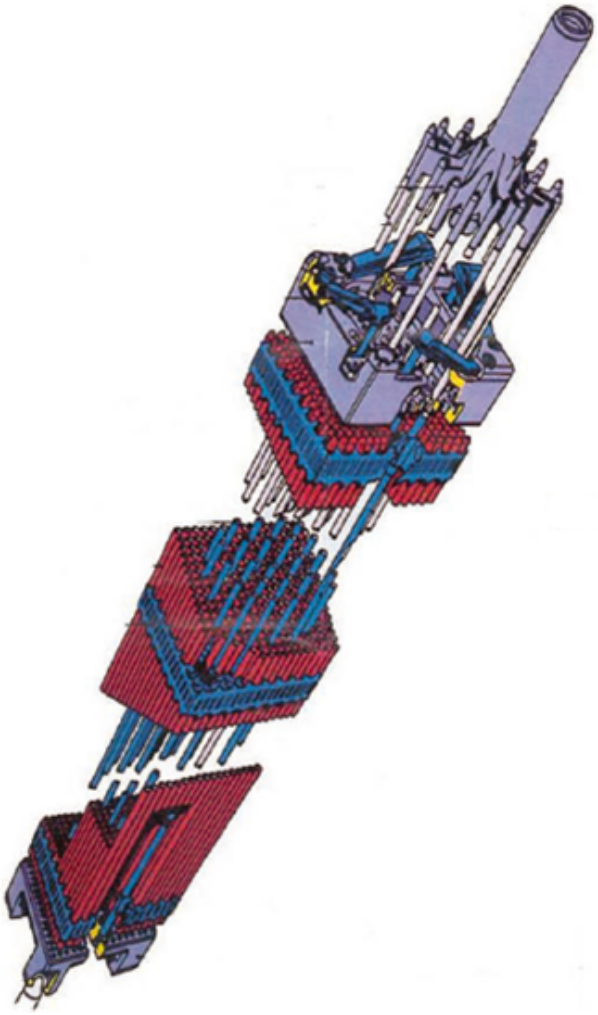


Figure 3.2: Assembly of a Pressurized Water Reactor.

Part II

New developments

Chapter 4

Higher order approximations for MOC

In Chapter 3 we have introduced the Method of Characteristics as a method for the solution of the neutron transport equation. We have seen that this method uses the transmission equation to compute the angular flux along straight characteristic lines crossing the problem geometry. We have also underlined that a different support is used to represent the spatial and angular dependence of the unknown flux appearing in the source term. The projection of the 'trajectory' flux onto the 'source' flux is done by using the SN quadrature formula for the angular integration, and the 'tube' discretization of the geometry to approximate spatial integrals. In this chapter we discuss different spatial representations of the moment of the flux and we then derive the equations for a polynomial expansion of the flux moments along the axial direction. The actual implementation of the higher-order MOC algorithm is detailed in the last section of this chapter.

The classical approximation used in the Method of Characteristics is the so-called *step* approximation, which assumes a partitioning of the geometry into homogeneous regions, and a constant-per-region spatial representation of the flux moments. These latter are obtained by integrating the angular flux over the trajectories that cross each region and by computing region-averaged values. The step approximation makes a zeroth order approximation of the space and, albeit it is a convergent scheme, it shows slow convergence rates. This obliges the use of refined meshes which translates into increased computational costs of the calculations. Higher order methods are used to overcome this problem by assuming a more detailed spatial representation of the unknown fluxes, such as a polynomial volume expan-

sion. By using higher order methods the number of meshes can be reduced without impacting the accuracy of the solution. The extent of such reduction depends on how faithfully the chosen representation describes the real solution. As a drawback, the implementation of higher order schemes always constitutes an overhead if the number of meshes is kept constant. This is due to the increased number of unknowns per mesh, and to the increased computational complexity of the equations to solve. The efficiency of a higher order method in terms of computational performances is always a trade-off between the increased complexity of the algorithms and the reduced number of unknowns required for the solution to converge.

In the literature we can find several works dedicated to the derivation of higher order schemes for the Method of Characteristics, most of them applied to two-dimensional geometries. In [3, 34] the authors use a Cartesian linear polynomial basis to represent the spatial variation of the volume fluxes in two-dimensional unstructured meshes. The spatial components of the flux are computed by projecting the flux along the characteristic onto the linear basis used for the flux expansion. A similar approach is also used in [35] where the authors extend the method to polynomials up to the fourth order. These works show that a linear source expansion performs better than the step approximation, whereas the improved precision of higher order expansions is not sufficient to justify the increased complexity of the algorithms. What we observe in general is that volume expansions perform efficiently for regions with small heterogeneities (e.g., reflector), while they provide small advantages in highly heterogeneous regions (e.g., PWR assembly). Differently from polynomial volume expansions, the Linear Surface Characteristics scheme represents the spatial variation of the fluxes by linearly interpolating between surface-averaged values. This method has shown better convergence rates as compared to the step-MOC approximation in cases such as PWR assemblies [33]. An extension of the LS to 3D cases is also possible, but it has not been developed since for the cases treated in our research it does not provide important advantages. Concerning the higher order methods for three-dimensional geometries, in [36] the authors describe an arbitrarily high-order scheme which only applies to tetrahedral geometries, while a linear volume expansion of the source is used in [37] to solve three-dimensional problems in axial geometries with arbitrary unstructured 2D section.

In the following sections we derive the equations for a higher order approximation of the Method of Characteristics based on a generic polynomial expansion of the volume flux along the axial direction. The choice of this approximation is based on the fact that for most of nuclear applications

the material composition is less heterogeneous along the axial direction so that polynomial expansions can effectively catch the large spatial gradients and allow a coarser discretization of the axial mesh. We will also propose a different method for computing the spatial components of the flux: instead of directly projecting the transmission equation onto the spatial basis, we impose balance-like equations over each region and solve them by using the net currents computed with the characteristics. Due to the timing of our research, the actual implementation of these methods have not been completed, and the actual calculations shown throughout this manuscript will be provided for the step-MOC approximation only.

4.1 Definition of the polynomial basis for the flux

In the derivation of our higher order scheme, we assume the geometry to be partitioned in homogeneous regions $\mathcal{D}_r$, and the moments of the flux in each region to be represented on the local polynomial basis

$$\vec{P}(\zeta_r) = \{\zeta_r^p, 0 \leq p \leq N_p\}, \tag{4.1}$$

where $\zeta_r = z - \bar{z}_r$, $\zeta_r \in [-\Delta h_r/2, \Delta h_r/2]$ is the local coordinate defined with respect to the center of the axial mesh, $\bar{z}_r$, while N_p is the maximum order chosen for the polynomial expansion. Under this assumption, the source can be written as:

$$q(\mathbf{r}, \mathbf{\Omega}) = \sum_r \Theta_r(\mathbf{r}) \vec{P}(\zeta_r) \cdot \vec{q}_r(\mathbf{\Omega}), \tag{4.2}$$

with Θ_r the characteristic function of the region r. The symbol $\vec{q}_r(\mathbf{\Omega})$ represents the spatial components of the angular source for direction $\mathbf{\Omega}$, and includes the contributions of the in-group flux moments, $\vec{\Phi}_{rn}$, and of the external source $\vec{S}_{rn}$:

$$\vec{q}_r(\mathbf{\Omega}) = \sum_n A_n(\mathbf{\Omega}) \left[\Sigma_{rn} \vec{\Phi}_{rn} + \vec{S}_{rn} \right] \tag{4.3}$$

with Σ_{rn} the macroscopic transfer cross section of order n. The explicit expression for $\vec{\Phi}_{rn}$ reads:

$$\vec{\Phi}_{rn} = \frac{\mathbf{P}^{-1}}{4\pi V_r} \int_{S^2} d\mathbf{\Omega} A_n(\mathbf{\Omega}) \int_{\mathcal{D}_r} d\mathbf{r} \vec{P}(\zeta_r) \psi(\mathbf{r}, \mathbf{\Omega}), \tag{4.4}$$

where V_r is the region volume, and $\mathbf{P}$ is the matrix defined as:

$$\mathbf{P} = \frac{1}{V_r} \int_{\mathcal{D}_r} d\mathbf{r} \vec{P}(\zeta_r) \otimes \vec{P}(\zeta_r). \tag{4.5}$$

The $\mathbf{P}$ matrix is symmetric with elements:

$$\mathbf{P}_{pp'} = \begin{cases} \frac{(\Delta h/2)^{p+p'}}{p+p'+1} & \text{for } p+p' \text{ even} \\ 0 & \text{for } p+p' \text{ odd} \end{cases} \tag{4.6}$$

This expression is true only in case of exact spatial integration. However, the trajectory-based discretization of the geometry introduces spatial integration errors, as well as a parametric dependence of the spatial integrals with respect to the angular variable. To take into account these effects, the matrix $\mathbf{P}$ is computed by integrating Eq. (4.5) along the characteristic lines crossing the region $\mathcal{D}_r$. The same approximation is applied to compute the angular volumes. This produces angle-dependent quantities which we will indicate with symbols $\mathbf{P}_j$ and V_{rj}.

4.2 Transmission equation

We derive now the expression for the transmission equation under the hypothesis of polynomial expansion of the source described above. We consider one single intersection (hereafter *chord*) of the trajectory with a homogeneous region, and assume the entering flux, ψ_{in}, to be known. This latter corresponds to the flux obtained by the boundary conditions for the first chord of the trajectory, or to the exiting flux of the previous chord.

Let us consider the chord i with direction $\mathbf{\Omega}_j$ entering the homogeneous region from the point with global axial coordinate z_i^{in}. By denoting with μ_j the cosine of the polar component of $\mathbf{\Omega}_j$, defined with respect to the z-axis, and by $t \in [0, l_i]$ the local coordinate along the trajectory, the coordinate of the polynomial basis, ζ_r, is written as:

$$\zeta_r = z_i^{in} + \mu_j t - \bar{z}_r. \tag{4.7}$$

By substituting this last expression in Eq. (4.2) we can write the external source as a function of the local coordinate of the trajectory:

$$q(t) = \vec{L}^{\intercal}(t)\mathbf{T}\vec{q}_{rj}, \tag{4.8}$$

where $\vec{q}_{rj}$ is the angular emission density (Eq. (4.3)) in direction $\mathbf{\Omega}_j$, and $\vec{L}$ is the basis of the powers of the trajectory coordinate:

$$\vec{L}(t) = \{t^k, 0 \leq k \leq N_p\}. \tag{4.9}$$

The quantity $\mathbf{T}$ is a $(N_p+1)\times(N_p+1)$ matrix providing the coordinate transformation from the basis $\vec{P}(\zeta)$ to $\vec{L}(t)$, and has elements $\mathbf{T}_{kp}$ equal to:

$$\mathbf{T}_{kp} = \binom{p}{k}\mu_j^k(z_i^{in}-\bar{z}_r)^{p-k}. \tag{4.10}$$

The $\mathbf{T}$ matrix is characteristic of each intersection, since it depends on the entering point, length, and direction of the chord. By substituting Eq. (4.8) into the transmission equation (Eq. (3.4)) we obtain:

$$\psi(t) = \psi_{in}\exp(-\tau) + \vec{E}^{\intercal}(t,\tau)\mathbf{T}\frac{\vec{q}_{rj}}{\Sigma_r}, \tag{4.11}$$

where $\tau=\Sigma_r t$ is the optical length and $\vec{E}$ are the *escape* coefficients, which we write in the following form:

$$\vec{E}(t,\tau) = \{t^k F_k(\tau), 0\le k\le N_p\}, \text{with} \tag{4.12}$$

$$F_k(\tau) = \frac{1}{\tau^k}\int_0^\tau d\tau'(\tau')^k\exp(\tau'-\tau). \tag{4.13}$$

The evaluation of Eq. (4.11) for $t=l_i$ provides the value of the exiting flux which is used as entering flux for the next chord. The computation of the escape coefficients and their properties will be detailed in Sec. 4.4.1.

4.3 Angular region balance

The calculation of the spatial components of the flux is done by applying the SN approximation (see Sec. 1.4.1) to obtain a discretized version of Eq. (4.4):

$$\vec{\Phi}_{rn} \approx \sum_j \omega_j \mathbf{P}_j^{-1} A_n(\mathbf{\Omega}_j)\vec{\Psi}_{rj}, \tag{4.14}$$

where we have defined with the symbol $\vec{\Psi}_{rj}$ the projections of the angular flux onto the basis defined by $\vec{P}$:

$$\vec{\Psi}_{rj} = \frac{1}{V_{rj}}\int_{\mathcal{D}_r} d\mathbf{r}\vec{P}(\zeta_r)\psi(\mathbf{r},\mathbf{\Omega}_j). \tag{4.15}$$

This quantity is usually computed by directly using Eq. (4.11) as expression for the flux along the trajectories crossing the region [37, 35]. Here we use an different approach which consists in directly projecting Eq. (3.1) onto the basis defined by $\vec{P}$. For region $\mathcal{D}_r$ with boundaries Γ_r we obtain:

$$\Delta\vec{J}_{rj} - \frac{1}{V_{rj}}\int_{\mathcal{D}_r} d\mathbf{r}\psi(\mathbf{r},\mathbf{\Omega}_j)\mathbf{\Omega}_j\cdot\hat{\mathbf{e}}_z\frac{\partial}{\partial z}\vec{P} + \Sigma_r\vec{\Psi}_{rj} = \mathbf{P}_j\vec{q}_{rj}, \tag{4.16}$$

where we have used the divergence theorem in order to write the angular balance in terms of components of the net currents:

$$\Delta\vec{J}_{rj} = \frac{1}{V_r}\int_{\Gamma_r} d\mathbf{r}\mathbf{\Omega}_j \cdot \hat{\mathbf{n}}\vec{P}_j\psi(\mathbf{r},\mathbf{\Omega}_j). \tag{4.17}$$

Remark now that the following identity holds for each polynomial P_m of the $\vec{P}$ basis:

$$\frac{\partial P_p}{\partial z} = pP_{p-1}. \tag{4.18}$$

This property allows recasting the integral in Eq. (4.16) as a function of the lower-order components of the angular flux:

$$\mathbf{C}_{rj}\vec{\Psi}_{rj} = \mathbf{P}_j\vec{q}_{rj} - \Delta\vec{J}_{rj}, \tag{4.19}$$

where the matrix $\mathbf{C}$ is defined as:

$$\mathbf{C}_{rj}\vec{P} : diag(\Sigma_r)\vec{P} - \mathbf{\Omega}_j \cdot \hat{\mathbf{e}}_z\frac{\partial}{\partial z}\vec{P}. \tag{4.20}$$

The $\mathbf{C}$ matrix is lower triangular, thus its inversion becomes straightforward by starting solving for the zeroth spatial moment, and subsequently solving for higher order modes up to the N_pth order. The inversion of Eq. (4.19) requires the knowledge of the moments of the net currents represented by Eq. (4.17). This quantity is computed using the values of the angular flux obtained by the trajectory discretization. By applying Eq. (3.8) we obtain:

$$\Delta\vec{J}_{rj} \approx \frac{1}{V_{rj}}\sum_{\substack{t\parallel\mathbf{\Omega}\\ t\cap\mathcal{D}_r}}\Delta_t\left\{\vec{P}[\zeta_r(t_t^+)]\psi(t_t^+) - \vec{P}[\zeta_r(t_t^-)]\psi(t_t^-)\right\}. \tag{4.21}$$

In last equation we have rearranged the contributions for exiting and entering surfaces in terms of the difference between the values at the exiting (t_t^+) and entering (t_t^-) points of all the trajectories crossing the region (denoted with $t\cap\mathcal{D}_r$). The notation $\zeta_r(t_t^{\pm})$ is used to indicate the value of the coordinate of the polynomial basis associated to these points.

4.4 High-order MOC algorithm

The pseudo-code in Algorithm 1 shows the basic steps of the MOC iteration. The iterative problem in Eq. (1.25) is solved by representing the unknown flux on the set of the spatial and angular moments of the flux, Φ_{nrp}. The dimension of the vector containing the flux moments is the product of the number of angular moments $(K+1)^2$ (see Sec. 1.1.3), the number of meshes N_r, and the number of axial polynomial components N_p.

Source calculation

At each iteration the source is computed with equation Eq. (4.3) using the flux moments at the previous iteration, the set of cross sections (Σ_r and Σ_{rn}) for each region and anisotropy order, and the fixed external source of each region, anisotropy and order of the polynomial expansion (S_{nrp}).

Trajectory sweep

Subsequently, the transmission equation is solved for the trajectories spanning the geometry for all the angles in the SN formula. At the beginning of each trajectory the flux is computed using the value provided by the boundary conditions. Then, Eq. (4.11) is solved for each chord composing the trajectory. For each chord, the net current for all the spatial components (subscript p) is computed and accumulated into ΔJ_{rjp} using Eq. (4.21). The dimensions of ΔJ_{rjp} are those of the total number of regions, times the order of the polynomial expansion, times the angles in the quadrature formula.

The solution of the transmission equation for all the trajectories is traditionally called trajectory sweep. The computational costs of the trajectory sweep linearly depend on the total number of chords, N_χ[1], and grow with the order of the polynomial expansion due to the operations required to solve the transmission equation, and to the number of chord-dependent coefficients that have to be computed. These include the chord length (l_i), region number (r_i), axial coordinate (z_i^{in}), and escape coefficients ($\vec{E}_i$). Some of these can be easily computed on-the-fly (e.,g., z_i^{in}), while others, like the chord length, have to be pre-computed and stored, increasing memory costs. We denote by symbol $\mathcal{T}$ the *tracking data* containing the set of coefficients for all the N_χ intersections.

[1]Here and throughout this manuscript the symbol N_χ indicates the number of crossed regions.

Angular balance and flux calculation

Once all the trajectories are swept Eq. (4.19) is solved to obtain the spatial components of the angular flux. A new estimation of Φ_{nrp} is computed using Eq. (4.14) and compared with the previous value to check for convergence.

4.4.1 Computation of the escape coefficients

The escape coefficients can be directly computed by numerically integrating Eq. (4.13). However, this is a costly numerical operation and is unwanted, especially if this has to be done on-the-fly for each chord. Alternatively these coefficients can be pre-computed and stored, but this strongly impacts the memory requirements since we have to store $N_p \times N_\chi$ elements. The factorization shown in Eq. (4.12) allows computing the escape coefficients by taking the product of the powers of the chord length and a factor F_k which depends only on the optical length τ (see Fig. 4.1). A good compromise is to pre-compute F_k factors to generate tables which are linearly interpolated on-the-fly. This still requires the storage of N_p tables, each one containing N_{int} interpolation points.

The problem can be mitigated by remarking that the following recursive relation holds for the F_k coefficients, obtained by integration by parts of Eq. (4.13):

$$F_k(\tau) = 1 - \frac{k}{\tau} F_{k-1}(\tau), \quad \text{for } k > 0, \quad F_0(\tau) = 1 - \exp(-\tau). \tag{4.22}$$

The direct application of this relation, however, is affected by numerical instabilities due to the division by τ, which in voided region and for vanishing chords can be very small ($\sim 10^{-12}$). A more stable approach is to start from the highest order (equal to N_p) and use the reciprocity relation backwards. In this way only one table for the highest order is computed while the lower order coefficients are obtained on-the-fly through the backward recursive relation:

$$F_k(\tau) = \frac{\tau}{k+1} \left[1 - F_{k+1}(\tau)\right]. \tag{4.23}$$

The absolute error between the forward and backward calculation of the F_k coefficients is shown in Fig. 4.3 for small values of τ up to order $k = 4$. Remark that the forward approach may produce completely wrong values of the F_k coefficients.

Input: Σ_r,Σ_{rn},S_{nrp},$\mathcal{T}$,S_N
Output: ϕ_{nrp}
// Initialization
$\phi_{nrp} \leftarrow \phi^{(0)}_{nrp}$;
while ϕ_{nrp} *NOT converged* **do**
  // One iteration
  // Angular emission density for all j directions
  $q_{jrp} \leftarrow$ // Eq. (4.3) ;
  for $\mathbf{\Omega_j} \in S_N$ **do** // Trajectory sweep
    for *all trajectories* $\parallel \mathbf{\Omega_j}$ **do**
      // Boundary flux
      $\psi_- \leftarrow \psi_{in}$;
      for *chords in trajectory* **do**
        // Coordinate transformation coefficients
        $\mathbf{T}_i \leftarrow \mathbf{T}(z_i^{in}, \mathbf{\Omega}_j)$ // Eq. (4.10);
        // Escape coefficients
        $E_i \leftarrow E(l_i, \tau_i)$ // Eq. (4.12);
        // Exiting flux
        $\psi_+ \leftarrow \psi(\psi_-, \mathbf{T}_i, E_i)$ // Eq. (4.11);
        // Net currents
        Cumulate ΔJ_{jrp} // Eq. (4.21);
        // Entering flux for next chord
        $\psi_- \leftarrow \psi_+$;
      end
    end
  end
  // Region-averaged angular fluxes
  $\Psi_{jrp} \leftarrow$ // Eq. (4.19);
  // Update moments
  $\phi_{nrp} \leftarrow$ // Eq. (4.14);
end

Algorithm 1: Pseudo-code describing the MOC algorithm for the solution of the mono-group transport equation. The unknown fluxes for all the regions (r), anisotropy order (n) and order of the polynomial expansion (p), are computed with an iterative scheme. At each iteration the angular emission density is computed with the last available estimation of the flux moments. Then the fixed source problem is solved for all the trajectories crossing the geometry for the directions in the S_N quadrature formula. At each intersection the exiting flux is computed using the coefficients $\mathbf{T}$ and E computed with the tracking data ($\mathcal{T}$). The net current of each chord is then accumulated into ΔJ. Once all trajectories are *swept*, the higher-order region balance is solved to obtain the spatial components of the angular flux, which are finally used to compute the angular moments of the flux.

Computational costs

A total number of $4 \times N_p + 2$ operations is required to compute the escape coefficients using the strategy just discussed. Two operations are required for the linear interpolation of the coefficient F_k of highest order, while two operations are required to solve Eq. (4.23) for each lower-order term (down to 0). Finally, $2 \times N_p$ operations are needed to compute the N_p powers of the chord length, and for the construction of the escape coefficients (Eq. (4.12)).

4.4.2 Generation of the interpolation table

The generation of the interpolation table for the F_k coefficient of highest order can be done using the forward recursive relation for large values of τ, for which no numerical instabilities occur. For small τ these coefficients can be computed using the following relation:

$$F_k = \sum_{i=1} \frac{(-1)^{i-1} k!}{(k+i)!} \tau^i. \tag{4.24}$$

This is obtained by expressing F_k with the recursive relation in Eq. (4.22) and by assuming a polynomial expansion of the exponential function around 0. In fig. 4.2 we show the approximating functions for the F_4 coefficient for varying degree of the approximated polynomial.

4.5 Convergence acceleration methods

The sole use of the Method of Characteristics for the iterative solution of the criticality and of the multi-group problems may show low convergence rates and excessive execution times. Accelerations techniques are methods used to speed up the convergence of the iterative solution. These techniques are generally based on the use of a lower order transport operator to compute a corrective factor which is applied to the unconverged solution to obtain a better estimation of the actual solution. The Group Rebalancing method, for example, is one of the acceleration techniques used in the MCI module of DRAGON [28]. It makes use of a global balance over the whole geometry to compute group-dependent corrective factors satisfying the zero-dimensional multi-group problem. This method helps reducing the number of outer iterations but it does not take into account local spatial effects. Instead, the Self Collision Rebalancing method (SCR), also used in MCI, solves the multi-group problem for each spatial region by neglecting the contribution of the entering currents, but including that of the volume (isotropic) sources [38].

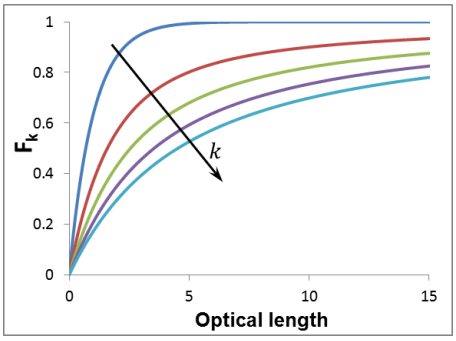


Figure 4.1: First 5 F_k coefficients as function of the optical length (τ).

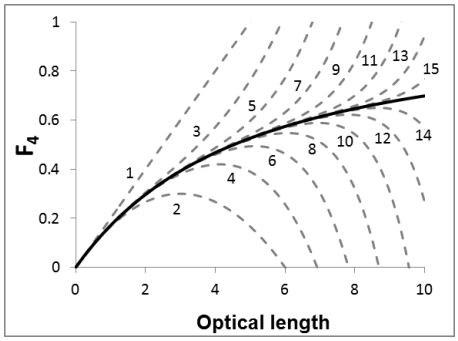


Figure 4.2: First 15 approximating polynomials of the F_4 function.

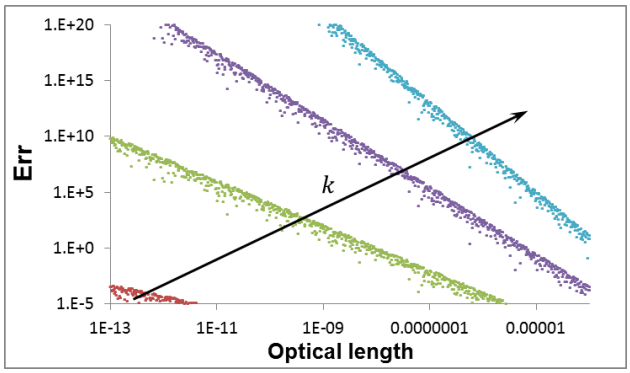


Figure 4.3: Absolute error between the backward and forward calculation of the F_k for small τ. The oscillations are due to numerical errors.

The Algebraic Collapsing Acceleration method has been firstly proposed by Khalil in 1988 [39] for mono-dimensional transport, and then applied to three-dimensional MOC first by Suslov [40] in MCCG3D, and then by Le Tellier and Hébert in MCI [41]. In this acceleration technique, the coupling between the currents exiting and entering each region is approximated through an average transmission coefficient computed by summing up the contributions of all the chords crossing that region. Another common acceleration technique is the CMFD method, which uses the fine transport solution to impose the equivalence with a diffusion operator on coarse regular spatial meshes (Cartesian in general). The diffusive problem is solved to obtain multiplicative correction factors for the fine transport solution [42].

The acceleration method proposed for TDT is an extension to 3D axial geometries of the $\mathrm{DP_N}$ synthetic acceleration method, which has been already successfully applied to 2D geometries [43]. In this approximation, a piecewise constant surface basis is used to represent the spatial variation of the angular flux, while a double Harmonic expansion (for the directions entering and exiting one surface) is assumed for the angular variable. This lower-order approximation is applied to solve the synthetic problem to obtain the corrective factor to apply to the unconverged solution. In the next section we will derive the synthetic problem, whereas in the following section we will recall the $\mathrm{DP_N}$ equations.

4.5.1 Synthetic Problem

The synthetic problem is obtained by rewriting the iterative version of the transport equation (Eq. (1.25) and Eq. (1.26)) for the error with respect to its converged solution. The synthetic problem can be derived for the three nested algorithms used for the iterative solution of the criticality (external), slowing-down (thermal), and self-scattering (internal) problems. We will write the synthetic problem for the *it* internal iteration, although it can be easily rewritten for thermal and external iterations. By indicating with *it* the index of the iteration, the general form of the of the iterative problem reads:

$$\mathcal{L}\Phi^{(it+1/2)} = \mathcal{H}\Phi^{(it+1/2)} + S, \tag{4.25}$$

where Φ represents the group-flux, $\mathcal{L}$ is the transport operator, and $\mathcal{H}$ the self-scattering operator. The transfer and fission sources are represented by S. By denoting with $\Phi^{(\infty)}$ the converged solution of Eq. (1.25), the synthetic problem is set by subtracting Eq. (1.25) to its converged form:

$$(\mathcal{L} - \mathcal{H})\delta\Phi = \mathcal{H}\Delta\Phi, \tag{4.26}$$

where $\delta\Phi = \Phi^{(\infty)} - \Phi^{(it)}$ is the unknown function of the synthetic problem, and $\Delta\Phi = \Phi^{(it+1/2)} - \Phi^{(it)}$ is the iteration error, acting as non-homogeneous source of the synthetic problem. By solving Eq. (4.26) we obtain an estimation of the group-flux for the next iteration:

$$\Phi^{(it+1)} \equiv \Phi^{(\infty)} = \Phi^{(it+1/2)} + \delta\Phi. \tag{4.27}$$

Remark that the complexity of the solution of the synthetic problem is equivalent to that of the original iterative problem. It is therefore not advantageous to solve it with the MOC approximation. Instead, a lower order operator can be used. This strategy accelerates the iterative solution since the solution of the lower order operator has smaller computational costs and it helps converging faster on lower order modes.

4.5.2 $\mathrm{DP_N}$ approximation

The diagonal transport operator $\mathcal{L}$ in Eq. (4.26) is inverted with a lower order transport-consistent $\mathrm{DP_N}$ approximation [43]. In such approximation the cell boundaries are decomposed into surfaces, α, having a spatially constant representation of surface fluxes. The angular variable is approximated with a double spherical harmonics expansion, for the directions entering and exiting each surface. The volume flux is represented with step functions and with the usual spherical harmonics expansion. By invoking the transmission equation and the cell balance for the unknown surface fluxes, $\Phi^{\nu}_{\alpha^{\pm}}$, and volume fluxes, Φ^{ν}_{r}, we obtain a system of coupled equations [43]:

$$\begin{cases} \sum\limits_{\mu} A^{\nu\mu}_{\alpha^+}\Phi^{\mu}_{\alpha^+} = \sum\limits_{\beta,\mu} T^{\nu\mu}_{\alpha\beta}\Phi^{\mu}_{\beta^-} + V_r \sum\limits_{\mu} E^{\nu\mu}_{\alpha r} q^{\mu}_{r} \\ \sum\limits_{\mu,\alpha} A^{\nu\mu}_{\alpha^+}\left(\Phi^{\mu}_{\alpha^+} - s^{\nu}s^{\mu}\Phi^{\mu}_{\alpha^-}\right) + \Sigma_{t,r}V_r\Phi^{\mu}_{r} = V_r \sum\limits_{\mu} B^{\nu\mu}_{r} q^{\mu}_{r} \end{cases} \tag{4.28}$$

where s^{μ} and s^{ν} take into account the parity of the spherical harmonics, and the external source is computed using the iteration error as in Eq. (4.26). The definition of the coefficients A, B, T and E can be found in [44].

To impose consistency of the $\mathrm{DP_N}$ approximation, the $\mathrm{DP_N}$ coefficients for all the energy groups are computed using the tracking $\mathcal{T}$ used for the transport sweep.

In our implementation, the lower order $\mathrm{DP_N}$ operator is also used to initialize the fluxes for the actual multi-group transport calculation. This strategy allows a faster convergence on diffusive modes and leaves to the MOC solver the task to converge on higher order modes.

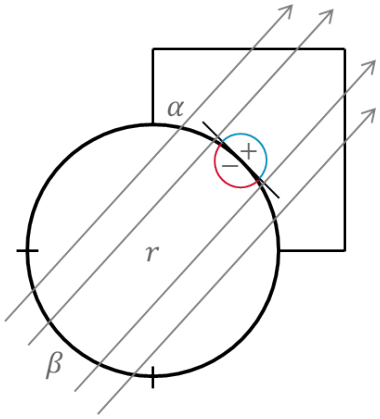


Figure 4.4: Sketch of the approximations done in the DP_N approximation. The angular flux is represented on constant spatial basis of regions, r, and surfaces, α/β. The coupling coefficients between surface and volume fluxes are computed using the MOC trajectories.

4.5.3 Results of the DP_N acceleration

The DP_N synthetic acceleration discussed above has been tested on a simple problem constituted by a hexagonal cell whose 2D section is shown in Fig. 4.5, and whose axial composition is heterogeneous. The criticality problem has been solved using three strategies. In case `A` no acceleration is used and the convergence is obtained only using free MOC iterations. In case `B` the DP1 acceleration is used for both internal and external iterations, whereas case `C` also includes the initialization of the first guess solution using the DP1 operator. Results of the analysis are shown in Tab. 4.1. These results show that the sole use of the MOC requires far more iterations (external and internal) and computational time with respect to the accelerated cases. Remark also the time gain obtained by initializing the iterative solution using the DP_N operator.

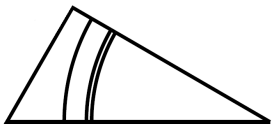


Figure 4.5: Triangular pincell representing the basic 2D section of an infinite hexagonal lattice configuration. Along the axial direction the composition of the fuel varies: fissile material composes the lower part, while fertile material composes the upper part.

	Externals	Internals	Time (sec)	Speed Up
Case `A`	12	150646	40474	–
Case `B`	7	13776	3825	10.6
Case `C`	5	9840	2720	14.9

Table 4.1: Results of the application of the DP1 acceleration for the test case in Fig. 4.5. Case `A` is the case where only MOC free iterations are used. Case `B` uses the DP1 synthetic acceleration of internals and external iterations. Case `C` also includes the initialization of the first guess solution using the DP1 solution of the multi-group problem.

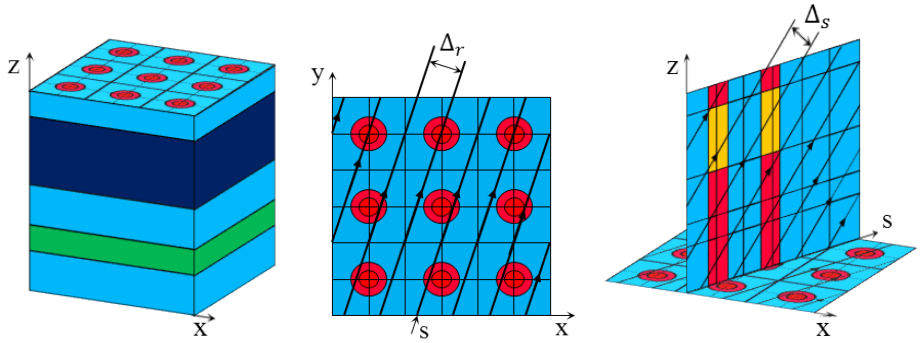
Chapter 5

Tracking strategies

This chapter is dedicated to the tracking techniques that have been implemented in the MOC solver TDT. This solver is capable of tracking trajectories in two-dimensional unstructured meshes of any kind by using the so-called method of compound trajectories, which allows the treatment of the geometrical symmetries without applying any approximation on the boundary fluxes. The tracking techniques discussed in this chapter provide an extension of the method of compound trajectories to 3D axial geometries and have been published in an international journal [45]. In the following section we will show how two-dimensional tracking techniques can be used to treat three-dimensional domains. Then, we will introduce the method of the compound trajectories and how this has been extended to 3D cases. In particular, we will introduce the concept of periodic trajectories: trajectories that allow the exact treatment of symmetries in infinite lattice domains. In the last section we also describe a method to assure constant spacing between trajectories during tracking. This property allows a reduction of operations needed to compute spatial integrals.

5.1 Basic tracking strategy

The Method of Characteristics makes use of a trajectory-based discretization of the geometry to solve the mono-group transport equation. For each direction $\mathbf{\Omega} \in S_N$ a set of trajectories is built in order to cover the problem geometry. We denote by φ and θ respectively the azimuthal and the polar angles of $\mathbf{\Omega}$. In the following we will discuss about specialized methods for tracking in 3D axial geometries. We limit our analysis to systems that can be described by the product of a 2D unstructured mesh, $\mathcal{D}_{2D}$, and a 1D



(a) (b) (c)

Figure 5.1: Tracking strategy for 3D axial geometries. On the left the original geometry composed of the 2D section extruded along z. The tracking is done first by considering the 2D section and tracking 2D trajectories with spacing Δr (center). Each 2D trajectory is used to generate an sz-plane, which represent an axial cut of the geometry. The actual 3D trajectories are tracked on each sz-plane with spacing Δs (figure on the right).

axial mesh, $\vec{H} = \{\Delta h_{r_Z}\}$ for $r_Z = 1, \ldots N_Z$ (fig. 5.1(c)). We assume $\mathcal{D}_{2D}$ to be partitioned into homogeneous regions, $\mathcal{D}_{r_{2D}}$.

For these geometries tracking can be done in two phases. First we fill $\mathcal{D}_{2D}$ with 2D trajectories with direction φ. Such operation is done by projecting the boundaries of $\mathcal{D}_{2D}$ in the direction perpendicular to φ, and by dividing such projection in segments of length Δr. Trajectories are then tracked starting from the middle points of these segments until they exit from the system. By $s \in [0, L_{2D}]$ we denote the coordinate of a 2D trajectory defined with respect to its entering point on the 2D section boundaries, and by $\vec{s} = \{s_{i_{2D}}\}$ the coordinates of the intersections with 2D region boundaries. In the second phase of the tracking, we use each 2D trajectory as support to build the actual 3D ones. By taking the product of the set $\vec{s}$ of 2D intersections and the axial mesh $\vec{H}$, we obtain a Cartesian mesh (hereafter *sz-plane*) describing an axial section of the 3D geometry (Fig. 5.1(c)). The boundaries of each sz-plane are projected in the direction perpendicular to θ. As for the 2D case, such projection is divided in segments of length Δs defining the starting point of 3D trajectories. Hence, the cross-sectional area associated to each trajectory is equal to $\Delta_t = \Delta r \Delta s$.

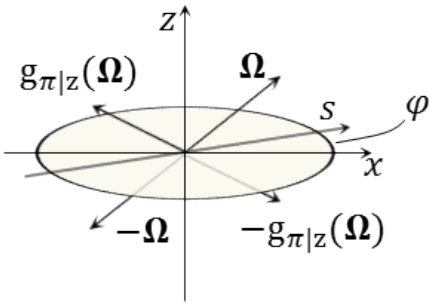


Figure 5.2: Reciprocity relations between tracking directions. Direction $\mathbf{\Omega}$ belongs to the upper unit sphere and it is concordant with the coordinate of the 2D trajectory ($\mathbf{s}$ in the figure). Directions $g_{\pi|z}(\mathbf{\Omega})$ are obtained by π rotation of $\mathbf{\Omega}$ around the z-axis, that is by running the 2D trajectory in backward direction. The directions of the lower unit sphere are obtained by sweeping the 3D trajectories backwards.

5.1.1 Reciprocity

A trajectory in the $\mathbf{\Omega}$ direction can be swept backward to obtain contributions for the direction $-\mathbf{\Omega}$. By exploiting this property we can track only for the directions $\mathbf{\Omega} \in \mathcal{S}^2_{up}$ (i.e., the directions of the upper unit sphere), and obtain the remaining directions by backward sweep. This approach requires both directions $\mathbf{\Omega}$ and $-\mathbf{\Omega}$ to be contained in the S_N quadrature set. Similarly, we can halve the size of the 2D tracking by noticing that a 2D trajectory with angle φ is identical to the trajectory with angle $\varphi' = \varphi + \pi$ swept in backward direction. To exploit this symmetry, the tracking of the 3D upward directions in each sz-plane is done by running the 2D trajectory forwards and backwards. In other words, by denoting with $\mathbf{s}$ the coordinate of the 2D trajectory, the tracking in each sz-plane is done for the directions $\mathbf{\Omega} \in \mathcal{S}^2_{up}$ such that $\mathbf{\Omega} \cdot \mathbf{s} > 0$, and for directions such that $\mathbf{\Omega} \cdot \mathbf{s} < 0$. This tracking strategy requires both directions $\mathbf{\Omega}$ and $g_{\pi|z}(\mathbf{\Omega})$, obtained by rotation of $\mathbf{\Omega}$ through π radians around the z-axis, to be contained in the quadrature set (see Fig. 5.2).

5.2 Exact treatment of symmetries

Some systems are characterized by symmetries such as invariance with respect to planar reflection (see Sec. 1.2.1). When these symmetries exist, we limit the track to only a portion of the domain, called *basic domain*, repre-

senting the generator of $\mathcal{D}$ under the action of the symmetries. The symmetries are taken into account by imposing the so-called *geometrical boundary conditions* (hereafter GC) on the interfaces generated by such domain reduction. In the following we will refer to these portions of the boundaries as *closed boundaries*, while the other boundaries will be denoted as *open*. GC are of the form:

$$\psi^-(\mathbf{r}, \mathbf{\Omega}) = \psi^+(g^{-1}\mathbf{r}, g^{-1}\mathbf{\Omega}),\ \mathbf{r} \in \Gamma_{closed} \tag{5.1}$$

The function g is the *geometrical motion* representing the symmetry, and provides the mapping between the flux entering the basic domain in position $(\mathbf{r}, \mathbf{\Omega})$ and the flux exiting from position $(g^{-1}\mathbf{r}, g^{-1}\mathbf{\Omega})$.

5.2.1 Method of compound trajectories

The method of *compound trajectories* has been introduced in [46] to exactly treat geometrical boundary conditions in 2D geometries. It consists in tracking so that when a trajectory leaves the domain from a point of a closed boundary, $(\mathbf{r}, \mathbf{\Omega})$, there is always another trajectory entering the domain in position $(g\mathbf{r}, g\mathbf{\Omega})$. In practice this is done by tracking each trajectory starting from an open boundary and by following its path until another boundary is reached. If such boundary is closed, the geometrical motion is applied to put the trajectory back into the basic domain and to continue its path. Conversely, the trajectory terminates if the crossed boundary is open. This technique generates compound trajectories containing one or more sub-trajectories with different angles. Since the continuity of the flux between two sub-trajectories is assured by Eq. (5.1), the sub-trajectories belonging to a compound can be swept consecutively starting from the open boundary, and by using the flux leaving a sub-trajectory as the entering flux for the following one. Remark that such method requires all the directions obtained by geometrical motion to be contained in the quadrature set. To impose such constraint, the quadrature formula is built in the *basic angular domain* and extended to the whole sphere by application of the symmetries. By basic angular domain we mean the generator of $\mathcal{S}_2$ under the action of the geometrical motions. For some cases, the construction of the quadrature formula in the basic angular domain does not guarantee the reciprocity relation. These cases include all geometries with azimuthal symmetry under rotation of $2\pi/N$ with N odd. This undesired behavior is eliminated by building the quadrature formula in the angular domain π/N, and by extending it to 2π by N applications of the rotation GC, and by applying the reciprocity condition [46]. Remark also that when we track trajectories

with one direction, we partially cover the domain also for those directions obtained by application of the geometrical motions. Special care is thus needed in order to complete the domain without repeating trajectories.

The current version of TDT already makes use of the method of compounds to track trajectories in arbitrary two-dimensional geometries with geometrical boundary conditions. A detailed analysis of the treatment of GC for 2D geometries goes beyond the scope of this manuscript and we refer the interested reader to [46] for a complete review. In the following we will provide the extension of this method to 3D axial geometries.

Periodic trajectories

A mention is due for the treatment of infinite lattice configurations for which the basic domain has only closed boundaries. In these cases, we can find particular trajectories, called *periodic* trajectories, which have the property of mapping onto themselves after repeated applications of the GC (Fig. 5.3). In [46] the authors analyze 2D infinite lattice configurations of rectangular and hexagonal geometries, and identify the conditions for which the periodicity condition is satisfied. In particular they find that the periodicity condition for a given geometrical configuration only depends on the trajectory angle. For a rectangle of sides a and b, for example, the periodicity condition for translation GCs reads:

$$\varphi_c = tg^{-1}\left(\frac{ma}{nb}\right), m, n \in \mathbb{N}. \tag{5.2}$$

Remark from this expression that cyclic angles can be arbitrarily chosen by finding suitable values of m and n. The choice of this pair of integers also determines the period of the trajectory which for the rectangular case is:

$$L_{2D} = \sqrt{(ma)^2 + (nb)^2}, \quad m, n \,\text{coprimes}. \tag{5.3}$$

It is important to remark that all parallel periodic trajectories share the same period. The analysis done in [46] shows that, for any 2D lattice configuration, the cyclic angle and the trajectory period only depend on the choice of the pair of integers (n, m).

5.2.2 Extension to 3D axial geometries

The method of compound trajectories developed for 2D geometries can be applied for tracking in 3D axial geometries. Let $\mathcal{D}$ be the 3D axial basic domain with boundaries Γ. The surfaces composing Γ can be either vertical

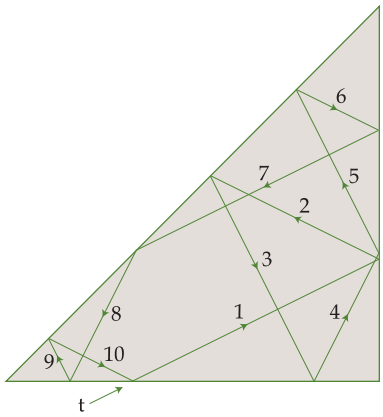


Figure 5.3: Periodic trajectory in a square geometry with 1/8 symmetry (image taken from [46]).

or horizontal depending if they are respectively perpendicular or parallel to the 2D section. GC applied on vertical surfaces are constant along the z-axis and can only affect the two-dimensional components of the trajectory, $(\mathbf{r}_{2D}, \varphi)$. Conversely, GC on horizontal surfaces may only affect the polar angle and the z-coordinate. By consequence, these boundary conditions can be treated separately in the two tracking phases described in Sec. 5.1. When the 2D section is considered, only boundary conditions applied on the vertical surfaces are taken into account, which allows the use of the 2D tracking techniques already developed in [46]. These methods can be applied to any kind of 2D geometry and geometrical boundary condition, and generate the minimum set of 2D compound trajectories covering the domain for the azimuthal directions $\varphi \in [0, 2\pi]$, avoiding repetition due to the equivalence between directions φ and $\varphi + \pi$. The tracking phase proceeds as usual by considering each 2D compound trajectory for the construction of the associated sz-plane. The 3D trajectories are then tracked in upward direction ($\mathbf{\Omega} \in \mathcal{S}^2_{up}$) by running the 2D trajectory forwards and backwards to obtain the directions $\mathbf{\Omega}$ with $\varphi \in [0, \pi]$, and those obtained by π rotation around the z-axis, $g_{\pi|z}(\mathbf{\Omega})$. Unlike the fully open case, the sz-plane is now composed

of *sub-planes* with varying azimuthal direction obtained by the product of the 2D sub-trajectories with the axial mesh. The boundary conditions applied on the vertical surfaces, however, guarantee the continuity of the flux across the interfaces between sub-planes. This property allows to 'unfold' the sub-planes, and to track 3D trajectories as if the sz-plane were a continuous Cartesian rectangular domain (see Fig. 5.4).

Remark from Fig. 5.2 that the direction obtained by reflection of $\mathbf{\Omega}$ with respect to the horizontal plane is equivalent to direction $g_{\pi|z}(\mathbf{\Omega})$ because of the reciprocity condition. Therefore, when reflective GCs are applied on the horizontal surfaces, a trajectory with initial direction $\mathbf{\Omega}$ may be reflected and partially cover the sz-plane for directions $g_{\pi|z}(\mathbf{\Omega})$. In order to avoid repetitions we need to consider the boundary conditions applied on each sz-plane. The *top* and *bottom* sides directly inherit boundary conditions from the respective horizontal planes. These can be either open boundaries, or closed boundaries with reflective or translation GC. The boundary conditions on the *left* and *right* sides require more discussion. In fact, we have seen that the 2D trajectories generated with the method of compounds enter and exit the geometry through open boundaries; in these cases the left and right boundaries of the sz-plane are *open*. Contrarily, when the 2D section is only delimited by closed boundaries, 2D trajectories are tracked in order to satisfy the periodicity condition which is of the form:

$$\psi(s) = \psi(s + nL_{2D}), n \in \mathbb{Z}, \tag{5.4}$$

with L_{2D} the period of the 2D trajectory. This condition formally corresponds to a translation, which means that whenever a 3D trajectory crosses the vertical boundaries of the sz-plane, it is put back inside the domain by applying the transformation $g(s) = s \pm L_{2D}$, with sign that depends on whether the crossed boundary is the left (+) or the right one (-). In the following section we will provide the details of the tracking strategy for all combinations of boundary conditions applied on the sides of the sz-planes.

Treatment of GC in 3D Axial Geometries

Here we describe the tracking techniques for several configurations of boundary conditions that can be applied on sz-planes of width L_{2D} and height H. In this section we will call *right* trajectories the 3D trajectories with direction $\mathbf{\Omega} \in S^2_{up}$ obtained by sweeping the 2D trajectory in the forward direction ($\mathbf{\Omega} \cdot \mathbf{s} > 0$, $\mathbf{s}$ being the coordinate of the 2D trajectory). In a similar way, we will call *left* trajectories those with direction $\mathbf{\Omega} \in S^2_{up}$ such that $\mathbf{\Omega} \cdot \mathbf{s} < 0$.

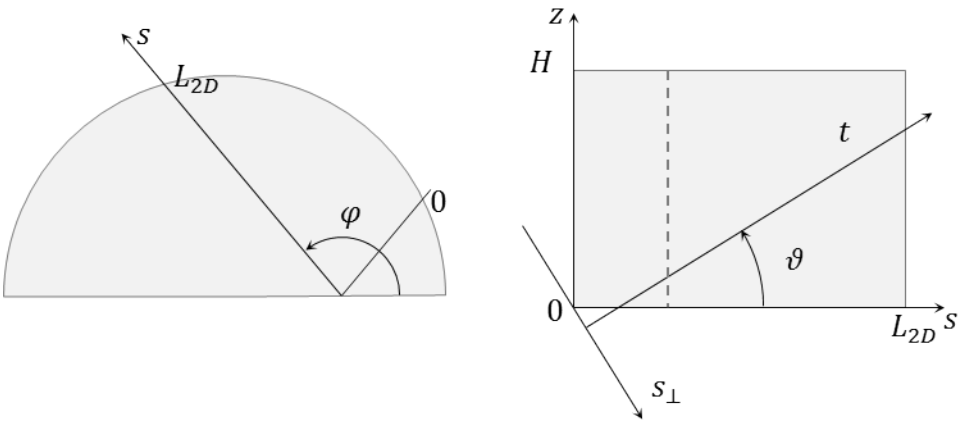


Figure 5.4: On the left one 2D compound trajectory entering the geometry in $s = 0$ and exiting in $s = L_{2D}$ after having bounced on the lower boundary. On the right the sz-plane with the 3D trajectory. The dashed line in the figure shows the limit between the two 2D sub-trajectories: the geometrical boundary conditions on vertical surfaces assures continuity of the 3D trajectory in the sz-plane.

For simplicity of notation we will indicate with symbol $\vartheta \in [0, 2\pi]$ the angle between the 3D trajectory and the positive direction of the 2D trajectory (see fig. 5.4). In this framework, $\vartheta_r \in [0, \pi/2]$ is the ϑ angle for right trajectories, while left trajectories have angle $\vartheta_l \in [\pi/2, \pi]$ with $\vartheta_l = \pi - \vartheta_r$. We will also denote with $s_\perp$ the axis with direction $\vartheta_{r/l} - \pi/2$ and zero on the origin of the sz-plane, and by $\mathcal{L}$ the portion of $s_\perp$ to track for right ($\mathcal{L}_R$) and left ($\mathcal{L}_L$) trajectories (see fig. 5.5). Remark also that, because of reciprocity, the trajectories with angle $\vartheta + \pi$ are equivalent to trajectories with angle ϑ swept in backward direction (denoted with symbol $\hat{\vartheta}$).

The first case is the case where all the boundaries are *open*. Right trajectories are obtained starting from the projection of the boundaries on $s_\perp$: $\mathcal{L}_R = [-H\cos\vartheta_r, L_{2D}\sin\vartheta_r]$. The resulting trajectories enter the domain either from the left or from the bottom boundaries. Left trajectories are tracked in the same way resulting in $\mathcal{L}_L = [0, L_{2D}\cos\vartheta_r + H\sin\vartheta_r]$. In the second case considered the vertical surfaces have *translation* GC, while horizontal surfaces are considered as open. In this case the tracking length for right trajectories is $\mathcal{L}_R = [0, L_{2D}\sin\vartheta_r]$, while for left trajectories it corresponds to $\mathcal{L}_L = [0, L_{2D}\sin\vartheta_r]$.

For geometries with *up/down symmetry* (Fig. 5.5), one of the horizontal

boundaries is characterized by reflective GC. For simplicity we assume this boundary to be the upper one. In such case $\mathcal{L}_R = [-H\cos\vartheta_r, L_{2D}\sin\vartheta_r]$ as for the previous case. However, one must notice that due to the reflection on the upper boundary, right trajectories include sub-trajectories with direction $2\pi - \vartheta_r$. These coincide with left trajectories because of reciprocity. Therefore, the tracking length for left trajectories differs from the open case and results being $\mathcal{L}_L = [0, H\cos\vartheta_r]$, which corresponds to the portion of the domain not covered by the tracking of right trajectories. For up/down symmetry with *translation* GC on the vertical sides, we only have to track right trajectories from the portion $\mathcal{L}_R = [0, L_{2D}\sin\vartheta_r]$ of $\mathbf{s}_\perp$, while no left trajectories are tracked since all right trajectories will eventually 'bounce' on the upper surface and re-enter the domain in the left (backward) direction.

If the upper and lower sides of the sz-plane are characterized by translation GC, and the vertical sides are open, the tracking lengths are $\mathcal{L}_R = [-H\cos\vartheta_r, 0]$ and $\mathcal{L}_L = [L_{2D}\sin\vartheta_r, L_{2D}\sin\vartheta_r + H\cos\vartheta_r]$. Finally, when upper and lower boundaries are reflective, and the vertical surfaces are open, right trajectories are tracked in the portion $\mathcal{L}_R = [-H\cos\vartheta_r, 0]$. For this case the tracking length for left trajectories has a complicate expression. An easier solution can be obtained by exploiting the reciprocity condition, and by tracking left trajectories in the opposite direction. In this case the tracking length for opposite left trajectories is $\hat{\mathcal{L}}_L = [-H\cos\vartheta_r, 0]$.

When Γ is only composed of closed boundaries, it is still possible to seek for 3D periodic trajectories. We have seen in Sec. 5.2.1 that the method of compounds for 2D geometries generates a set of cyclic angles φ_c and periodic 2D trajectories with period $L_{2D}(\varphi_c)$ which depends only on the value of φ_c. The associated sz-planes are rectangles of height H and length $L_{2D}(\varphi_c)$ with translation boundary conditions on the vertical sides. In this configuration, cyclic polar angles are defined as follows [46]:

$$\tan(\vartheta_c) = \frac{m f_z H}{n L_{2D}(\varphi_c)}, \tag{5.5}$$

whit period:

$$L_c = \sqrt{(m f_z H)^2 + (n L_{2D}(\varphi_c))^2}. \tag{5.6}$$

In these equations $m, n \in \mathbb{Z}$ are coprimes and define the periodicity of the trajectory. The f_z parameter takes into account the boundary conditions applied on the top and bottom surfaces of the geometry:

$$f_z = \begin{cases} 1, & \text{Translation} \\ 2, & \text{Reflection} \end{cases} \tag{5.7}$$

z		s: Open	s: Translation
Open	$\mathcal{L}_R$	$[-H\cos\vartheta_r, L_{2D}\sin\vartheta_r]$	$[0, L_{2D}\sin\vartheta_r]$
	$\mathcal{L}_L$	$[0, L_{2D}\cos\vartheta_r + H\sin\vartheta_r]$	$[0, L_{2D}\sin\vartheta_r]$
One Refl.	$\mathcal{L}_R$	$[-H\cos\vartheta_r, L_{2D}\sin\vartheta_r]$	$[-H\cos\vartheta_r, L_{2D}\sin\vartheta_r]$
	$\mathcal{L}_L$	$[0, H\cos\vartheta_r]$	$[\varnothing]$
Transl.	$\mathcal{L}_R$	$[-H\cos\vartheta_r, 0]$	$[0, L_{2D}\sin\vartheta_r/n]$
	$\mathcal{L}_L$	$[L_{2D}\sin\vartheta_r, L_{2D}\sin\vartheta_r + H\cos\vartheta_r]$	$[0, L_{2D}\sin\vartheta_r/n]$
Two Refl.	$\mathcal{L}_R$	$[-H\cos\vartheta_r, 0]$	$[0, L_{2D}\sin\vartheta_r/n]$
	$\mathcal{L}_L$	$[-H\cos\vartheta_r, 0]^\star$	$[\varnothing]$

$\star$ values are tracked for the opposite left angle $\hat{\vartheta}_l$.

Table 5.1: Tracking length, $\mathcal{L}_{R/L}$, in the sz-plane for different configurations of geometrical boundary conditions. On the s-axis only open or translation conditions are possible. Along the axial direction all possible combinations of open, reflective and translation conditions are considered.

For full translation we track for both right and left trajectories on the portion $\mathcal{L} = [0, L_{2D}\sin\vartheta_c/n]$, while for reflection only right trajectories are tracked in the segment $\mathcal{L} = [0, L_{2D}\sin\vartheta_r/n]$.

From Eq. (5.5) we can see that values of $\tan(\vartheta_c)$ are dense in $\mathbb{R}$. The same property holds true for φ_c [46]. By consequence, we can always construct an angular quadrature formula S_n^c of cyclic angles $\mathbf{\Omega}_c = (\varphi_c, \theta_c)$ arbitrarily close to an optimal S_N set. However, an accurate value of cyclic angles requires both m and n to be large, increasing the length of the trajectory. Generally, such effect does not impact the total size of the tracking since this is always built to cover the whole geometry without repetitions. However, it may happen that, for sufficiently long periods, the tracking length is smaller than the actual trajectory spacing. For these cases, the latter is adjusted to fit in the tracking length, resulting in an increase of the tracking size. Tab. 5.1 summarize the tracking techniques for the set of GC considered in this section.

5.2.3 Boundary flux for periodic trajectories

In the previous section we have seen that, as for the 2D case, it is possible to seek for periodic trajectories for completely closed 3D domains. These trajectories can be exploited to directly retrieve the unknown fluxes entering

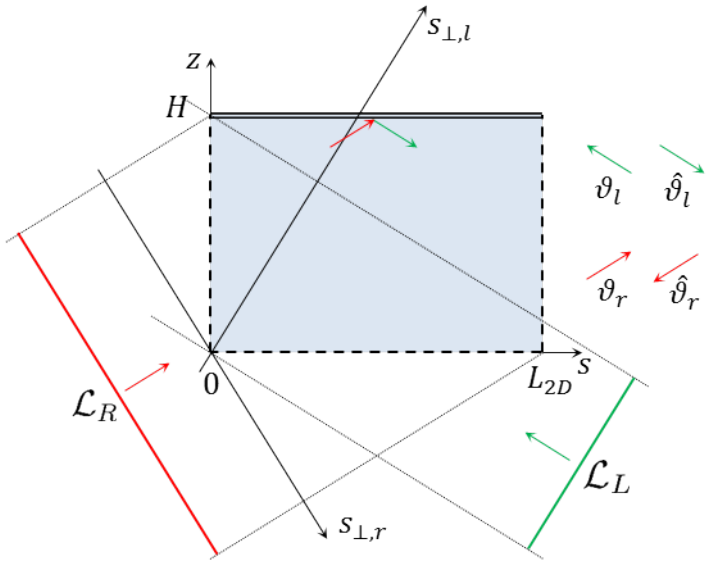


Figure 5.5: Tracking strategy for open boundary conditions (thick dotted lines) on the vertical sides, and up/down symmetry on the top horizontal surface (thick double line). Trajectories in the *right* direction (red) are tracked from $\mathcal{L}_R$ and partially cover the domain also for left trajectories (green) due to the reflection on the top surface. *Left* trajectories are tracked in order to avoid repetition.

the boundaries. By imposing the periodicity condition to the integral form of the transport equation for a trajectory with period L_c we obtain:

$$\psi^- = \frac{\int_0^{L_c} dt' q(t') e^{-\tau(t', L_c)}}{1 - e^{-\tau(0, L_c)}}, \tag{5.8}$$

where $t \in [0, L_c]$ is the local coordinate along the trajectory and $\tau(t_1, t_2)$ is the total optical length between points t_1 and t_2 of the trajectory. The solution of the last equation requires an additional transport sweep, since we need to collect the contributions of the angular sources along the whole trajectory. However, we can exploit the continuity condition to write the integral form of the transport equation for the incoming boundary flux with respect to the flux at position $t^\star < L_c$:

$$\psi^- \equiv \psi(L_c) = \psi(t^\star) e^{-\tau(t^\star, L_c)} + \int_{t^\star}^{L_c} dt' q(t') e^{-\tau(t', L_c)}. \tag{5.9}$$

From the last equation we can see that the contribution of $\psi(t^\star)$ becomes negligible for $\tau(t^\star, L_c) \gg 0$. Therefore, the boundary flux can be computed with arbitrary precision by sweeping the trajectories only for $t > t^\star$ and by setting $\psi(t^\star) = 0$, where $t^\star$ is defined such that $\tau(t^\star, L_c) > \tau_{cut}$. Remark that with this approach we only have to sweep part of the trajectory to obtain the boundary fluxes, resulting in a speed up of the calculations. In particular, for very long trajectories the speed up asymptotically tends to a factor 2.

5.2.4 Constant Trajectory Spacing

The cross sectional area associated to each 3D trajectory defines the spatial integration weight ($\Delta_t = \Delta r \Delta s$) used to compute the region integrated leakage term (Eq. (4.21)). The cost of the calculation of such integral is due to the calculation of the net current and to the multiplication by the spatial weight, and linearly increases with the number of trajectories that cross the region. The multiplication by the spatial weight can be avoided by imposing a constant value of the spatial weights associated to the chords that cross each region. This allows to move out from the sum the spatial weight and to rewrite the integral in the following optimized form:

$$\Delta \vec{J}_{rj} \approx \frac{\Delta_t}{V_r} \sum_{\substack{t \parallel \boldsymbol{\Omega} \\ t \cap \mathcal{D}_r}} \left\{ \vec{P}_r[\zeta(t_t^+)] \psi(t_t^+) - \vec{P}_r[\zeta(t_t^-)] \psi(t_t^-) \right\}. \tag{5.10}$$

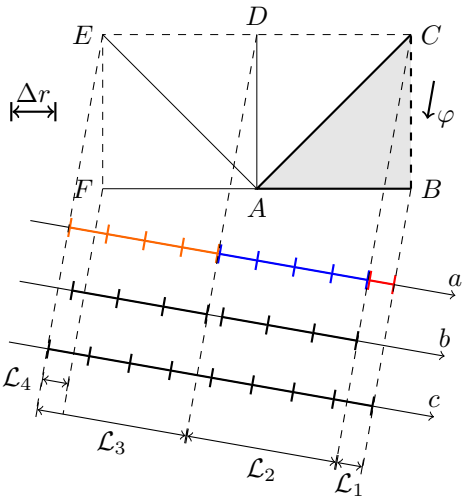


Figure 5.6: Different methods of spatial weight adaptation. The basic domain (gray) is a 1/8 geometry with one open boundary (dashed). Reflective boundary conditions are applied on the other boundaries (black solid lines). $\mathcal{L}_i$ is the projection of the open boundary on the axis perpendicular to the trajectories. Δr is the reference value of trajectory spacing. For simplicity of representation the adaptation methods are shown on the complete geometry although these are applied only on the basic domain. a) Standard Compound-Trajectory [46]: the weight is adapted for each $\mathcal{L}_i$; this results in trajectory-dependent spatial weights (different colors). b) Constant weight. c) Modified adaptation criteria: the weight is adapted to the whole projection, resulting in constant-per-angle spatial weights.

A constant spatial weight can be imposed by tracking trajectories with constant spacing. However, a crude application of constant spacing may produce an incomplete tracking, especially when applied together with the method of compound trajectories. To understand this phenomenon we need to recall the basic steps of the method of compound trajectories. We will take Fig. 5.6 as a practical example, while a complete treatment of 2D geometries can be found in [46]. The figure shows the basic domain (in grey) of a 2D square open geometry with one-eight symmetry. In the same figure, the symmetry boundary conditions are applied repeatedly in order to complete the geometry in the domain $[0, \pi]$. This is done only to simplify the representation of the tracking strategy: in the figure, consider the open boundary DC, symmetric of BC with respect to AC, and its projection in the direction orthogonal to φ ($\mathcal{L}_2$). This projection is equivalent to the one obtained by projecting BC in the direction orthogonal to the symmetric of φ with respect to AC. By keeping this in mind, the tracking technique for compound trajectories proceeds as follows: first the basic direction φ is considered and trajectories are tracked from the *tracking length* $\mathcal{L}_1$, corresponding to the projection of the open boundary in the direction orthogonal to φ. Then, tracking moves to angle $g\varphi$ obtained by applying the reflection (g) to the basic angle φ, and trajectories are tracked from the projection $\mathcal{L}_2$. This procedure follows until the domain $[0, \pi]$ is completed. By using a constant spacing for all tracking lengths (case b in figure), the resulting tracking may be incomplete and even miss important parts of the system (missing part in $\mathcal{L}_2$). To avoid this problem the standard method consists in adjusting the trajectory spacing to be divisor of each tracking length. In this way, however, the spacing between trajectories becomes dependent on the initial tracking angle (case 'a' in Fig. 5.6). Now, recall that one trajectory may change its direction depending on the boundary surfaces crossed along its path (Section 5.2.1). Therefore, it is easy to imagine that the trajectories crossing a region for a given angle may have been generated by geometrical motion of trajectories with different initial angle, preventing the use of the optimized formula in Eq. (5.10). A different approach is to use a generalized adaptation criteria: instead of adjusting the spacing with respect to the tracking length for a given angle, the spacing is adjusted with respect to the projection of the complete geometry:

$$\mathcal{L}_{TOT} = \bigcup_n \mathcal{L}_{g^n\varphi} = k\Delta r,\ k \in \mathbb{N}. \tag{5.11}$$

In this expression, the geometrical motion is applied in order to obtain the projection of the full open boundary onto the direction perpendicular to

φ. Because of the periodicity of the symmetry group, the projection of the total boundary is equal for all the angles generated by geometrical motion of φ. Therefore, by adapting the trajectory spacing to the total projection we assure a constant weight for all the trajectories with the same angle, as well as a regular tracking (method *c* in Fig. 5.6). Compared to the standard adaptation method (case a), this new strategy never takes into account the discontinuities of the geometry boundaries, introducing an additional error in the spatial integration. However, such error is of the same nature of the one introduced by not adapting the tracking to all the discontinuities internal to the domain.

Unfortunately, this approach can not be completely extended to the adaptation of Δs for all those trajectories with same angle $\mathbf{\Omega}$. The difficulty lies in the fact that, in general, 2D trajectories with same azimuthal direction have different length, and it is impossible to find a Δs divisor of all these lengths. However, if the 2D section of the domain is characterized only by exact boundary conditions, the length of the trajectory depends only on the azimuthal angle φ. For these cases a unique Δs can be found and constant spacing is possible without approximation of the geometry. For open cases, a constant spacing can be still be applied by accepting the integration error derived of 'missing' trajectories. This error can be reduced by tracking so that the missing parts corresponds to peripheral outgoing trajectories.

Chapter 6

Optimized methods for the trajectory sweep

The application of the MOC algorithm for actual calculations in three-dimensional geometries is mainly limited by its computational costs. In particular, we have seen in Chapter 4 that the cost of the MOC is mainly due to the memory required for trajectory data, and due to the number of operations required to solve the transmission equation along the characteristic lines. In three-dimensional geometries, the number of intersections for a relatively small problem can be of the order of some hundreds of millions, an amount which already challenges standard workstations in terms of memory usage, and which requires several hours of calculation.

In this chapter we describe in detail some tracking methods that have been implemented in TDT to accelerate its execution and reduce its memory needs. These methods strongly exploit the regularities that can be found in Cartesian axial geometries, for which each sz-plane is Cartesian. Optimized tracking techniques for these geometries are mentioned in [47] and in [48] but no detail is provided. The chord classification method has been developed to reduce both memory and computational costs of the sweep algorithm. In this method we recognize chords with same length to avoid their storage. A similar categorization is applied to pre-compute escape coefficients for recognized optical lengths, and reduce the costs of the trajectory sweep.

An alternative storage method for trajectories is also proposed. In this approach, trajectories in each sz-plane are represented by sequences of crossed surfaces. This representation allows a fast reconstruction of the trajectories, and a reduction of the memory requirements.

The treatment of these techniques and some results have been subject

of a publication in an international conference [49], and in an international journal [45].

6.1 Chord Classification Method

The sweep algorithm requires, among other data, the knowledge of the chord length associated to each intersection of the trajectories with the geometry. Chord lengths are generally stored in records of the type:

$$C_t = \{l_i, 1 \leq i \leq N_\chi^t\}. \tag{6.1}$$

Each record contains the chord lengths of all the N_χ^t intersections of each trajectory, ordered according to the actual sequence of regions crossed along the trajectory path. This kind of storage is well suited for the trajectory sweep because it naturally follows the order of solution of the transmission equation, allowing an optimal data access (see Sec. 6.1.1 for details): at the ith intersection the ith chord length is loaded to compute the associated optical length and escape coefficients; then, Eq. (4.11) is solved to obtain the exiting flux and to proceed to the treatment of the next intersection. The computational costs associated to this strategy linearly increase with the number of intersections: on one side because of the memory required to store the chord lengths, and on the other because of the operations required to compute the escape coefficients.

Remark that the storage format in Eq. (6.1) requires the chord lengths to be stored for each intersection, regardless of the fact that this information is actually required. This storage is therefore well suited whenever the chord lengths cannot be estimated a priori, which is the case of completely unstructured meshes. However, the axial regularity of 3D axial geometries allows identifying recognized types of chords that share the same chord length (*chord-classes*). The idea of the *Chord Classification Method* is to identify the set of chord classes representative of the whole set of chords. A similar approach is used to classify escape coefficients, $\vec{E}_i$ in Eq. (4.12), which only depend on the macroscopic cross section and chord length. The advantages are twofold:

- Reduction of the number of chords that have to be stored;
- Reduction of the number of escape coefficients that have to be computed for the sweep.

The definition of the classes is done according to some geometrical property of the chord. A first classification criteria is the kind of surfaces delimiting the chord: a *H-chord* enters and exits the homogeneous region through two horizontal surfaces (Fig. 6.1). Each H-chord has length:

$$l_h(r_Z, \vartheta) = \frac{\Delta h_{r_Z}}{\sin \vartheta}, \tag{6.2}$$

where Δh_{r_Z} represents the height of the r_Z-th axial node. According to the last definition, the number of H-chord classes is equal to the number of axial meshes times the number of polar directions in the quadrature formula. Concerning the escape coefficients, these are function of both the chord length and the optical length. The latter can be computed by taking into account the dependence of the cross section on the 2D region number, so we can write:

$$\tau_h(r_{2D}, r_Z, \vartheta) = l_h(r_Z, \vartheta)\Sigma(r_{2D}, r_Z). \tag{6.3}$$

By consequence the escape coefficient of polynomial order k (Eq. (4.12)) for H-chords can be expressed with the following relation:

$$E_{k,h}(r_{2D}, r_Z, \vartheta) = l_h^k(r_Z, \vartheta)F_k(\tau_h). \tag{6.4}$$

The number of H-classes for escape coefficients is equal to the number of 3D regions times the number of polar angles in the quadrature formula.

Similarly to H-chords, *V-chords* (Fig. 6.1) enter and exit the homogeneous region through vertical surfaces. Given their projection on the 2D-plane (Δs_i), chord length, optical length and escape coefficients result in:

$$l_v(i_{2D}, \vartheta) = \Delta s_{i_{2D}} / \cos \vartheta, \tag{6.5}$$

$$\tau_v(i_{2D}, r_Z, \vartheta) = l_v(i_{2D}, \vartheta)\Sigma(\mathcal{R}_{2D}(i_{2D}), r_Z), \tag{6.6}$$

$$E_{k,v}(i_{2D}, r_Z, \vartheta) = l_v^k F_k(\tau_v). \tag{6.7}$$

In these equations $\mathcal{R}_{2D}$ represents the mapping between the local numbering of chords along the 2D track (i_{2D}) and the absolute order number of the 2D region (r_{2D}). The number of V-classes for chord lengths is equal to the number of intersections of 2D trajectories with the 2D section of the geometry times the number of polar angles in the quadrature formula, whereas for escape coefficient this number is also multiplied by the number of axial planes to take into account the dependence on the cross section.

Finally, chords delimited by two different kind of surfaces are those of the mixed type (*M-chords*). Their length depends on the points of intersection

of the trajectory with the vertical and the horizontal surfaces, thus are more difficult to classify. We will discuss the details of the classification techniques for M-chords in Sec. 6.1.3.

Depending on the classification strategy, H/V/M chords can be either considered as *recognized* or *unrecognized*. The difference between the two cases is the storage method and the way the escape coefficients are computed: for recognized chords, chord lengths and escape coefficients are retrieved using the respective class parameters (e.g., chord type, region number, direction). Differently, unrecognized chords are stored using the standard (sequential) storage method, and the associated escape coefficients are computed using the strategy described in Sec. 4.4.1. Clearly, the size of the standard record is reduced to the sole number of unrecognized chords contained in each trajectory:

$$C_t^\star = \{l_i, 1 \leq i \leq N_{NR}^t\}. \tag{6.8}$$

When classification is applied, a two bit information is assigned to each chord to identify the chord type: H-chord, V-chord, M-chord or unrecognized type. $\vec{E}$-classes are created before the in-group iterations using the mono-group macroscopic cross sections and using chord lengths of each class. In particular, M-chords are directly retrieved from stored values (see Eq. (6.14)), while H/V-chords are computed using the 2D tracking, the z-mesh and pre-computed values of the trigonometric functions (Eq. (6.2) and Eq. (6.5)). For each chord swept, the escape coefficient is either computed on-the-fly using the method described in Sec. 4.4.1 (unrecognized chords), or loaded from pre-computed values (recognized).

6.1.1 Computational efficiency

Some remarks are due concerning the computational efficiency of the classification method. We have seen that thanks to this method we can reduce the number of operations required during the trajectory sweep by pre-computing the escape coefficients for each class and by retrieving their values using class identifiers (e.g., chord type, direction, axial node,...). An estimation of this reduction is provided by the Chord Classification Efficiency (CCE), defined as the average number of chords per class. By denoting with index cls a generic class (e.g., escape coefficient for the H-chord with polar angle θ in region (r_{2D}, r_Z)), and by N_χ^{cls} the number of intersections belonging the cls class, the CCE is defined as:

$$CCE = \frac{\sum_{cls} N_\chi^{cls}}{N_{cls}}. \tag{6.9}$$

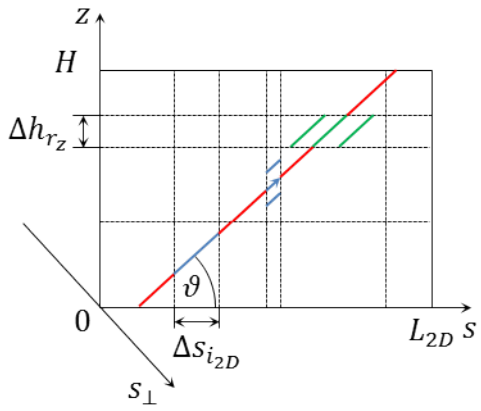


Figure 6.1: One trajectory in the sz-plane. Chords in red are M-chords, while H-chords and V-chords are showed respectively in green and blue. The $s_\perp$ is the axis perpendicular to the trajectory and passing through the zero of the sz-plane. The figure also shows V-chords and H-chords belonging to the same class.

In this equation the sum runs over the N_{cls} classes used for representing the whole set of intersections (unrecognized chords count as belonging to a unique class).

Although the CCE provides an estimation of the floating point operations avoided for the computation of escape coefficients, the actual computational efficiency of the method should also take into account the time required for the memory accesses. This quantity can be hardly estimated a priori because it strongly depends on the data *size* and on the access *pattern*. To understand the phenomena we need an insight of the actual memory architecture. A detailed analysis can be found in **??**.

In general, the total memory is separated in two or more levels ordered hierarchically with respect to the proximity to the CPU. The first level of memory (cache) is a low-latency/low-capacity memory used by the CPU to perform the actual calculations. This memory contains local copies of the actual data contained in higher levels (main memory), which are characterized by larger capacity but higher latency (about $10 - 20$ times larger than cache latency). When the CPU performs an operation on a piece of data, it looks for it inside the first level of memory. If the data is already present, the access to data is practically immediate, and we talk about a *cache hit*. Contrarily, a *cache miss* happens when the data is not found, and it must be

copied from higher levels. Clearly this entails a threshold effect, called *cache effect*, for which the performances of an algorithm abruptly fall whenever the size of the processed data is larger than the size of the cache. For large data the *access pattern* is also fundamental for determining the average access latency. In fact, to reduce the number of accesses to the main memory, data is generally copied per contiguous blocks. A *sequential* access is clearly the optimal access pattern for large data since it produces only one cache miss per block. Contrarily, if the access is *random* the number of cache misses is likely to increase, with consequent reduction of performances.

In the light of this, we can give an heuristic analysis of the performances of the classification method: from the one hand, the application of this method reduces the total data size and the number of operations; on the other hand, it adds randomness to the data access, increasing the execution time. The overall efficiency is therefore a trade-off between these effects and, in general, large CCE are required to see the benefits of the classification methods on the execution time.

6.1.2 Efficiency of H/V-classification

The definition of CCE in Eq. (6.9) provides a *global* estimation of the number of operations avoided by applying a partitioning of the set of chords into classes defined by some class parameters (e.g., chord type, region,..). In a similar way, we can define *partial* CCEs for subset of classes (e.g., only H-chords). This allows measuring the efficiency of the single classification strategy.

A detailed analysis of the efficiency of the chord classification method cannot be easily carried out: the main difficulty is to predict the population of intersections of 2D trajectories with the section of the axial geometry. Here we provide a simplified analysis of the efficiency of the H/V-classification methods in a rectangle R of height Δh and width $\Delta \tilde{s}$, representing one region in the sz-plane crossed by a set of parallel trajectories of direction ϑ. We assume $\Delta \tilde{s}$ to be a characteristic parameter of the 2D chord population (e.g., average 2D chord), and analyze the CCE for varying Δh, representing a characteristic length of the z-mesh. We will also assume the trajectory spacing Δs (see Sec. 5.1) to be sufficiently small to correctly catch the boundary discontinuities. These assumptions do not correspond to the actual TDT implementation, which, instead, uses global tracking techniques (i.e., without taking into account region discontinuities) in unstructured 2D meshes. However, this simplified analysis still allows to catch the main parameters that influence the classification efficiency.

Let us consider the class v of V-chords of direction ϑ crossing rectangle R. The total number of chords belonging to this class can be computed with the following expression:

$$N_\chi^v = \begin{cases} \frac{\Delta\tilde{s}\cos\vartheta}{\Delta s}\left(\tan\vartheta^\star - \tan\vartheta\right), & 0 \le \vartheta < \vartheta^\star \\ 0, & \vartheta^\star \le \vartheta \le \pi/2. \end{cases} \tag{6.10}$$

In this equation we have introduced the quantity $\tan\vartheta^\star$, indicating the *aspect ratio* of the region:

$$\tan\vartheta^\star = \frac{\Delta h}{\Delta\tilde{s}}. \tag{6.11}$$

The number of V-chords assumes a maximum for small ϑ, while it goes down to zero for $\vartheta = \vartheta^\star$. The partial CCE of this class is therefore optimal for horizontal directions and falls to 1[1] at $\vartheta \sim \vartheta^\star$.

A similar analysis can be done for H-chords. The total number of intersections in this case is:

$$N_\chi^h = \begin{cases} \frac{\Delta\tilde{s}\sin\vartheta}{\Delta s}\left(1 - \frac{\tan\vartheta^\star}{\tan\vartheta}\right), & \vartheta^\star < \vartheta \le \pi/2 \\ 0, & 0 \le \vartheta \le \vartheta^\star. \end{cases} \tag{6.12}$$

The CCE for H-chords has a symmetric behavior with respect to the V-classification, having a maximum for vertical directions and decreasing for $\vartheta \to \vartheta^\star$. Concerning M-chords, we can derive the total number of chords with the following expression:

$$N_\chi^m = \begin{cases} \frac{\Delta\tilde{s}\sin\vartheta}{\Delta s}, & 0 \le \vartheta < \vartheta^\star \\ \frac{\Delta h\cos\vartheta}{\Delta s}, & \vartheta^\star \le \vartheta \le \pi/2 \end{cases} \tag{6.13}$$

The efficiency of the M-classification will be not considered in this section and it will be treated later.

We stress out that the values of $N_\chi^{h/v/m}$ are only an estimation of the number of trajectories crossing the actual region. In fact, TDT uses global tracking techniques in unstructured meshes without taking into account region discontinuities. The actual number of trajectories crossing a sz-plane tends to $N_\chi^{h/v/m}$ only for sufficiently small values of Δs. What it is important to note from these equations is that the distribution of the chords depends both on the trajectory direction and on the aspect ratio. In the MOC the first dependence is hidden by the fact that trajectories are tracked for all the angles required to span the direction space. The dependence on

[1] When there is no chords belonging to a given class, the pre-computation of the escape coefficient for that class is not done.

the aspect ratio, in contrast, has a greater impact on the chord population: for decreasing values $\vartheta^\star$ for example, the range of existence of V-chords is progressively reduced, while that of H-chords becomes larger. The opposite is true for large values of the aspect ratio. This behavior is confirmed by the results shown in Fig. 6.2, showing the chord population in an infinite lattice configuration. Calculations are performed for varying values of the mesh height Δh, and by keeping a constant number of tracking angles. In this configuration, the number of H/V-classes is kept constant, so that a direct relation exists between the CCE of H/V-classification and the H/V-chords population. Results show that the CCE reaches a minimum for values of the aspect ratio close to unity, while it increases for very large and very small values.

In typical reactor configurations, the average 2D chord is small as compared to the axial mesh ($0.3\,cm$ for case in Fig. 6.2). This is due to the fact that the majority of heterogeneities are found on the 2D section, whereas the axial composition is more homogeneous. This implies that the classification method is most likely to 'work' on the right side of the minimum shown in Fig. 6.2. Improvements of the efficiency of the classification method can be obtained if the size of the axial mesh is increased. This behavior puts in evidence the synergy between the application of the classification method, and the use of higher order expansion of the axial flux, which allows for the use of wider axial meshes.

6.1.3 Chord Classification for M-chords

Chords of the mixed type can not be grouped in a finite number of classes because the length depends on the intersections of each trajectory with the delimiting surfaces. However, we can still group these chords according to the crossed vertical surface and the direction of the trajectory (see Fig. 6.3). Within this macro-class, a *M-class* is defined by the chord length itself with m_l its unique identifier:

$$l_m(\mathbf{\Omega}, \Gamma_v, m_l) = l. \tag{6.14}$$

When a M-chord is found during the tracking phase, its value is compared to the values of M-chords having the same direction $\mathbf{\Omega}$ and impacting the same vertical surface Γ_v. If there is no corresponding class (within a given tolerance), then a new one is created. Concerning the classification of escape coefficients, notice that, given the surface Γ_v and the direction $\mathbf{\Omega}$, the region order number can be directly retrieved through the mapping $R(\mathbf{\Omega}, \Gamma_v)$.

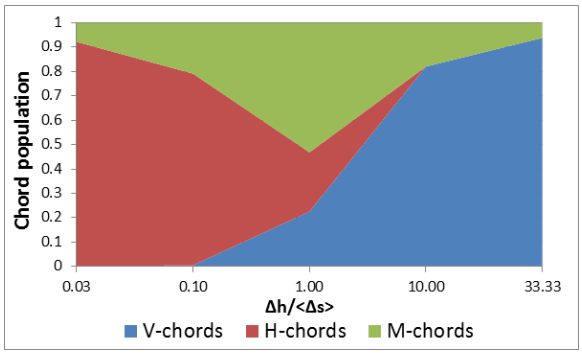


Figure 6.2: Chord population in a infinite lattice configuration of an axial geometry composed of the 2D section shown in Fig. 6.8 and height Δh. The figure shows the distribution of chords for varying values of Δh. Chords are classified only according to the type of delimiting surface (H/V/M) and no distinction is done between chords having different tracking angles. The symbol $< \Delta s >$ denotes the average chord length of the trajectories crossing the 2D section.

Therefore, the number of τ_m-classes and of $E_{k,m}$-classes equals the number of M-classes:

$$\tau_m(\mathbf{\Omega}, \Gamma_v, m_l) = \Sigma(R(\mathbf{\Omega}, \Gamma_v)) l_m(\mathbf{\Omega}, \Gamma_v, m) \tag{6.15}$$

$$E_{k,m}(\mathbf{\Omega}, \Gamma_v, m_l) = l_m^k F_k(\tau_m). \tag{6.16}$$

While for H-chords and V-chords the region number and the trajectory direction are sufficient to identify the class, a M-chord still requires a unique identifier (m_l in Eq. (6.14)) which depends on the chord itself. This information is stored using a sequential storage for the M_m^t chords belonging to trajectory t:

$$M_t = \{m_{i_m}, 1 \leq i_m \leq N_m^t\}. \tag{6.17}$$

This makes the M-classification always disadvantageous in terms of memory usage. However, this method may provide an advantage in terms of operations since $E_{k,m}$ have to be evaluated only for a reduced number of chords.

Analysis and Optimization of M-classification for Infinite Lattice Calculations

In this section we analyze the M-classification method for the case of a prism with rectangular basis of sides a and b, height H and translation boundary

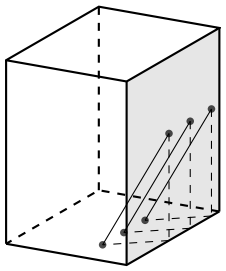


Figure 6.3: M-chords (black solid lines) belonging to the same class. The three black points on the horizontal surface share the same distance from the vertical surface (in grey).

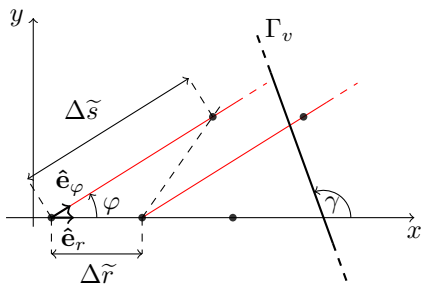


Figure 6.4: Points of intersection (black dots) of 3D trajectories with a horizontal plane. $\Delta\widetilde{r}$: spacing between two 2D tracks (red lines) of azimuthal angle φ. $\Delta\widetilde{s}$: spacing between two 3D trajectories lying on the same 2D track. γ: inclination of the projection of the vertical plane Γ_v on the horizontal surface.

conditions on all surfaces. At the same time we will investigate whether the M-classification can be improved by modifying some tracking parameters such as the trajectory spacing or periodicity. For the case considered, the periodic 2D trajectories are tracked from the horizontal side of the rectangular section (x-axis) with direction, period and spacing [46]:

$$\tan\varphi = \frac{m_{2d}b}{n_{2d}a},\ L_{2D} = \frac{n_{2d}a}{\cos\varphi},\ \Delta r = \frac{a\sin\varphi}{m_{2d}k_{2d}}, \tag{6.18}$$

where the pairs (n_{2d}, m_{2d}) are integers representing the horizontal and vertical periodicity of the trajectory, and $k_{2d} \in \mathbb{N}$ is suitably chosen so that Δr is close to a reference value. Periodic 3D trajectories are then tracked from the bottom of the sz-planes. The expressions for polar angles are similar to the 2D case:

$$\tan\vartheta = \frac{m_{3d}H}{n_{3d}L_{2D}},\ L_c = \frac{n_{3d}L_{2D}}{\cos\vartheta},\ \Delta s = \frac{L_{2D}\sin\vartheta}{m_{3d}k_{3d}}. \tag{6.19}$$

This tracking technique generates trajectories starting from the bottom surface of the geometry.

Let P be the intersection of a trajectory with the horizontal surface Γ_h, and Γ_v be the target vertical surface. According to the definition in Eq. (6.14), the distance between P and Γ_v also determines the chord class (Fig. 6.3). Consider now the regular *grid* generated by the intersections of an infinite set of trajectories with spacing Δr and Δs, and let $\mathbf{P}$ be the vector connecting two generic points of the grid:

$$\mathbf{P}(k,h) = k\Delta\widetilde{r}\,\hat{\mathbf{e}}_r + h\Delta\widetilde{s}\,\hat{\mathbf{e}}_\varphi,\ \ k,h \in \mathbb{Z}. \tag{6.20}$$

In last equation, k and h represent discrete movements respectively along the $\hat{\mathbf{e}}_r$ and the $\hat{\mathbf{e}}_\varphi$ directions (Fig. 6.1.3), while $\Delta\widetilde{r} = \Delta r/\sin\varphi$ and $\Delta\widetilde{r} = \Delta s/\sin\vartheta$ represent the spacing on the horizontal plane. Consider now a vertical surface with orientation γ. We want to find the points on the grid that have the same distance from the target vertical surface. In other terms, find (k,h) solutions of:

$$\mathbf{P}(k,h) \parallel S_v. \tag{6.21}$$

The pair $(k,h)_{basic}$ is the particular solution of Eq. (6.21) with k and h coprimes. The value $\|\mathbf{P}(k,h)_{basic}\|$ is an index of the efficiency of the classification method as it represents the minimal distance between two chords belonging to the same class. The efficiency of the classification method clearly depends on the orientation of the target vertical surface. For surfaces with $\gamma = 0$ the tracking method includes by construction the solution

$(k,h)_{basic} = (1,0)$ which is optimal. In addition, we want to optimize for surfaces with $\gamma = \pi/2$ obtaining:

$$\frac{\Delta\widetilde{s}\cos\varphi}{\Delta\widetilde{r}} = \frac{k_{2d}n_{2d}m_{2d}}{k_{3d}m_{3d}} = \frac{k^\star}{h^\star};\;\; k^\star, h^\star \in \mathbb{Z};\;\; k^\star, h^\star \; coprimes. \tag{6.22}$$

For this case, the distance $\|\mathbf{P}(k^\star,h^\star)_{basic}\|$ is proportional to $k^\star$. Therefore, the chord classification efficiency can be improved by choosing values of the tracking parameters that satisfy Eq. (6.22) with minimum value of $k^\star$.

Until now we have considered an infinite grid of intersections of regularly spaced parallel trajectories. In practice, when geometrical motion boundary conditions are applied, trajectories undergo several geometrical motions and repeatedly cross the same horizontal surface before exiting the domain. Such approach does not assures regularity of the grid. Indeed, every time boundary conditions are applied to the trajectories, the resulting grid of intersections is shifted with respect to the original one, generating new M-classes. Such behavior can be avoided by imposing invariance of the grid with respect to the geometrical motion g:

$$g\left[\mathbf{P}(k,h)\right] = \mathbf{P}(k',h');\;\; k,h,k',h' \in \mathbb{Z}. \tag{6.23}$$

For $g_x(\mathbf{v}) = \mathbf{v} - a\hat{\mathbf{e}}_x$, representing the translation applied to trajectories crossing the vertical sides of the 2D section, the solution of Eq. (6.23) requires $a = k_x\Delta\widetilde{r}$ with $k_x \in \mathbb{N}$. This condition is satisfied by construction since $k_x = k_{2d}m_{2d} \in \mathbb{N}$ (see Eqs. (6.18)). The geometrical transformation for the translation on the horizontal sides of the 2D section is $g_y(\mathbf{v}) = \mathbf{v} - b\hat{\mathbf{e}}_y$, and provides:

$$\frac{b}{\Delta\widetilde{r}\tan\varphi} = k_{2d}n_{2d} = k_y, \qquad k_y \in \mathbb{N} \tag{6.24}$$

$$\frac{b}{\Delta\widetilde{s}\sin\varphi} = \frac{m_{3d}k_{3d}}{m_{2d}} = h_y, \qquad h_y \in \mathbb{N} \tag{6.25}$$

In last two equations we used Eqs. (6.18) and (6.19) for trajectory spacing and tracking angles. The solution of Eq. (6.24) is included by construction, while Eq. (6.25) is satisfied only for particular values of (m_{3d}, k_{3d}, m_{2d}). In our implementation, polar cyclic angles are chosen by assuming $m_{3d} = j_{3d}m_{2d}$ with j_{3d} integer. This simplification assures Eq. (6.25) to be satisfied for any value of tracking parameters.

Finally, we consider the effect of boundary conditions applied on the top and bottom surfaces of the geometry. In case of translation, a 3D trajectory reaching the top surface is put back into the basic domain on a point lying

on the bottom side of the geometry. The action of the translation does not guarantee that the new point belongs to the original grid. We need to impose the grid to be invariant to the geometrical motion $g_z(\mathbf{v}) = \mathbf{v} - H\cot\theta\hat{\mathbf{e}}_\varphi$, which leads to

$$\frac{H\cot\theta}{\Delta\widetilde{s}} = n_{3d}k_{3d} = k_z,\ k_z \in \mathbb{N}, \tag{6.26}$$

which is already satisfied since both n_{3d} and k_{3d} are integers.

Notice that, according to Eq. (6.14), classification is done for chord with same angle and chord length. Therefore, we only have to concern about translation boundary conditions, or composition of geometrical motions resulting in a translation. By consequence, the analysis done in this section can be easily extended to 1/4 and 1/8 symmetries on the 2D section by making the substitutions $a \to 2a$ and $b \to 2b$. The substitution $H \to 2H$ is done for taking into account reflective axial conditions.

6.2 Trajectory storage and reconstruction

In the previous section we developed a method for the reduction of the memory needed for the storage of chord lengths. Another important contribution to the memory is the sequence of region order numbers, r_i, crossed by the trajectories:

$$R_t = \{r_i, 1 \le i \le N_\chi^t\}. \tag{6.27}$$

An alternative method has been implemented to further compress trajectory data. As for the chord classification, this method relies on the regularities of axial geometries, in particular on the fact that sz-planes are Cartesian. Trajectories in sz-planes can cross either a *vertical* or a *horizontal* surface: crossing a horizontal surface means a change of the z-region coordinate while the 2D-region order number remains unchanged; the opposite is true for vertical surfaces. By consequence, given the direction of the trajectory, the coordinates on the sz-plane of its first region, and the type of surfaces crossed during its path, the sequence of crossed regions is univocally determined. In Fig. 6.5 for example trajectory t_2 starts from local coordinates $(1, 1)$ and crosses 2 vertical surfaces before crossing a horizontal one. The z-coordinate of the first 2 chords remains unchanged while only the s-coordinate varies.

For each sz-plane, the quantity $\mathcal{R}_{2D}$ provides the correspondence between the local 2D chord number (i_{2D}) and the 2D region number (r_{2D}). This quantity is common for all 3D trajectories built from the same 2D track. Notice that, in case of 2D cyclic domain, local 2D chord numbering

is periodic by translation (see Eq. (5.4)). Similarly, $\mathcal{R}_z$ provides the correspondence between the local z-numbering (i_z, invariant to translation) and the actual z-region number (r_z). This is common to all 3D trajectories.

The new method requires additional operations for the trajectory reconstruction but it allows reducing considerably the tracking size. In fact, the sequence of hit surfaces (hereafter HSS) can be stored in a compact record:

$$HSS_t = \{\pm n_j, 1 \leq j \leq N^t_{seq}\}, \tag{6.28}$$

where the sign is used to identify the kind of surface (horizontal or vertical), the integer n defines the number of *consecutive* surfaces of that kind crossed by the trajectory, and N^t_{seq} is the total number of sequences of the trajectory. If we reconsider the example of trajectory t_2 in Fig. 6.5, the HSS record is composed of $N^t_{seq} = 5$ elements: $HSS_2 = \{-2, 1, -3, 1 - 2\}$, and describes the first sequence of 2 vertical surfaces, the horizontal surface, and so on until the trajectory exits the sz-plane. The standard storage (Eq. (6.27)) for the same trajectory requires 8 elements corresponding to the total number of intersections. The Memory Compression Factor (MCF) of the HSS method is defined as:

$$MCF_{HSS} = \frac{\sum_t N^t_\chi}{\sum_t (N^t_{seq} + 4)}, \tag{6.29}$$

where N^t_χ is the number of region crossed by trajectory t, N^t_{seq} the number of sequences of crossed surfaces, while the four additional data represent the coordinates of the first and last crossed regions (for forward and backward reconstruction). The maximal dimension which can be assumed by the HSS descriptor is $N^t_\chi + 5$, containing all the $N^t_\chi + 1$ surfaces delimiting the trajectory and the coordinates of the first and last regions. This condition however is unlikely to happen so that we can safely assume that the HSS storage is always advantageous in terms of MCF. The MCF is hardly estimated a priori because the number of sequences, N^t_{seq}, can vary from trajectory to trajectory (see t_1 and t_2 in Fig. 6.5). However, from the analysis provided in Sec. 6.2.2 it results that the MCF strongly depends on the aspect ratio (width/height) of the regions.

The HSS storage method is well suited for the Chord Classification method. In fact, a set of n_j surfaces of the same kind (horizontal for example) defines $n_j - 1$ recognized chords (H-chords). This information can be used together with other class parameters (e.g., trajectory direction) to identify the chord class and to retrieve the respective values of chord length and escape coefficients. Remark that the presence of two sequences of surfaces of

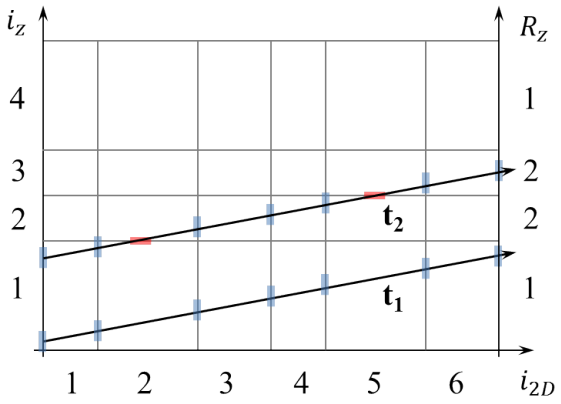


Figure 6.5: The figure shows two trajectories crossing vertical (blue) and horizontal (red) surfaces in the sz-plane. The principal axes indicate the local numbering on the sz-plane, while the secondary vertical axis show the actual numbering of z-regions in case of reflective boundary conditions. The HSS descriptor for t_1 is composed of only one element ($HSS_1 = \{-7\}$) indicating the 7 vertical surfaces crossed by trajectory. Instead, 5 elements are required for trajectory t_2: $HSS_2 = \{-2, 1, -3, 1, -2\}$. The reconstruction starts from the coordinates of the first region, $(1, 1)$, and proceeds by subsequently processing the entries of the HSS descriptor. For trajectory t_2 the first entry indicates that 2 vertical surfaces are crossed, implying the existence of 1 V-chord followed by a joining M-chord. Similarly, the second entry indicates the presence of 0 H-chords, and a joining M-chord. Remark that for the last entry of the HSS no joining chords are required.

different type in the HSS record entails the existence of a *joining* M-chord which has to be correctly accounted during the reconstruction. Trajectory t_2 in Fig. 6.5 provides an example of reconstruction using the HSS storage while Alg. 2 shows the pseudo-code used for the reconstruction.

Forward/backward reconstruction

The sweep algorithm in Alg. 1 shows that the solution of the transmission equation only requires data (e.g., escape coefficient) of the chord which is being swept. For this reason, reconstruction and sweep can be done simultaneously with a negligible impact on the memory. However, since the same trajectory is used twice for the forward and backward sweeps, it can be computationally advantageous to keep the trajectory on a temporary buffer

Input: Σ,$C_t^\star$,HSS_t,i_{2D}^{ini},i_z^{ini},$\mathcal{R}_{2D}$,$\mathcal{R}_z$,M_t,$E_{k,v}$,$E_{k,h}$,$E_{k,m}$
Output: R_t,E_t
$r_1 \leftarrow (\mathcal{R}_{2D}(i_{2D}^{ini}), \mathcal{R}_z(i_z^{ini}))$;
for *all j elements in HSS* **do**
  // $n \leftarrow HSS_t(j)$
  if $n < 0$ **then** // ABS(n) vertical surfaces
    for *(ABS(n)-1) V-chords* **do**
      Update local s-coordinate i_{2D};
      $r_i \leftarrow (\mathcal{R}_{2D}(i_{2D}), \mathcal{R}_z(i_z))$;
      if *recognized* **then**
        $E_{k,i} \leftarrow E_{k,v}$;
      else // Standard strategy
        Update i_{NR}; $l \leftarrow C_t^\star(i_{NR})$; $\tau = l\Sigma$; $E_{k,i} = l^k F_k(\tau)$;
      end
    end
    // 1 M-chord (V→H)
    Update local s-coordinate i_{2D};
    $r_i \leftarrow (\mathcal{R}_{2D}(i_{2D}), \mathcal{R}_z(i_z))$;
    // Get escape coefficient for M-chords
    if *recognized* **then**
      Update i_m; $m \leftarrow M_t(i_m)$; $E_{k,i} \leftarrow E_{k,m}(m)$;
    else // Standard strategy
  else // n horizontal surfaces
    for *(ABS(n)-1) H-chords* **do**
      Update local z-coordinate i_z;
      $r_i \leftarrow (\mathcal{R}_{2D}(i_{2D}), \mathcal{R}_z(i_z))$;
      if *recognized* **then**
        $E_{k,i} \leftarrow E_{k,h}$;
      else // Standard strategy
      end
      // 1 M-chord (H→V)
      Update local z-coordinate i_z;
      $r_i \leftarrow (\mathcal{R}_{2D}(i_{2D}), \mathcal{R}_z(i_z))$;
      // Get escape coefficient for M-chords (as above)
  end
end

Algorithm 2: Reconstruction of region sequence, $R_t = \{r_i\}$, and escape coefficients, $E_t = \{E_{k,i}\}$, of trajectory t with HSS storage and Chord Classification methods. For simplicity of visualization, repeated blocks are not shown. $E_{k,r}$: pre-computed escape coefficients for recognized chords ($r = h/v/m$); M_t: addresses of M-chords (accessed through i_m); $C_t^\star$: unrecognized chord lengths (accessed through i_{NR}); i_{2D},i_z: local coordinates in the Cartesian plane; $\mathcal{R}_{2D}$,$\mathcal{R}_z$: mappings (invariant to translation) from local coordinates to the absolute order number of the 3D regions; i_{2D}^{ini} and i_z^{ini} are the local coordinates of the first chord of the trajectory.

and sweep it backwards avoiding its reconstruction. This constitutes a small overhead in terms of memory requirements.

6.2.1 M-chords reconstruction

The storage of M-chords can be completely avoided if the HSS storage and H/V-classification methods are used. In fact, we can find relations between the chord lengths which allow to compute on-the-fly the M-chord lengths during the reconstruction of region sequence.

We will take the Fig. 6.6 as practical example to explain the method. In the figure you can see that the length of a generic H-chord crossed by n vertical surfaces (macro H-chord) can be written as the sum of two M-chords (the first and last ends of the macro H-chord) plus the length of the $n-1$ V-chords generated by the intersections with these surfaces:

$$l_h = l_m^1 + l_m^2 + \sum_v l_v. \tag{6.30}$$

Here for simplicity we have used the symbol $\sum_v$ to include all the V-chords comprised in the macro H-chord. Similarly, the length of a macro V-chord crossed by multiple horizontal surfaces can be expressed as the sum of two M-chords and the respective set of H-chords. By using the notation in the figure we can write:

$$l_v = l_m^2 + l_m^3 + \sum_h l_h. \tag{6.31}$$

Since the lengths of H-chords and V-chords are all known (see beginning of Sec. 6.1), it is sufficient to know one single value of M-chord to determine the value of the others. These relations can be used during the trajectory reconstruction to compute the values of the M-chords on-the-fly. This method does not allow, however, to pre-compute the escape coefficients, that have to be evaluated for each M-chord as they were treated as unrecognized.

In practice, for each trajectory, only the value l_m^0 is stored. This represents the length of the missing M-chord which joins the beginning of the trajectory: (a) with the previous vertical surface, if the first surface is horizontal; (b) with the previous horizontal surface, in the other case (see Fig. 6.6). During reconstruction, when a set of horizontal (vertical) surfaces is crossed, the sum of H-chords (V-chords) is accumulated on-the-fly. When a M-chord is found, its length is computed using Eq. (6.31) or Eq. (6.30) depending on whether the last surface is vertical or horizontal. In the first case, the macro V-chord is the one corresponding to the s coordinate of the M-chord, while for the other case the macro H-chord is the one corresponding to its z

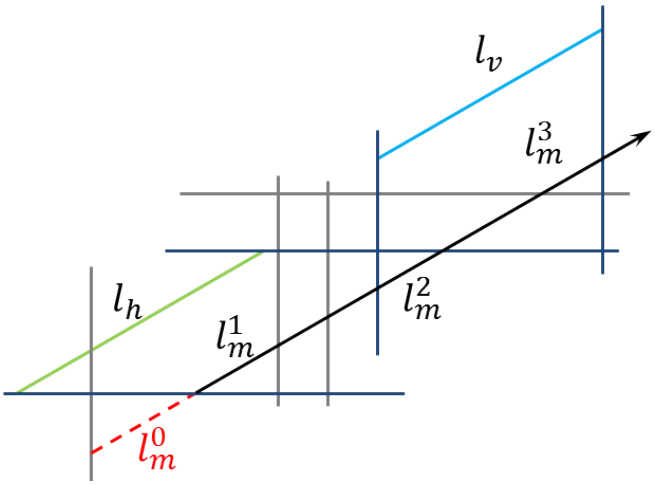


Figure 6.6: A trajectory in the sz-plane: l_v, l_h are the V-chord and the H-chord for a given s-coordinate and z-coordinate; l_m^1,l_m^2,l_m^3 are three M-chords composing the trajectory; l_m^0 is the M-chord joining the beginning of the trajectory crossing a surface of a given type (horizontal in this case) with the previous surface of the other type (in this case vertical).

coordinate. The pseudo-code in Alg. 3 shows a simplified version of Alg. 2 containing only the details required for the M-chord reconstruction. Remark that, while the standard algorithm only requires escape coefficients for the H/V-classes, the algorithm for the M-chord reconstruction also requires the knowledge of the H/V-chord lengths, which can be either computed on-the-fly or retrieved from pre-computed values. This last strategy constitute an overhead in terms of memory usage, whereas the calculation on-the-fly requires one additional operation.

6.2.2 Effect of the axial mesh on the HSS storage

As we have seen in the previous section, the HSS method stores the sequence of crossed region as the sequence of relative displacements along the axis of the (Cartesian) sz-planes. The smaller the number of 'switches' from vertical to horizontal surfaces, the higher the memory compression factor of the HSS. This can be seen, for example, by comparing the HSS descriptors for trajectories t_1 and t_2 in Fig. 6.5. A HSS descriptor of reduced size also reduces the complexity of the reconstruction algorithm (Alg. 2), and favors its serialization. As for the classification method (see Sec. 6.1.2),

Input: $l^0_{m,t}$,l_v,l_h,HSS_t,i^{ini}_{2D},i^{ini}_z
Output: l^m_i
$l \leftarrow l^0_m$;
for *all* j *elements in* HSS **do**
  // $n \leftarrow HSS_t(j)$
  if $n < 0$ **then** // ABS(n) vertical surfaces
    for *(ABS(n)-1) V-chords* **do**
      Update local s-coordinate i_{2D};
      $l = l + l_v$;
    end
    // 1 M-chord (V→H)
    Update local s-coordinate i_{2D};
    $l^i_m \leftarrow l = l_h - l$;
  else // n horizontal surfaces
    for *(ABS(n)-1) H-chords* **do**
      Update local z-coordinate i_z;
      $l = l + l_h$;
    end
    // 1 M-chord (H→V)
    Update local z-coordinate i_z;
    $l^i_m \leftarrow l = l_v - l$;
  end
end

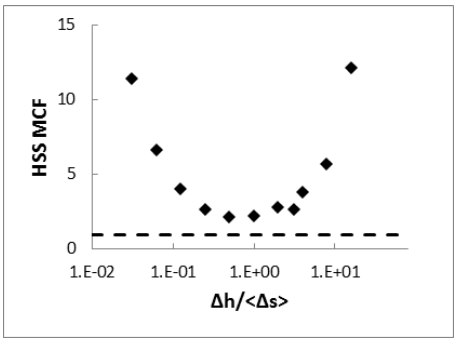
Algorithm 3: Reconstruction of M-chords. Starting from the value l^0_m (see Fig. 6.6), the sequence of M-chords is rebuilt using the HSS storage. For each sequence of surfaces, the values of H/V-chords are accumulated on the local variable t, which is in turn used to compute the value of the joining M-chord.

the efficiency of the HSS storage can be correlated to the aspect ratio of the meshes composing the sz-planes. This dependence is easily explained. Take for example the case of 'tall' meshes (see first axial node in Fig. 6.5): in such configuration a trajectory will most likely cross multiple vertical surfaces before running in a horizontal one. Irrespective of the number of crossed regions, the HSS descriptor always requires two entries to describe this sequence (the number of vertical surfaces and the horizontal surface); conversely, the size of the standard storage grows with the number of intersections. The symmetric situation happens for small values of the aspect ratio, for which it is more likely that the trajectories cross multiple horizontal surfaces before crossing a vertical one. The most unfavorable condition corresponds to the case of 'square' meshes (unitary aspect ratio), for which trajectories have in average the same chance to run into horizontal or vertical surfaces, increasing the size of the HSS storage. This heuristic analysis is confirmed by the results shown in Fig. 6.7(a), which shows the MCF for the infinite lattice configuration shown in Fig. 6.8. Results are provided for a fixed number of tracking angles, and varying height of the axial mesh. The MCF in the figure has a minimum (> 1) for values of the aspect ratio close to unity, while it drastically increases for very large or very small values of the mesh height. As already noticed in Sec. 6.1.2, actual calculations are generally performed for values of aspect ratio larger than unity. This, again, shows the synergy between the efficiency of the MCF and the polynomial expansion of the source along the axial direction, which allows using coarser axial meshes.

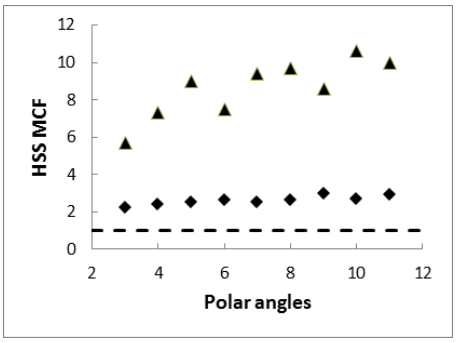
Concerning the dependence of the MCF with respect to the number of the tracking angles, we can make the same argumentation done is Sec. 6.1.2: although the actual sequence of crossed surfaces depends on the trajectory direction, this dependence is weakened by the fact that in MOC trajectories are built in order to span the unit sphere. The variation of the MCF with respect to the number of tracking angles is shown in Fig. 6.7(b). The figure shows the results for the test case in Fig. 6.8 for two values of aspect ratio. The numerical results confirm the weak dependence of the MCF with respect to the number of polar angles in the SN quadrature formula.

6.3 Results

The methods discussed in this chapter have been tested on a geometry whose 2D section is shown in Fig. 6.8 and for varying heights of the axial mesh with respect to the average 2D chord ($\sim 0.3\,cm$). For the case `SMALLTG` the



(a) MCF with respect to the average aspect ratio $\Delta h / \langle \Delta s \rangle$.



(b) MCF with respect to the number of polar tracking angles. Black squares: $\Delta h / \langle \Delta s \rangle = 1$; Black triangles: $\Delta h / \langle \Delta s \rangle = 8$.

Figure 6.7: Memory Compression Factor (MCF) of the HSS storage method with respect to some tracking parameters. The black dashed line represents $MCF = 1$.

axial mesh is uniformly divided in 15 bins of height $0.01\,cm$. In case `UNITTG` the height of each axial bin is equal to the average chord ($0.3\,cm$) and the axial mesh is composed of 50 uniform bins. In the `LARGETG` case, the axial mesh is composed of 5 bins of height $3\,cm$. The calculations have been done using only the step-MOC approximation of the source.

All cases use the same product quadrature formula composed of 16 azimuthal direction and 6 polar directions. The reference trajectory spacing is also common to all cases and it is about $0.05\,cm$ for 2D trajectories and $0.1\,cm$ for trajectories on the sz-planes. In Tab. 6.1 are shown the total number of chords and the MCF_{HSS} as defined in Eq. (6.29). As already discussed, the efficiency of the HSS method is minimal for unitary values of the aspect ratio. The performances of the sweep algorithm in term of computational time and memory are shown in Tab. 6.2. Remark that the Chord Classification method is effective for the classification of the H/V-chords whereas the classification of M-chords actually constitutes an overhead. The reason of this overhead derives from the additional memory required for the M-chord classification and to the low classification efficiency of the method. Remark, in fact, the small global CCE gain obtained by applying the M-classification. From the same table you can also see that the speed up due to the H/V-classification is larger for the `SMALLTG` and `LARGETG` cases due to the higher CCE.

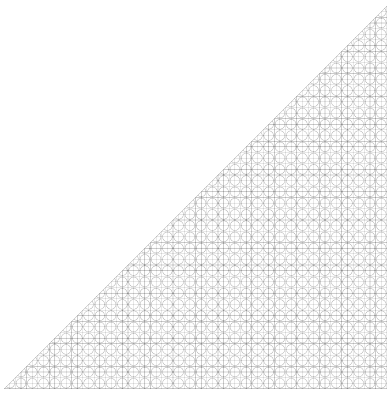


Figure 6.8: Two-dimensional section of an infinite lattice configuration of a cluster of assemblies with $2\pi/8$ symmetry.

	SMALLTG	UNITTG	LARGETG
Chords	$62M$	$102M$	$581M$
MCF_{HSS}	12.8	1.9	5.6

Table 6.1: Total number of intersections (M stands for millions) and Memory Compression Factor due to the HSS method for the three cases considered.

	MEM (GB)	CCE	Time inner (sec)
		SMALLTG	
C0	0.57	1	531
C1	0.45	12.7	459
C2	2.4	14.5	662
		UNITTG	
C0	0.8	1	878
C1	0.7	1.9	857
C2	2.7	2.2	1464
		LARGETG	
C0	2.8	1	3470
C1	1.1	5.5	2583
C2	3.3	7.7	3568

Table 6.2: Memory required for the sweep, Chord Classification Efficiency and execution time of one inner iteration for the three cases considered and for varying classification strategies. C0: no classification applied. C1: only H/V-classification applied. C2: H/V/M-classification.

	M1	M2	M2/M1 overhead
SMALLTG	398	475	+20%
UNITTG	881	1231	+40%
LARGETG	2279	3023	+32%

Table 6.3: Execution time (in seconds) of one internal iteration for the three cases considered and varying reconstruction strategy. In M1 the trajectory is reconstructed and stored on a temporary buffer for backward/forward sweep. In M2 the trajectory is rebuilt and swept separately for the two directions, avoiding the additional buffer. The last column shows the overhead due to the additional reconstruction required for the backward sweep.

In Tab. 6.3 we show the results of different reconstruction strategies under the step-MOC approximation of the source. In case `M1` the trajectory is reconstructed and kept on a temporary buffer for forward/backward sweep, while in case `M2` the reconstruction is done twice for the forward and backward directions. The advantage of this last option is that it does not require any additional buffer during the trajectory sweep. The time overhead of the `M2` strategy is a measure of the time required for the trajectory reconstruction. Remark its dependence with respect to the MCF_{HSS} (see Tab. 6.1), being larger for small MCF_{HSS} and decreasing for increasing compression factors. These results confirm the discussion about the efficiency of reconstruction done in Sec. 6.2.2.

Remark that the relative cost of the reconstruction should, in principle, decrease for higher order expansion of the source due to i) the additional floating point operations required for the transmission equation (see Sec. 4.4), ii) the reduced costs of reconstruction due to a coarser discretization of the axial mesh. For these cases the `M2` strategy becomes more attractive due to the reduced memory requirements.

Chapter 7

Parallel algorithms for the Method of Characteristics

7.1 Introduction

The last decades have seen a drastic change of the architecture of computers. The main reason of this change is the fact that the single processing units have reached the peak performances imposed by current technologies. To compensate this lack of growth, modern computers have multiple independent computing units which inter-communicate through different levels of memory [50]. By consequence, the numerical methods and the programming paradigms are also evolving in order to take advantage of these new architectures.

Concerning the programming paradigms, two are the most trending standards. The *shared memory* paradigm is conceived mostly for multi-core architectures, in which the cores share the same main memory. This paradigm assumes that all the data can be accessed at the same time by the *threads*[1] without too much concern about data locality. Simultaneous I/O operations on the same memory are called *race conditions* and must be carefully considered in shared-memory parallel execution. Only-read race conditions constitute a minor problem since they may only determine an increased memory latency. On the contrary, particular care is required on the shared variables whose values are modified during the parallel execution by the different threads. Simultaneous writing-after-reading or reading-after-writing of the same variable, for example, may invalidate the algorithms if not correctly considered. Contrarily from shared memory, the *distributed memory*

[1] Threads: sequences of tasks executed in parallel.

paradigm assumes that each computing unit has its own copy of data to process, and communications between the units must be explicitly provided in 'synchronization' points. The distributed memory paradigm is more adapted to architectures such as computer clusters for which data locality and communications are important. In our research we only concentrated on shared memory paradigms which are better suited to solve problems of medium size (e.g., assembly, cluster of assemblies) on standard desktop working stations. Our developments have been done using the `OpenMP` libraries [51].

The application of parallel programming methods for neutron transport is a current topic in applied research. One classical approach is the parallelization of the solution of the multi-group problem (see Sec. 1.3) which is done by treating the mono-group problems independently. This approach is not compatible with the standard Gauss-Seidel algorithm which, instead, couples the unknown group-flux with all the fluxes of faster groups (see Sec. 1.4). The independence between group fluxes can be imposed, for example, by substituting the Gauss-Siedel algorithm with a Gauss-Jacobi. Although this approach theoretically determines a deterioration of the convergence rate of the multi-group problem, the use of a low order approximation to accelerate the MOC calculation has shown stabilizing properties. This is, for example, the strategy adopted in [42] where a non-linear Coarse Mesh Finite Difference (CMDF) approximation is used to accelerate the multi-group MOC solution computed withe Gauss-Jacobi algorithm. In this paper the authors show that the convergence of the algorithm is only determined by the convergence of the CMFD acceleration, whereas the stability of the multi-group solution is not affected by the use of the Gauss-Jacobi algorithm. It is reasonable to expect that similar results can be obtained by applying alternative acceleration methods, such as the synthetic DP_N acceleration [43]. This method, however, has not been implemented and results will not be shown.

An alternative method used in the MOC is to keep the Gauss-Seidel scheme and to parallelize the transport sweep for each mono-group problem. This is the approach which we have chosen to explore in our research and it will be detailed in Sec. 7.3.

The last class of parallel methods considered are the so-called domain-decomposition methods. These techniques are based on the partitioning of the spatial domain in sub-domains which are treated independently by different threads. Domain-decomposition methods are particularly adapted to solve large problems on clusters of computers. The reason is that each thread only requires to access data of its associated sub-domain, while data for other sub-domains can be stored elsewhere [26]. The coupling between

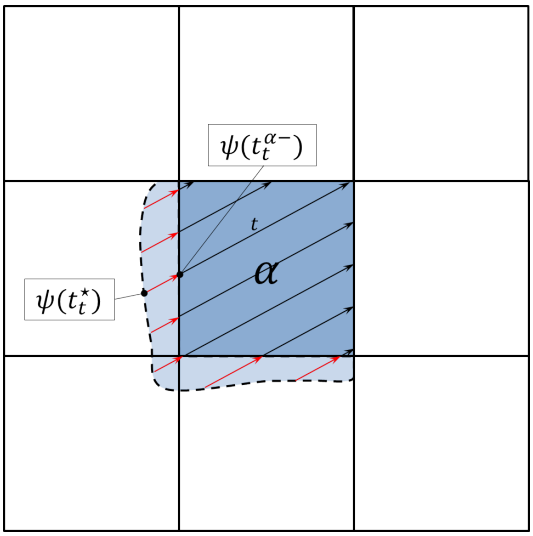


Figure 7.1: Domain-decomposition with trajectory-coherent boundary conditions. Entering fluxes are computed with auxiliary trajectories (red), obtained by backward prolongation of the actual trajectories (black) in the neighbor domains.

the sub-domains is imposed by additional boundary conditions applied on the boundaries of each sub-domain. These boundary conditions provide an expression of the fluxes entering each sub-domain as a function of the fluxes of neighbors, and must be computed and communicated among threads at each iteration. In Sec. 7.2 we discuss the approximations introduced by these additional conditions and we propose a method to exactly treat the coupling between sub-domains.

7.2 Boundary conditions for Domain-Decomposition methods

The definition of additional boundary conditions in domain-decomposition methods may introduce errors in the MOC solution. This is due to the fact that the representation basis used for the boundary fluxes generally differs from that provided by the trajectories. In the short characteristics approach, for example, the trajectory-flux exiting a sub-domain is projected on a spatial basis of the sub-domain boundaries, and then used as entering flux for the trajectories crossing the neighbor domain [26]. This approach re-

duces the quality of the solution, since it introduces numerical diffusion. An alternative method to define trajectory-coherent boundary conditions with arbitrary precision has been considered, but its implementation has been postponed to favor the parallelization of the transport sweep, which results more useful in relatively small cases (e.g., assembly or cluster of assemblies). In this alternative approach, the boundary fluxes are computed using auxiliary trajectories obtained by backward prolongation of the trajectories of each sub-domain in the neighbor domains (see Fig. 7.1). The transmission equation is then used to compute the flux of trajectory t entering the sub-domain α in position $t_t^{\alpha-}$:

$$\psi(t_t^{\alpha-}) = \psi(t_t^{\star})e^{-\tau(t_t^{\star},t_t^{\alpha-})} + \int_{t_t^{\star}}^{t_{in}^{\alpha}} dt' q(t') e^{-\tau(t',t_t^{\alpha-})}. \tag{7.1}$$

This equation requires the value of the angular flux, $\psi(t_t^{\star})$, entering the auxiliary trajectory, and the sources, q', belonging to the neighbor sub-domains. Concerning the first term, one must notice that its contribution to the entering flux, $\psi(t_t^{\alpha-})$, becomes negligible for sufficiently large auxiliary trajectories (large $\tau(t_t^{\star}, t_t^{\alpha-})$). However, the length of the auxiliary trajectories also determines the extent of the coupling between the fluxes of a sub-domain and those of neighbor sub-domains, a fact that increases the amount of data that must be transferred among threads at each iteration. A compromise solution can be obtained by representing the entering flux of auxiliary trajectories with a coarse surface-based spatial representation, and by reducing the length of auxiliary trajectories.

To assure global consistency of the trajectory discretization, the auxiliary trajectories of one sub-domain must coincide with those of neighbors domains. This can be done by applying global tracking techniques to the whole geometry, and by cutting trajectories when they cross sub-domains. Alternatively, a modular ray-tracing method [30, 32, 52] can be applied. It consists in tracking special trajectories in typed sub-domains such that there is always a correspondence between one trajectory exiting one sub-domain and another entering its neighbor.

7.3 Transport sweep parallelism

The sweeping operation is one of the most resource consuming operation in MOC. This is especially true when the MOC is applied to three-dimensional geometries for which the number of intersections may easily reach the order

of tens of billions. A speed up of the MOC algorithm can be obtained by parallelizing the sweep operation. In Alg. 4 we show a modified version of Alg. 1 describing the main steps of the inversion of the fixed-source problem with the MOC, including details of the implementation of method of compound trajectories (see Sec. 5.2.1), and of the trajectory reconstruction (see Sec. 6.2). The trajectory sweep computes the region-integrated value of the net angular current, ΔJ_{jrp}, by accumulating the contributions of the intersections of all the trajectories crossing the problem geometry for all the angles contained in the S_N quadrature formula. The sweep of one single trajectory is naturally sequential since the exiting flux of each intersection directly depends on the exiting flux of the previous one. Differently, separate trajectories can be swept simultaneously since they only depend on the angular emission density, q, which has fixed value at each iteration. It is however important to notice from Alg. 4 that all trajectories access and modify the region-integrated angular net currents, ΔJ_{jrp}, to compute the sum in Eq. (4.21). This operation may rise race conditions since multiple trajectories may simultaneously cross the same region during the parallel execution. In this case all trajectories would read the same value of ΔJ_{jrp} and overwrite it with an updated value which, despite of the order of the writing operations, would miss the contributions of all the trajectories but one (the last executed). This is a typical problem of parallel algorithms and can be solved in three ways:

Data parallelism : it consists in arranging the parallel tasks so that all parallel I/O operations are done on independent data. This method does not require additional memory and operations, making the parallelism optimal.

Data duplication : it consists in creating a private copy of the shared data for each working thread. Each thread accesses sequentially to its own copy preventing the appearance of race conditions. This method requires additional memory as well as additional operations and synchronizations for gathering the partial results computed by each thread.

Mutual exclusion : in this method simultaneous I/O operations are avoided by blocking the access to data when this is being modified by another thread. This method does not require additional memory, but slows down the algorithm since each I/O operation require additional runtime verification of the memory status.

In literature we find an attempt to apply the method of mutual exclusion to the MOC but it has shown very poor performances [53], and it will not

be discussed any further.

The data parallelism strategy requires the definition of groups of trajectories that independently access separate values of the region-integrated angular net currents, which depend on the region order number, r, and of the trajectory direction, $\mathbf{\Omega}_j$. Global tracking techniques in unstructured meshes prevent a simple definition of groups of trajectories crossing separate subsets of regions. Differently, trajectories with different directions access separate values of ΔJ_{jrp} and can be therefore swept in parallel. This strategy theoretically generates N (the number of angles in the S_N quadrature formula) groups of independent tasks.

It is however important to notice that the application of the method of compound trajectories (see Sec. 5.2.1) slightly complicates the parallelization over the angles. In fact, in this method one compound trajectory may change its direction when it crosses a closed boundary. Therefore, the sweep of a trajectory with initial angle $\mathbf{\Omega}_j$ may require to access data for all the angles obtained by application of the geometrical motions to the initial angle. In other words, geometrical boundary conditions couple the sweep of all the trajectories whose initial direction is included in the set

$$S(\hat{\mathbf{\Omega}}_j) = \{g^n(\hat{\mathbf{\Omega}}_j), 0 \leq n < G\},$$

obtained by $G-1$ applications of the geometrical motion to the basic angle $\hat{\mathbf{\Omega}}_j \in S_N^{basic}$, with G being the order of the cyclic group of symmetry (g^G the identity transformation), and S_N^{basic} being the set of angles of the S_N quadrature formula contained in the basic angular domain (see Sec. 5.2.1). In the light of this, the data parallelization of the trajectory sweep can be done by grouping the trajectories with respect to the basic angles $\hat{\mathbf{\Omega}}_j$. This strategy, however, deteriorates the degree of parallelism[2], since the number of independent tasks reduces to N/G. As a practical example let us consider a typical S_N product quadrature formula with 48 azimuthal angles in $[0, 2\pi]$, and 8 polar angles in $[0, \pi]$. In a fully open case the number of independent tasks corresponds to 384, which largely suffices to parallelize the sweep operation on standard working stations. However, if we consider an infinite lattice configuration with, for example, $2\pi/8$ symmetry over the azimuthal component, and reflective boundary conditions along the axial direction, we must divide the number of directions by a factor 16, which corresponds to the number of angles connected by the symmetry conditions. This reduces the degree of parallelism to only 24 independent tasks.

[2]Degree of parallelism: maximum number of tasks that can be executed in parallel

The reduced degree of parallelism imposed by the parallelization over the basic angles can be mitigated by keeping the separation of the trajectories, and by duplicating the region-integrated net currents for each thread to avoid race conditions. This approach, however, requires additional operations at the end of the trajectory sweep in order to sum up the partial contributions of each thread. The implementation of this strategy is shown in the pseudo-code in Alg. 5. Remark the use of *private* ΔJ^q_{jrp} and the presence of the *reduction* block summing up their contribution after the trajectory sweep.

Input: Σ_r,Σ_{rn},q_{jrp},$\mathcal{T}$,S_N
Output: ϕ_{nrp}
forall the $\hat{\mathbf{\Omega}}_j \in S_n^{basic}$ **do**
  forall the *compound trajectories* $\parallel g(\hat{\mathbf{\Omega}}_j)$ **do**
    `TrajRebuild(`$\mathcal{T}_t$`) // Alg. 2;`
    $\psi^- \leftarrow \psi_{in}$; `// Boundary flux`
    for *i chords in trajectory* **do**
      `// Exiting flux`
      $\psi_+ \leftarrow$ `// Eq. (4.11);`
      `// Net currents`
      $\Delta J_{jrp} = \Delta J_{jrp} + \omega_t \Delta\psi_{ip}$ `// Eq. (4.21);`
      `// Entering flux for next chord`
      $\psi_- \leftarrow \psi_+$;
    end
  end
end
`// Region-averaged angular fluxes`
$\Psi_{jrp} \leftarrow$ `// Eq. (4.19);`
`// Update moments`
$\phi_{nrp} \leftarrow$ `// Eq. (4.14);`

Algorithm 4: Modified version of Alg. 1 showing the inversion of the fixed source problem. The algorithm also includes the details of the method of compound trajectories and of the trajectory reconstruction discussed, respectively, in Sec. 5.2.1 and Sec. 6.2. The algorithm computes the net currents for all regions (r), angles (j) and order of the polynomial expansion (p), by accumulating the contribution of each intersection, $\Delta\psi_{ip}$, into ΔJ_{jrp}. The symbol S_N^{basic} represent the directions of the S_N quadrature formula contained in the basic angular domain.

Input: Σ_r,Σ_{rn},q_{jrp},$\mathcal{T}$,S_N
Output: ϕ_{nrp}
$q \leftarrow$ `GetThreadNum();`
forall *trajectories* **in parallel do**
    `TrajRebuild(`$\mathcal{T}_t$`) // Alg. 2;`
    $\psi^- \leftarrow \psi_{in}$; `// Boundary flux`
    for *i chords in trajectory* **do**
        `// Exiting flux`
        $\psi_+ \leftarrow$ `// Eq. (4.11);`
        `// Net currents`
        $\Delta J^q_{jrp} = \Delta J^q_{jrp} + \omega_t \Delta \psi_{ip}$ `// Eq. (4.21);`
        `// Entering flux for next chord`
        $\psi_- \leftarrow \psi_+$;
    end
end
`// Reduction`

$$\Delta J_{jrp} = \sum_{q=1}^{Q} \Delta J^q_{jrp}$$

`// Region-averaged angular fluxes`
$\Psi_{jrp} \leftarrow$ `// Eq. (4.19);`
`// Update moments`
$\phi_{nrp} \leftarrow$ `// Eq. (4.14);`

Algorithm 5: Modified version of Alg. 4 in which the parallelism is done over the single trajectories and the private copies of integrated net currents are used to avoid race conditions. The reduction block is introduced to sum up the contributions of all Q threads. The *GetThreadNum* function provides the order number q of the thread.

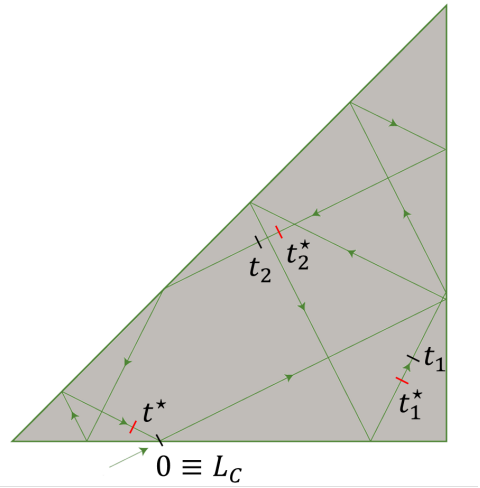


Figure 7.2: Graphical representation of the trajectory-cut. The single cyclic trajectory is divided in three sub-trajectories with entering points 0, t_1 and t_2. The flux entering each sub-trajectory is obtained by solving the transmission equation respectively from the coordinates $t^\star$, $t_1^\star$ and $t_2^\star$ (modified version of an image taken from [46].)

7.3.1 Trajectory-cut

In infinite lattice configurations it may happen that the trajectory period (see Sec. 5.2.1 and Sec. 5.2.2) becomes so large that few trajectories suffice to cover the whole problem geometry for a given basic angle. In these cases the parallelization of the sweep algorithm over the trajectories slightly contributes to the increase of the degree of parallelism. The number of independent tasks can be increased by dividing the single trajectory in multiple sub-trajectories. The entering flux of each sub-trajectory can be computed with arbitrary precision by writing the integral form of the transport equation from a point $t_{st}^\star$ of the trajectory to the point $t_{st} > t_{st}^\star$ representing the entering point of the (st)th sub-trajectory (see Fig. 7.2):

$$\psi(t_{st}) = \psi(t_{st}^\star)e^{-\tau(t_{st}^\star, t_{st})} + \int_{t_{st}^\star}^{t_{st}} dt' q(t') e^{-\tau(t', t_{st})}. \tag{7.2}$$

As already noticed in previous sections, the contribution of $\psi(t_{st}^\star)$ to $\psi(t_{st})$ can be neglected for a sufficiently large total optical length $\tau(t_{st}^\star, t_{st})$. This last equation has to be solved for each sub-trajectory generated by the

trajectory-cut, which entails that additional operations are needed for the trajectory sweep. These generally correspond to the operations required for sweeping some tens of chords, which is an acceptable cost if compared to that of the sweep of the whole trajectory ($\sim 10^5$ chords), and if we consider the enhancement of the degree of parallelism that can be obtained.

Remark that the trajectory-cut can be also applied to fine-tune the load balance among different threads. In the next sections we will see that this is fundamental to obtain an efficient parallel execution.

7.3.2 Load balance and scheduling

A drawback of the parallelization of the sweep algorithm is constituted by the non-uniform distribution of the computational cost of the sweep among the independent tasks. The variability of the cost of each task is determined by varying number of intersections composing each trajectory (see Sec. 4.4), and by the varying computational complexity of the reconstruction algorithm (see Sec. 6.2). Fig. 7.3 shows the load profile of the 9 tasks generated by grouping the independent angles of a 36×6 product quadrature formula applied to an infinite hexagonal lattice with $2\pi/12$ azimuthal symmetry and axial reflection. Similarly, Fig. 7.4 shows the profile obtained by keeping the trajectories as separate tasks. Both cases underline the typical non-homogeneous distribution of the load of the sweep algorithm. Such non-uniform distribution may determine a deterioration of the parallel performances of the algorithm, since some threads may finish to execute their tasks before the others, and become idle until all threads terminate. In this situation, the actual execution time corresponds to the time required by the most loaded thread to execute its tasks, which is clearly larger than the ideal case for which the total load is evenly distributed among the threads.

The tasks executed by each thread during the parallel execution are determined by the *scheduling strategies*. When the number of threads is smaller than the number of tasks, adapted scheduling strategies can be used to distribute the tasks in order to balance the parallel execution and improve the parallel efficiency. This will be the subject of discussion of the next two paragraphs. The first one is dedicated to the scheduling strategies available with the `OpenMP` directives, while the second paragraph is dedicated to other custom scheduling strategies that have been developed in TDT.

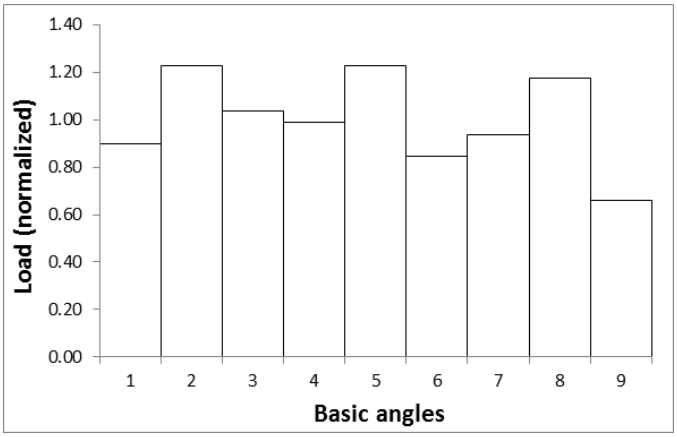


Figure 7.3: Load profile (normalized to the average load) of the tasks generated by grouping trajectories with respect to the basic tracking angle. The case corresponds to a hexagonal lattice configuration with $2\pi/12$ azimuthal symmetry and axial reflection. A 36×6 product quadrature formula is used.

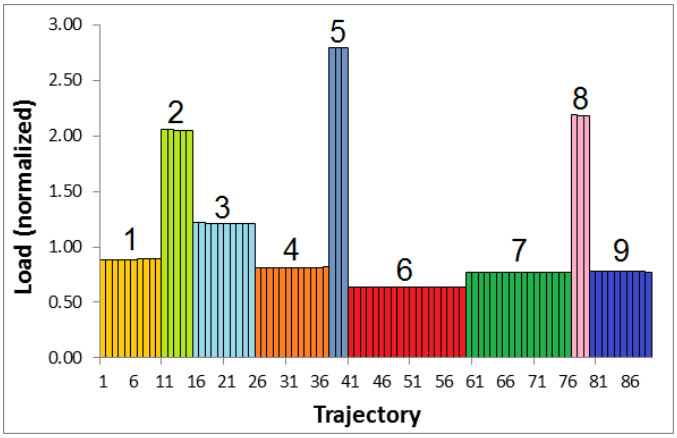


Figure 7.4: Load profile (normalized to the average load) of the same case in Fig. 7.3 obtained by keeping the trajectories as separate tasks. Different colors identify trajectories with the same angle.

OpenMP scheduling strategies

The OpenMP libraries already provide several possibilities to schedule the parallel execution of a set of tasks. The native OpenMP scheduling strategies can be mainly divided in two classes: `STATIC` and `DYNAMIC`. In the static scheduling strategies the iteration space (the set of elementary tasks) is divided in `CHUNKS` of fixed size and assigned to the threads prior to the actual execution. Each thread can therefore execute the tasks without concerning about other threads. A drawback of the static scheduling is that it does not take into account the effective execution time of the chunks. Therefore, the static strategy is not appropriate for non-uniform load profile, since some chunks may be more loaded than others (see Fig. 7.5).

Differently, in dynamic scheduling the tasks are assigned to the threads at runtime: when a thread terminates the execution of its current task, the scheduler assigns it a new task, if any. The dynamic scheduling adds an overhead to the parallel execution since the availability of tasks and threads must be determined at runtime and requires additional synchronizations between threads. This overhead increases with the number of tasks because of the larger number of synchronizations. To avoid excessive overhead OpenMP libraries allow grouping the tasks in chunks of equal size. For a large number of tasks, the degree of unbalance resulting from the dynamic scheduling is mainly due to the last executed tasks. The `GUIDED` scheduling strategy of OpenMP libraries exploits this property and divides the iteration space in chunks of non-uniform size, following a geometric series [51]. Largest tasks are the first assigned, whereas smallest tasks are used to reduce the unbalance at the end of the execution.

Self-scheduling strategies

The OpenMP scheduling strategies allow a good improvement of the parallel performance of the sweep algorithm. However, the choice of the optimal strategy often requires the tuning of the scheduling parameters such as the chunk size. In addition, none of the strategies listed above take into account the actual load profile of the tasks which can be computed, for example, at runtime. Alternatively, this can be estimated prior the execution by assuming a linear dependence with respect to the number of chords of the trajectory. Such approximation does not take into account the cost of the reconstruction, whose variation among trajectories maybe not negligible (see Sec. 6.3). A first method to include the load profile without changing the implementation of Alg. 5 is to sort trajectories for decreasing values of the

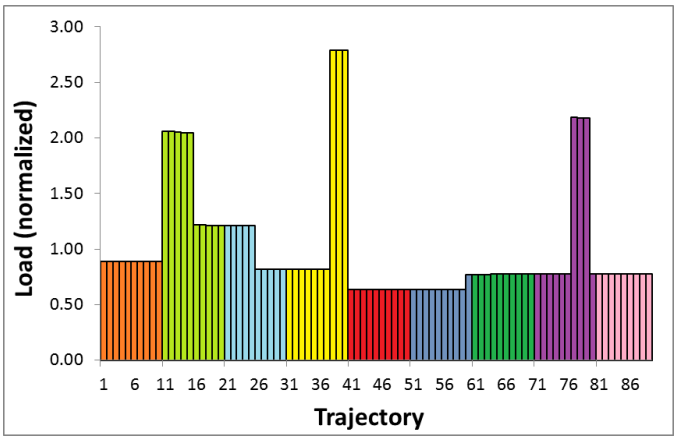


Figure 7.5: Static scheduling of the 89 tasks in Fig. 7.4 using 9 threads. The iteration space is divided in chunks containing the same number of tasks. With this strategy some threads have more work than other reducing the efficiency of the parallel execution.

load, and to use a dynamic scheduling.

To better manage the scheduling and avoid manual tuning, an alternative version of the algorithm has been implemented (see Alg. (6)). A similar technique is mentioned in [54] but little detail is provided. In our implementation the function `GetTask()` takes as input a *schedule map* $\mathcal{S}$ and the thread identifier, and it returns a contiguous chunk of the iteration space (a task) which is executed by the thread. This operation is performed by all the working threads until no tasks are left. Contrarily to Alg. 5, the new implementation allows the definition of tasks of varying size, providing an additional degree of freedom for the scheduling. Remark that the time required to sweep the trajectories depends only on the tracking. Therefore this information, as well as the schedule map, has to be computed only once, and can be reused for all iterations and for all the energy groups.

The schedule map $\mathcal{S}$ is constituted by a number of *pools* equal to the number of working threads, each pool containing the set of tasks that have to be executed. For a *static* scheduling, each thread executes the tasks contained in its pool and becomes idle when no tasks are left. Alternatively, a *dynamic* reallocation of tasks is possible: whenever a thread finishes its scheduled jobs, it is allowed to 'steal' tasks from other pools. This option requires additional synchronizations between threads to avoid the simultaneous execution of the same task, thus introducing an overhead to the

calculation.

The number of tasks contained in each pool may vary depending on the schedule strategy. For the `ONETASK` option, each pool is constituted of one task containing a number of trajectories such that the total weight of the task, $\omega(Task)$, is equal to the optimal weight computed by summing up the weights, $\omega(t)$, of the single elementary tasks of the iteration space and by dividing it by the number of threads (Q):

$$\omega_{opt} = \frac{\sum_t \omega(t)}{Q}. \tag{7.3}$$

In Fig. 7.6 we show the tasks generated by applying the `ONETASK` strategy to the iteration space in Fig. 7.4. Remark the difference between the tasks in the `ONETASK` strategy and the equivalent static strategy of the OpenMP libraries (Fig. 7.5). This strategy has the advantage to generate the minimum number of tasks, and to execute statically. However, if for any reason the load is not balanced, this strategy introduces a systematic loss of efficiency at each execution of the sweep algorithm. This might be caused, for example, by the discrete nature of the load profile, which does not allow to define an integer number of tasks having total weight equal to ω_{opt}. The trajectory-cut discussed in the previous section can be used to fine-tune the scheduling but it has not been implemented. The load unbalance of the `ONETASK` strategy can be also due to a wrong estimation of the load of each elementary task.

To avoid these effects of unbalance, the `SELFGUIDED` strategy applies a further partitioning of the tasks and a dynamic scheduling. In such method the iteration space is first divided in *macrotasks* with weights approximately equal to ω_{opt} (Eq. (7.3)). Each macrotask is assigned to a pool and is then further split in N_S tasks such that the sth task in the pool takes an $\alpha < 1$ fraction of the remaining work (see Fig. 7.7):

$$\omega_s = \omega_s^{left}\alpha, \;\; \omega_s > \omega_{min}, \tag{7.4}$$

with $\omega_1^{left} = \omega_{opt}$, and $\omega_{s+1}^{left} = \omega_s^{left} - \omega_s$. In this way, the first tasks that are executed by each thread are those with most of the load, while last tasks are much smaller and are used to balance the parallel execution thanks to their dynamic reallocation. By noticing that $\omega_s^{left} = \omega_{s-1}^{left}(1-\alpha)$, and imposing $\omega_{N_S} = \omega_{min}$, the number of tasks is:

$$N_S = 1 + \frac{\log\left(\frac{\omega_{min}}{\omega_{opt}\alpha}\right)}{\log(1-\alpha)}. \tag{7.5}$$

This expression is valid under the hypothesis of continuous load profile and assuming a real value of N_S. However, since $N_S \in \mathbb{N}$, and because of the discrete nature of the load profile, the actual number of tasks has to be adjusted during the construction of the schedule map. This number never exceeds the value $\bar{N}_S$ obtained by rounding up N_S to the next integer.

Input: Σ_r,Σ_{rn},q_{jrp},$\mathcal{T}$,S_N,$\mathcal{S}$
Output: ϕ_{nrp}
while *Job Not Done* **do**
  $q \leftarrow$ `GetThreadNum();`
  Task $\leftarrow$ `GetTask` (q,$\mathcal{S}$);
  for *trajectories in Task* **do**
    `TrajRebuild(`$\mathcal{T}_t$`) // Alg. 2;`
    $\psi^- \leftarrow \psi_{in}$; `// Boundary flux`
    for *i chords in trajectory* **do**
      `// Exiting flux`
      $\psi_+ \leftarrow$ `// Eq. (4.11);`
      `// Net currents`
      $\Delta J^q_{jrp} = \Delta J^q_{jrp} + \omega_t \Delta\psi_{ip}$ `// Eq. (4.21);`
      `// Entering flux for next chord`
      $\psi_- \leftarrow \psi_+$;
    end
  end
end
`// Reduction`

$$\Delta J_{jrp} = \sum_{q=1}^{Q} \Delta J^q_{jrp}$$

`// Region-averaged angular fluxes`
$\Psi_{jrp} \leftarrow$ `// Eq. (4.19);`
`// Update moments`
$\phi_{nrp} \leftarrow$ `// Eq. (4.14);`

Algorithm 6: Alternative version of the trajectory sweep with manual scheduling. The symbol $\mathcal{S}$ represents the schedule map of the tracking. The `GetTask()` function manage the scheduling and provide the group of trajectories that have to be swept by each thread.

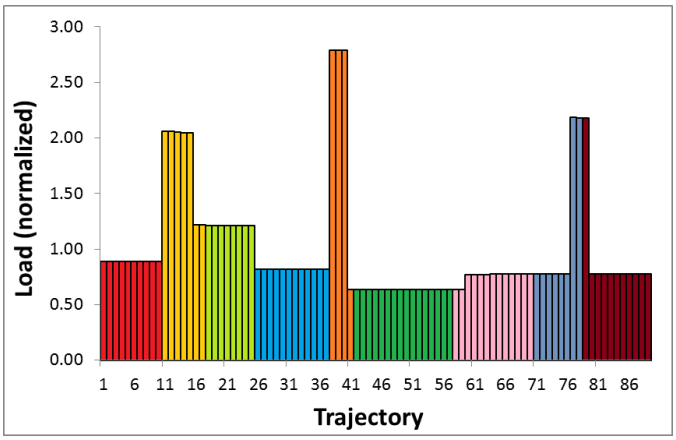


Figure 7.6: `ONETASK` scheduling of the 89 tasks in Fig. 7.4 using 9 threads. The iteration space is divided in 9 tasks having approximately the same cost. Remark the difference with the static scheduling in Fig. 7.5

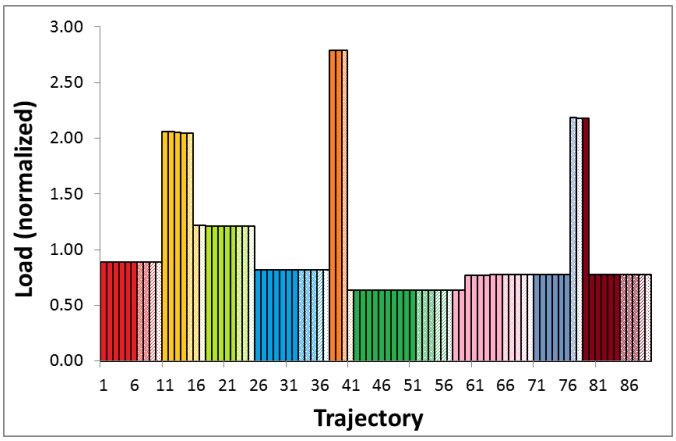


Figure 7.7: `SELFGUIDED` scheduling of the 89 tasks in Fig. 7.4 using 9 threads. The iteration space is divided in macro-tasks with approximately the same cost (different colors). Each macro-task is further split in tasks with non-uniform weight (different patterns): larger tasks are the ones that are firstly executed, while small tasks are reallocated dynamically to balance the execution.

7.3.3 Results

We tested the parallel methods of the sweep algorithm on a test case representing the assembly shown in Chapter 9. The methods have been compared using the parallel efficiency, defined as the ratio between the ideal and the actual execution time of the parallel calculation, $T_{\|}$. The ideal execution time corresponds to the time required for a serial execution, T_1, divided by the number of threads, Q, used for the parallel execution. The parallel efficiency then reads:

$$\epsilon_{\|} = \frac{T_1}{T_{\|} Q}. \tag{7.6}$$

Figure 7.8 shows the efficiency of the parallel execution of the tasks shown in Fig. 7.3, using the static and dynamic scheduling strategies provided by OpenMP libraries. Data parallelism over the basic angles is used in this case, which entails that no memory duplication is required, nor any additional operations for reduction. Tests have be done with a number of threads up to 9, which represents the maximal number of threads that can be used for such an iteration space. Results show that the efficiency of the parallelism deteriorates for increasing number of threads. This is mainly due to the load unbalance deriving from the non-uniform distribution of the load. The figure shows that there is no important difference between the use of the static and the dynamic scheduling, especially for a number of threads comparable with the number of tasks. This behavior is easily explained since the effect of the scheduling strategies becomes important only for a number of tasks sufficiently larger than the number of threads.

In Fig. 7.10 we show the parallel efficiency of the sweep algorithm for varying scheduling strategies applied to the load profile in Fig. 7.9. The figure only shows the time required for the sweep while the reduction time is neglected in order to underline the effect of scheduling strategies. Tests have been done on a a Xeon E5-2680@2.8 GHz, which is composed of 2 CPUs sharing their memory, each CPU having 10 cores. The figure shows that OpenMP `DYNAMIC`, `SELFGUIDED` and `ONETASK` have similar performances with a parallel efficiency linearly decreasing with the number of threads and reaching a value of 0.5 for 20 threads. Contrarily, the OpenMP `STATIC` strategy has a non-uniform scaling with a parallel efficiency immediately decreasing to less than 0.5 for 3 threads and reaching a value of 0.25 for 20 threads.

The linear decreasing of the parallel efficiency for the most performing strategies is due to the overheads of the parallel execution, which include memory duplication, additional memory accesses and synchronizations. The

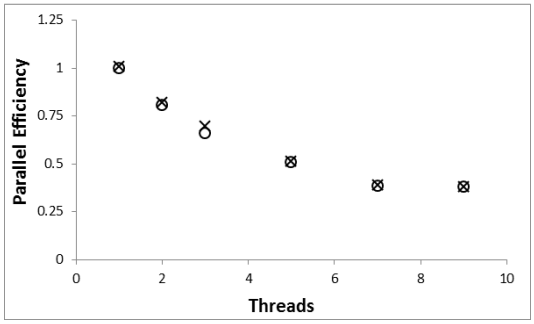


Figure 7.8: Parallel efficiency of OpenMP `STATIC` (circles) and `DYNAMIC` (crosses) strategies for the parallel execution of the case in Fig. 7.3 using data parallelism over the basic angles.

loss of efficiency from 10 to 11 threads underlines the loss due to communication latency. In the 10-threaded execution, in fact, all the cores involved in the calculation are allocated on the same CPU and have equal access time to the main memory. Differently, for a parallel execution with 11 threads, one of the cores is allocated in the other CPU for which the communication time is larger. Another effect of the memory is shown in Fig. 7.11 which underlines the effects of different memory allocation strategies: in one case the memory used by threads is allocated in the shared memory, whereas in the second case the memory is private to each thread. The second case performs better since it requires less synchronizations of the memory. The Fig. 7.12 shows the speed up of the whole internal iteration for different strategies. The best results are obtained using the `SELFGUIDED` strategy which provides a speed up close to 10 for a 20-threaded execution.

7.4 DP_N parallelism

In an accelerated MOC solution an important part of the execution time is due to the construction and solution of DP_N operator used for the synthetic problem (see Sec. 4.5). To further speed up the whole algorithm we considered the application of parallel paradigms to the DP_N operator.

As we have already anticipated in Sec. 4.5.2, the transport-coherent DP_N operator is built by using the same trajectories used for the MOC calculation. This operation has to be done for all the energy groups and can be parallelized by assigning an equal number of groups to each thread. This strategy has multiple advantages. It does not require any memory dupli-

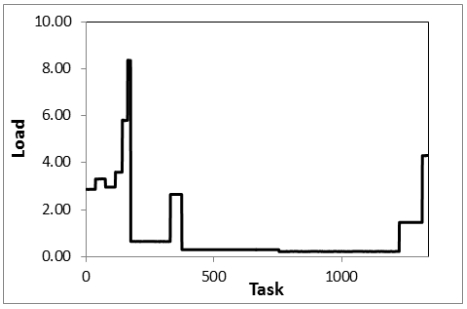


Figure 7.9: Load profile (normalized to the average task load) used for testing different scheduling strategies (Fig. 7.10). It corresponds to the case in Chapter 9.

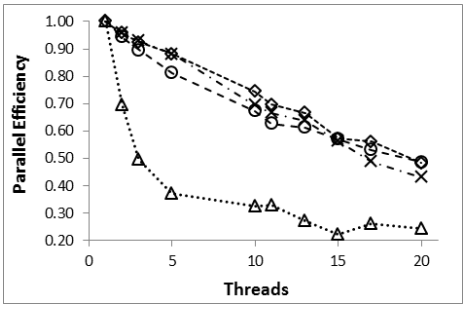


Figure 7.10: Parallel efficiency of the sweep algorithm (no reduction included) applied to the load profile in Fig. 7.9 for different scheduling strategies. Circles: OpenMP `DYNAMIC` scheduling with unitary chunk size. Diamonds: `SELFGUIDED`. Crosses: `ONETASK`. Triangles: OpenMP `STATIC`.

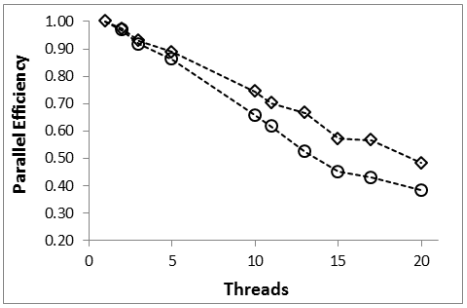


Figure 7.11: Effect of the memory on the execution time of the internal iteration. Circles indicate the efficiency by using global variables. Squares indicate the case where the memory is private to the thread (less memory synchronizations).

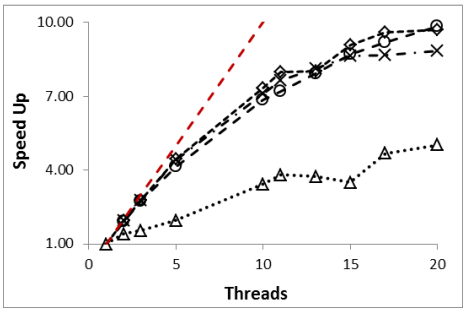


Figure 7.12: Speed up of the internal iteration corresponding to the load profile in Fig. 7.9 for different scheduling strategies. Circles: OpenMP `DYNAMIC` scheduling with unitary chunk size. Diamonds: `SELFGUIDED`. Crosses: `ONETASK`. Triangles: OpenMP `STATIC`. Dashed line: ideal scaling.

cation because all writing operations are performed on independent zones of memory. Also, this strategy produces a naturally balanced parallel execution since the cost of the construction only depends on the size of the tracking, which is equal for all the energy groups. Finally, this strategy provides a good degree of parallelism since in standard calculations the number of groups easily reaches the order of some hundreds.

The parallelization of the solution of the DP_N system is a topic that has not been addressed although it should provide a good reduction of the execution time.

7.5 Conclusions

In this chapter we have discussed several strategies for the parallelization of the Method of the Characteristics. In our research we have concentrated mostly on the parallelization of the sweep algorithm which is the most resource-demanding task of the MOC. The difficulties in applying these techniques are of different nature. From one side, the sweep algorithm accesses simultaneously to the shared values of the net currents which may introduce race conditions and invalidate the results of the algorithm. To avoid these problems we adopted two different approaches: the data parallelism over the basic tracking angles automatically avoids these problems since the parallel tasks access separate values of the net currents and allow a parallelism without additional overhead. Unfortunately, the number of independent tasks generated by the data parallelism is generally small and does not allow the execution with a large number of threads. This strategy is therefore better suited for small cases, and for multi-core architectures with a relatively small number of cores. Alternatively, the parallelization over the trajectories has been implemented to allow a higher degree of parallelism, but it requires additional memory to avoid race conditions, and additional operations for the reduction. The second major problem of the parallel execution of the sweep derives from the non-uniform distribution of the load among the tasks. Such non-uniformity may generate load unbalance between threads and deteriorate the parallel efficiency. To mitigate this effect we developed and tested different scheduling strategies. Results have shown a well-balanced distribution of the load with the parallel efficiency decreasing linearly with the number of threads due to parallel overheads. Among the scheduling strategies, the custom `SELFGUIDED` and `ONETASK` strategies have shown slightly better performances with respect to the OpenMP `DYNAMIC` scheduling. Some results of the parallel performances of TDT have been

subject of a publication in an international conference [55]

Alternative parallel strategies have been also considered but their implementation has not been done. In particular, the trajectory-cut can be used to split large tasks and increase the degree of parallelism and the efficiency of scheduling strategies.

For optically large cases (e.g., core-level calculations), an alternative domain-decomposition method has been proposed for which the effect of neighbor sub-domains is taken into account with trajectory-coherent boundary conditions that do not introduce any additional approximation on the values of boundary fluxes. The drawback of this method is that it is based on the overlapping of sub-domains and it requires additional memory (and communication time) as compared to other domain-decomposition methods for which boundary conditions have a surface-based representation.

Chapter 8

Angular quadrature formulas

The MOC uses the SN quadrature formula to compute the angular moments of the flux:

$$\Phi_n = \frac{1}{4\pi} \int_{S^2} d\mathbf{\Omega} A_n(\mathbf{\Omega}) \psi(\mathbf{\Omega}) \approx \sum_j \omega_j A_n(\mathbf{\Omega}_j) \psi(\mathbf{\Omega}_j). \tag{8.1}$$

The order of the angular quadrature formula determines the number of directions required to approximate Eq. (8.1) and, by consequence, the number of trajectories that have to be tracked and swept. The choice of the quadrature set is therefore important in determining both memory and time requirements of the MOC algorithm.

In general, quadrature formulas are built to exactly integrate a finite set of functions used as representation basis of the integrand function. A quadrature formula is then more accurate if the integrand function can be expanded into this basis with a small residual.

In the case of neutron transport, numerous papers are dedicated to the research of optimal quadrature formulas. Clearly these formulas depend on the problem considered. Some quadrature sets, for example, are built in order to reduce the so-called ray-effect, which mostly affects problems with highly absorbing media and localized sources [13, 14]. Other quadrature sets are built in order to preserve the invariance of the solution to rotations and symmetries, such as the Level Symmetric quadrature set [9], or quadrature sets containing the symmetries of platonic solids. The most used quadrature sets for 2D transport calculations with the MOC are product-type quadrature formulas, for which the representation basis of the direction $\mathbf{\Omega}$ is obtained by the tensor product of two mono-dimensional bases of the azimuthal and polar angles. Generally, a uniform rectangle formula is used for the azimuthal component. This formula is characterized by constant weights and

uniformly distributed nodes, and is optimized to integrate functions that are periodic over $\varphi \in [0, 2\pi]$. Concerning the polar variable, the usual choice is a high-order quadrature formula such as the Gauss-Legendre formula [8]. This choice is justified by the fact in 2D cases the angular flux is a smooth function of the polar variable. A very successful approach is the one firstly introduced in [56], and then generalized in [46], which computes the weights and nodes of the polar quadrature formula by minimizing the integration error of Bickley-Naylor functions. These functions are the exact solution of 2D transport in a medium with no scattering for a line source [46, 57].

In 3D axial geometries, a separation of the azimuthal and the polar variables is a natural choice due to the cylindrical structure of the geometry. For this reason we limit our research only to product-type quadrature formulas. In the next section, we will briefly recall the construction of product-type quadrature sets, and we will provide details of the polar formulas that have been implemented in TDT. A particular attention is given to high-order polar sets that include directions $\theta = 0$, $\theta = \pi$, or $\theta = \pi/2$, which correspond to the *vertical* and *horizontal* directions. These 'special' directions are particularly attractive from the computational point of view since the cost of their sweep is negligible as compared to that of other 'standard' directions. In Sec. 8.2 we provide the details of the specialized sweep algorithms for the horizontal and vertical directions for the step-MOC approximation. The generalization of these methods to higher order expansions of the source has not been carried out. The last section of this chapter is dedicated to convergence tests of these quadrature formulas in typical reactor configurations.

8.1 Product-type formulas

Product-type formulas approximate two-dimensional integrals using two separates mono-dimensional quadrature sets for the integration variables. We define then the azimuthal quadrature set of order M,

$$S_M^{\varphi} = \{(\omega_i^{\varphi}, \varphi_i), 1 \leq i \leq M\},$$

as the set of weights, ω_i^{φ}, and nodes, φ_i, approximating the integral of a generic function of the azimuthal variable, $f(\varphi)$:

$$\frac{1}{2\pi} \int_{(2\pi)} d\varphi f(\varphi) \approx \sum_{i=1}^{M} \omega_i^{\varphi} f(\varphi_i). \tag{8.2}$$

With a similar notation, we define the polar quadrature set of order N, S_N^θ, by the approximating integral:

$$\frac{1}{2}\int_0^\pi d\theta \sin(\theta) g(\theta) \approx \sum_{j=1}^{N} \omega_j^\theta g(\theta_j). \tag{8.3}$$

The product quadrature formula of order $M \times N$ for the two-dimensional variable $\mathbf{\Omega}$ used in Eq. (8.1) is obtained by taking the tensor product of the two mono-dimensional quadrature sets S_M^φ and S_N^θ. The resulting nodes and weights are respectively $\mathbf{\Omega}_{ij} = (\varphi_i, \theta_j)$ and $\omega_{ij} = \omega^\varphi \omega_j^\theta$:

$$S_{M\times N} = S_M^\varphi \otimes S_N^\theta = \{(\mathbf{\Omega}_{ij}, \omega_{ij}); 1 \le i \le M, 1 \le j \le N\}. \tag{8.4}$$

When the polar quadrature formula includes the endpoints of the integration interval (subscript v), the directions $\mathbf{\Omega}_{iv}$ coincides with the vertical directions (up/down). Remark from the definition in Eq. (8.4), and by using the normalization condition of the azimuthal formula (Eq. (8.2)), that the weight associated to the vertical direction coincides with ω_v^θ.

Reciprocity and symmetries

The construction of the angular quadrature set must obey to the reciprocity condition in order to preserve the forward/backward symmetry of the Transport Equation in a pure absorbing medium. This is done by defining the quadrature formula only for the directions in the upper unit sphere and by extending it to the whole set by imposing the reciprocity. This requires that if $\mathbf{\Omega} = (\varphi, \theta) \in S_N$ than also $-\mathbf{\Omega} = (\varphi + \pi, \pi - \theta)$ is in the quadrature set. The symmetry also requires the weights associated to opposite directions to be equal.

In the TDT implementation, the construction of the azimuthal quadrature formula is done in for the angles $\varphi \in [0, \pi]$ and then extended to the whole azimuthal space by imposing reciprocity to obtain directions $\varphi + \pi$. When the geometry is characterized by axial symmetries, the azimuthal formula is built in the basic angular domain, $\Delta\varphi_b$, and then extended to $[0, \pi]$ by repeated application of the geometrical motions (see Sec. 5.2). Concerning the polar quadrature set, we always impose the symmetry of the quadrature formula with respect to the horizontal direction ($\theta = \pi/2$) to obtain the direction $\pi - \theta$ required by reciprocity. This condition already takes into account both reflective and translational boundary conditions on the horizontal planes and does not require additional restrictions.

8.1.1 Polar quadrature formulas

We focused our research on the convergence analysis of different types of polar quadrature formulas. These quadrature sets are built in order to satisfy Eq. (8.3). For some quadrature sets, it is useful to rewrite this last integral applying the change of variables $\mu = \cos(\theta) \in [-1, 1]$, which gives:

$$\frac{1}{2}\int_{-1}^{1} d\mu g(\mu) \approx \sum_{j=1}^{N} \omega_j^{\theta} g(\mu_j). \tag{8.5}$$

Uniform step and trapezoidal sets

These are quadratures belonging to the family of Newton-Cotes formulas, for which the integrand function is approximated with interpolation rules [58]. For these kind of formulas the integration interval is divided into bins of equal size, and, for each bin, a constant (step) or a linear (trapezoidal) representation of the integrand is used. In the **UT** set (Uniform Theta) the integrand is assumed to be constant within the bins of size $\Delta\theta = \pi/N$, with N the order of the quadrature set. The nodes are chosen with the mid-point rule, whereas the weights are computed by substituting the step approximation of the integrand in Eq. (8.3):

$$\omega_j^{UT} = \frac{1}{2}\{\cos[j\Delta\theta] - \cos[(j-1)\Delta\theta]\}. \tag{8.6}$$

The **UM** (Uniform Mu) applies the same approximation to Eq. (8.5) to represent the variation of the integrand within the bins of size $\Delta\mu = 2/N$. The weights of the UM formula are constant and equal to $\omega_j^{UM} = 1/N$. The **TM** and **TT** sets are built using a linear interpolation rule between the values of the integrand evaluated at the $N+1$ endpoints of the intervals $\Delta\mu$ (TM set), and $\Delta\theta$ (TT set). In the first case, the weights correspond to the standard trapezoidal rule [58], whereas for the case of the **TT** formula they assume the following expression:

$$\omega_j^{TT} = \frac{1}{2\Delta\theta}\begin{cases} \Delta\theta - \sin(\Delta\theta), & j = 0, j = N \\ 2\sin(\theta_j) - \sin(\theta_{j-1}) - \sin(\theta_{j+1}), & 1 \leq j \leq N-1 \end{cases} \tag{8.7}$$

All these quadrature sets are symmetric with respect to the horizontal direction, as required by the reciprocity constraints. For odd values of N the UM and UT sets contain the horizontal direction, whereas for trapezoidal rules the horizontal direction is included for even values of N. Remark that trapezoidal rules also contain the vertical direction.

Gauss-Legendre and Gauss-Lobatto sets

The polar integral in Eq. (8.5) can be approximated using Gauss-Legendre formulas (**GL**). These are well known formulas which guarantee an exact integration of polynomials of the integration variable (i.e., μ) up to order $2N-1$, with N the order of the quadrature set. Hence, GL formulas are optimal to integrate regular functions, since these are well approximated by the polynomial basis.

The Gauss-Lobatto (**GLob**) quadrature set is a modified version of the GL set which contains also the endpoints of the integration interval [58]. These formulas have a reduced precision with respect to the classic GL set, since they guarantee exact integration only for polynomials up to order $2N-3$. However, the endpoints of the integration integral correspond to the vertical directions, for which the cost of the solution of the transport equation is negligible, as compared to that of 'internal' points (see Sec. 8.2). A 'fair' comparison of the two sets can be done by considering only the internal points of the two quadrature sets. With this adjustment, the actual order of the GLob set increases to $2(N+2)-3$, making it theoretically advantageous with respect to the GL set.

Both GL and GLob sets satisfy the symmetry with respect to $\theta = \pi/2$. Also, both sets contain the horizontal direction for odd values of N.

Clenshaw-Curtis set

In the Clenshaw-Curtis (**CC**) formula the integral in Eq. (8.3) is solved analytically by assuming a truncated cosine expansion of the integrand, while the coefficients of this expansion are approximated using a trapezoidal formula [59]. These quadrature sets show very good convergence rate and outperform GL sets for periodic irregular functions [60]. The nodes of the CC set are those of the TT formula and are, therefore, evenly distributed in $[0,\pi]$, and symmetric with respect to $\theta = \pi/2$. They also include the vertical and, for even orders, the horizontal directions.

Minimized Bickley set

MB formulas [46] are built by minimizing the integration error with respect to the Bickley-Naylor function of third order:

$$Ki_3(\tau) = \int_0^{\pi/2} d\theta \sin^2\theta \exp\left(-\frac{\tau}{\sin\theta}\right). \tag{8.8}$$

The Bickley-Naylor function represents the exact solution of the neutron transport equation for a line source along the axial direction and no scattering [57]. In Eq. (8.8) τ is the projection of the optical length on the xy-plane.

The quadrature set is obtained by non-linear minimization of the functional [46]:

$$F(\vec{\omega},\vec{\theta};\tau_{max}) = \int_0^{\tau_{max}} \left[\frac{2}{\pi} Ki_3(\tau) - \frac{\sum_j \omega_j \sin^2\theta_j \exp\left(-\frac{\tau}{\sin\theta_j}\right)}{\sum_j \omega_j}\right]^2 d\tau, \quad (8.9)$$

representing the integration error introduced by using the quadrature formula composed of the weights ω_j and nodes θ_j so that $\theta_j \in [0,\pi/2]$ and $\sum_j \omega_j = 1$. In Eq. (8.9) τ_{max} has a fixed value (e.g., the average/maximal 2D chord). Remark that the definition of the functional in Eq. (8.9) is invariant with respect to the normalization chosen for the weights. Therefore, its minimization can be done without considering the condition $\sum_j \omega_j = 1$. This condition is imposed only once the optimal set is found, by applying the same scaling factor, $1/\sum_j \omega_j$, to the set of optimal weights.

The minimization of the functional is done using the *gradient descent* algorithm [61] for which the minimum is sought following the negative direction of the gradient of the functional with respect to the free parameters ω_k and θ_k:

$$-\frac{\partial F}{\partial \omega_k} = \int_0^{\tau_{max}} \frac{K(\vec{\omega},\vec{\theta},\tau)}{\left(\sum_j \omega_j\right)^2} \left[\sin^2\theta_k \exp\left(-\frac{\tau}{\sin\theta_k}\right) \sum_j \omega_j + \right.$$
$$\left. - \sum_j \omega_j \sin^2\theta_j \exp\left(-\frac{\tau}{\sin\theta_j}\right)\right] d\tau, \quad (8.10)$$

$$-\frac{\partial F}{\partial \theta_k} = \int_0^{\tau_{max}} \frac{K(\vec{\omega},\vec{\theta},\tau)}{\sum_j \omega_j} \omega_k \cos\theta_k (\tau + 2\sin\theta_k) \exp\left(-\frac{\tau}{\sin\theta_k}\right) d\tau, \quad (8.11)$$

where the following definition has been used for the K function:

$$K(\vec{\omega},\vec{\theta},\tau) = 2\left[\frac{2}{\pi} Ki_3(\tau) - \frac{\sum_j \omega_j \sin^2\theta_j \exp\left(-\frac{\tau}{\sin\theta_j}\right)}{\sum_j \omega_j}\right]. \quad (8.12)$$

In the standard approach the minimization is done by assuming $2N$ free parameters representing the weights and the nodes of the quadrature set. The minimization algorithm can be modified in order to impose the existence of particular directions in the quadrature set. This allows, for example, to build quadrature sets containing the vertical and the horizontal directions. In these cases, the special directions are fixed parameter whereas only their associated weights are used for the minimization.

Since the MB formula is sought only for values of $\theta \in [0, \pi/2]$, an extension to the whole polar integration interval is required. This is done by imposing the symmetry of the quadrature set with respect to the horizontal direction. Remark that, if the horizontal direction is included, its weight must be doubled to take into account the up/down symmetry.

8.2 Step approximation for special directions

In axial geometries horizontal and vertical directions are particularly attractive in terms of cost of the sweep. Vertical directions are those that are parallel to the axial mesh, $\vec{H}$ (see Sec. 3.2). Geometrically speaking, all vertical trajectories are equals, and the set of intersection within the domain clearly coincides with the axial mesh. This allows the treatment of the vertical direction without additional memory. The leakage term in Eq. (4.21) for the step approximation is exactly integrated if

$$\Delta J_{rv} = A_{r_{2D}} \Delta \psi_{rv}, \tag{8.13}$$

where $A_{r_{2D}}$ is the area of the 2D region, and ψ_{rv} is obtained by sweeping only one vertical trajectory for each 2D region in the upward and downward directions. When geometrical boundary conditions are applied, we only have to consider those applied on the horizontal boundaries. In some special cases (double reflection or translation), vertical sweep can be done with periodic trajectories even if the geometry is not completely closed. The cost of the algorithm corresponds to the operations needed to solve the transmission equation for $N_Z \times N_{2D}$ chords, N_Z being the number of subdivisions along the axial direction, and N_{2D} the number of regions of the 2D section.

Horizontal trajectories are those parallel to the 2D section. In axial geometries a translation along the axial direction of a horizontal trajectory does not change the number of intersections within the 3D domain. The sequence of chord lengths and crossed regions on the 2D mesh are also invariant. Remark also that horizontal trajectories coincide with the 2D trajectories used for the construction (and reconstruction) of the actual 3D

trajectories. This implies that no additional memory is required to store trajectories for the horizontal directions. In terms of contribution to the leakage term in the region-balance, parallel horizontal trajectories within the same axial mesh are equals, which allows to simplify the leakage term with the following:

$$\Delta J_{rh} \approx \sum_{\substack{t \| \mathbf{\Omega}_h \\ t \in \mathcal{D}_r}} \Delta_t^{2D} \Delta \psi_t, \tag{8.14}$$

where $\Delta_t^{2D} = \Delta h_{r_z} \Delta r$ is the weight of the associated to 2D directions, which correspond to the height of the axial node, Δh_{r_z}, times the spacing, Δr, between 2D trajectories. By using this simplified equation, the leakage term for the horizontal direction can be obtained by sweeping the set of 2D trajectories for each axial node. As for vertical trajectories, an exact treatment of geometrical boundary conditions can be done by taking into account the conditions applied on the vertical surfaces. If all the vertical surfaces are characterized by geometrical boundary conditions, then the treatment of the horizontal direction is done using periodic trajectories.

8.3 Results of the convergence analysis

The quadrature formulas implemented in TDT have been tested on heterogeneous cases which can be typically encountered in actual reactor configurations. These analyses have been carried out by considering the error committed in the estimation of the k_{eff} by employing a given quadrature set, as compared to a reference value. For an actual validation of the solver, a true reference value should be used, such as the value provided by a Monte Carlo simulation [62]. However, in this section we are only concerned with the errors introduced by employing a quadrature formula for the estimation of the angular integrals. Is therefore out of scope to compare the performances of quadrature formulas using the Monte Carlo simulation. Instead, the comparisons are done by assuming that all quadrature sets converge to the same value, and by taking this value as a reference for the error estimate. In particular, reference values are computed using a GL set of high order (~ 30) for the polar integration. In order to single out only the effects of the polar integration, the comparisons are done by keeping the same azimuthal formula and by only varying the type and the degree of the polar set. The azimuthal quadrature formula employed is a uniform quadrature set with equally spaced nodes, which is the common choice for 2D MOC calculations [46]. Some of these results have been subject of a publication

in a international conference [63] and in an international journal [45].

The first case considered is a three-dimensional fuel cell in an infinite lattice configuration whose elementary geometry is described in Fig. 8.1. This case is characterized by homogeneous axial composition of the clad and of the moderator, and a heterogeneous composition of the fuel. The latter is constituted of enriched Uranium oxides (Uox) in its lower part, and by a mixture of depleted Uranium and other fissile isotopes (Mox) in the upper part. Since both Uox and Mox have similar nuclear properties, the presence of the Uox/Mox heterogeneity mildly perturbs a problem which, otherwise, would be two-dimensional. We expect therefore convergence rates similar to those obtained in 2D cases, for which the most performing sets are GL and MB [46]. This reasoning is confirmed by the numerical results shown in Figs. 8.2(a) and 8.2(b), which show the error on the reactivity for several quadrature formulas. To make a fair comparison between formulas with vertical and horizontal directions, results are plotted with respect to the number of 3D angles in the polar quadrature formula, that is by not counting these special directions. Results show that, in general, high order quadrature formulas perform better than Newton-Cotes formulas, a fact that is explained by the regularity of the angular flux. A surprising result concerns the convergence rate of the GLob set. In fact, as we have seen in the previous section, this quadrature set theoretically integrates polynomials up to order $2N+1$, if N is the number of 3D angles. Conversely, an equivalent GL quadrature exactly integrates polynomials up to the order $2N-1$, which makes it less advantageous from the computational point of view. However, this theoretical result is not confirmed by the numerical simulations which show that far more angles are required for the GLob set to attain an accuracy similar to that of the GL quadrature. In Fig. 8.3(a) we compare the error on the reactivity of GL and GLob sets with respect to the maximum order of the integrating polynomial. The figure shows that, although both sets exhibit a similar convergence rate, the presence of the vertical direction in the GLob set introduces a bias which distances the solution from the converged value. A similar bias is found whenever the horizontal direction is included in the quadrature set, as it can be seen by comparing the convergence series of odd and even orders of the same quadrature set (see Figs. 8.3(a) and 8.3(b)). The nature of this bias has not been understood and requires more research.

In the second configuration, the 2D section of the geometry is the triangular domain in Fig. 8.4, representing the basic domain of an infinite hexagonal lattice configuration with $\pi/6$ symmetry. The axial composition of materials is homogeneous for the clad and for the coolant, whereas the fuel is composed of stacked fissile and fertile materials. Differently from

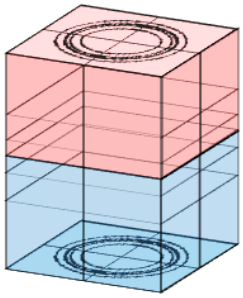
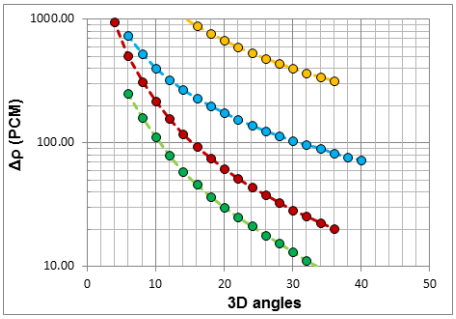
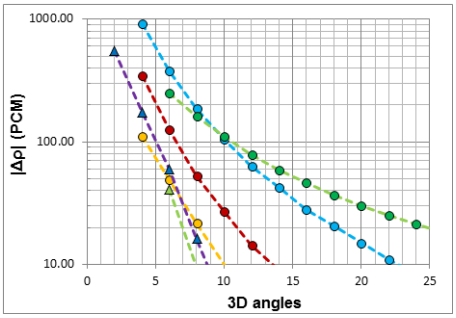


Figure 8.1: Geometrical description of the Uox/Mox pincell (Uox: lower part, Mox: upper part).

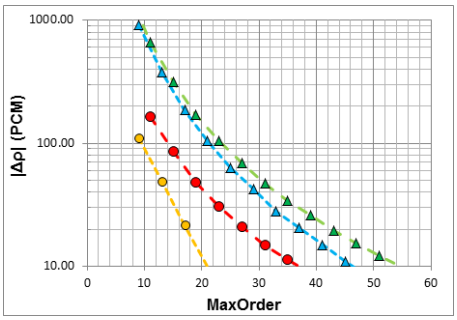


(a) Newton-Cotes formulas. Green: UT set; red: TT set of even order (with vertical and horizontal directions); cyan: UM; yellow: TM set of even order (with horizontal and vertical directions).

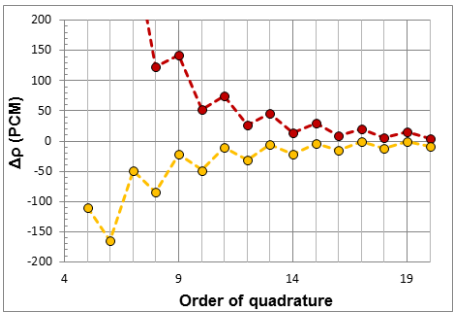


(b) Cyan circles: GLob of even order (with vertical direction); red circles: CC of odd order (with vertical direction); violet triangles: MB set (with horizontal and vertical directions); green triangles: MB set; yellow circles: GL set of odd order (with horizontal direction); green circles: UT set.

Figure 8.2: Convergence series of quadrature formulas for the case in Fig. 8.1. Fig. (a) shows Newton-Cotes formulas: sets with uniformly distributed θ angles converge faster than those with uniformly distributed μ. Fig. (b) shows the convergence of high order formulas. The UT set (green circles) is shown in both figures as reference.



(a) Convergence series with respect to the theoretical order of Gaussian formulas. Circles: GL sets of odd (yellow) and even (red) order. Triangles: GLob sets of odd (green) and even (cyan) order.



(b) Convergence series of GL (yellow) and GC (red) sets with respect to the order of the polar quadrature. The horizontal direction belongs to the quadrature for odd orders of the GL set, and for even orders of the CC set.

Figure 8.3: A bias appears when the vertical and horizontal directions are considered. The comparison between the GL and GLob sets in Fig. (a) shows the bias introduced by the vertical direction. Fig. (b) underlines the effect of the horizontal direction. An alternate convergence series is obtained by switching between even and odd orders of the quadrature formulas.

the previous case, the heterogeneity along the axial direction constitutes a strong perturbation of the angular flux, since the nuclear properties of the fertile and fissile mixtures are substantially different. In such configuration, the majority of neutrons is emitted by the fissions produced in the lower part of the fuel, whereas the fertile zone behaves more as an absorbent. In Fig. 8.5 we show the error on the reactivity for some polar quadrature sets. The behavior of quadrature formulas is similar to that exhibited for the previous case: GL sets are the most accurate sets, followed by the CC set, whereas the uniform set converges more slowly.

An important remark is due for the MB set. We cross-compare Fig. 8.2(b) and Fig. 8.5 by taking the GL set as reference. This comparison shows that the MB set exhibits a good convergence rate in the case of Uox/Mox transition, whereas its accuracy deteriorates in the fissile/fertile case. The reason of such deterioration derives from the fact that MB sets are built by minimizing the integration error with respect to the two-dimensional solution. This is an acceptable approximation for mild axial heterogeneities but it breaks down if strong axial heterogeneities are introduced.

Finally, the Fig. 8.7 shows the convergence rates and errors of GL, CC,

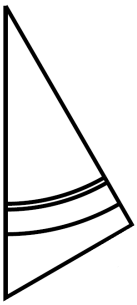


Figure 8.4: Triangular pincell representing the basic 2D section of an infinite hexagonal lattice configuration. Along the axial direction the composition of the fuel varies: fissile material composes the lower part, while fertile material composes the upper part.

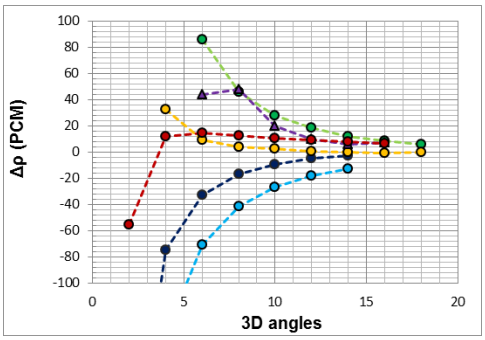


Figure 8.5: Converge series of quadrature formulas for case in Fig. 8.4. Yellow circles: GL set of even order; red circles: GL set of odd order (with horizontal); violet triangles: MB set; green circles: UT set; navy circles: CC set of odd order (with vertical); cyan circles: GLob of even order (with vertical).

GLob, and MB formulas for the hexagonal assembly whose 2D section is shown in Fig. 8.6. The axial composition of the fuel materials is the same of the previous case (Fig. 8.4) while other material are axially homogeneous. The figure shows that quadrature formulas in this case behave similarly as the previous case.

8.4 Conclusions

The convergence analysis done in this chapter has shown that product-type quadrature formulas with low order azimuthal quadrature formulas and high order polar quadrature sets are an optimal choice for typical reactor configurations. In particular, the classical Gauss-Legendre sets have shown the best results: a polar formula of sixth order generally suffices to obtain a converged solution. The Clenshaw-Curtis formula has also shown good convergence rate and might be appropriate for cases with more severe heterogeneities. Concerning the Minimized Bickley, results show that the accuracy of the formula deteriorates in cases with strong axial heterogeneities.

In this chapter we have also explored the convergence rate of quadrature formulas containing the vertical and horizontal directions. We explored

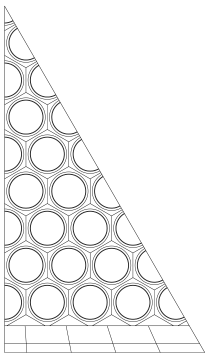


Figure 8.6: Section of an assembly in a infinite hexagonal lattice configuration. Axial composition of fuel as for Fig. 8.4.

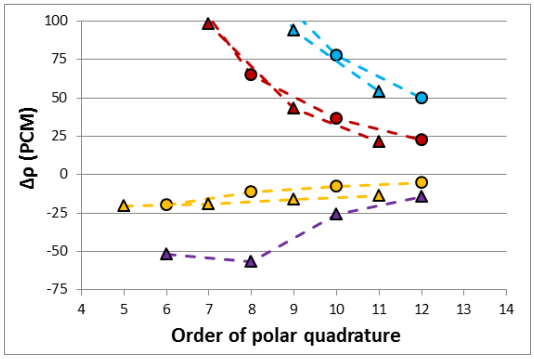


Figure 8.7: Converge series of quadrature formulas for case in Fig. 8.6. Yellow: GL set; violet: MB set; red: CC set; cyan: GLob.

these formulas because the computational cost of the associated sweep is negligible, as compared to that of the '3D angles'. The numerical results show that these directions introduce a bias in the solution which has not been explained yet. Further research is required on this topic.

Part III

Application

Chapter 9

Application and verification of the MOC

The methods discussed throughout this manuscript have been implemented in the APOLLO3® version of TDT, developed at the Laboratoire de Transport Stochastique et Déterministe (LTSD) of the CEA of Saclay. In this chapter we propose an actual application of the solver with the aim, on the one hand, of verifying its correct implementation, and, on the other hand, of showing its usefulness for the design of innovative reactor designs. This study has been done in collaboration with the Service de Physique des Réacteurs et du Cycle (SPRC) of the CEA of Cadarache, which is in charge of defining a calculation scheme for the ASTRID project.

The chosen case is a heterogeneous three-dimensional case representing one assembly of the innovative Fast Breeder Reactor ASTRID, developed at the CEA in France. The verification has been carried out by comparing the APOLLO3® results with a reference Monte Carlo calculation obtained with the TRIPOLI4 code, also developed at the LTSD.

This chapter is structured as follows. In the first section we will briefly describe the ASTRID project, its goals and the design solutions proposed to achieve these goals. We will then recall the two-levels scheme classically used for core analysis and we will then provide the details of the model used for the assembly-level calculation. An important point for the correct calculation of the case proposed is the generation of the multi-group cross sections, which is done with the self-shielding model. The coupling between the MOC and the self-shielding will be subject of discussion of Sec. 9.3.1. The results and discussion of the APOLLO3®/Tripoli4 comparisons will be provided in the last sections of this chapter.

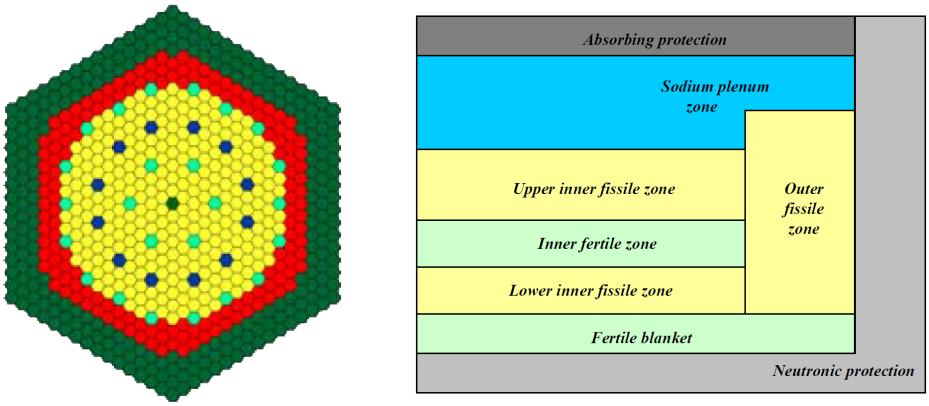
9.1 ASTRID: a Gen IV Sodium-cooled fast reactor

Fast reactors differ from the standard thermal reactors mainly for their neutron energy spectrum, which is characterized by a larger population of highly energetic neutrons ($\sim 1MeV$). The objective of fast reactors is to use fast neutrons to transmute the hardly fissile $^{238}_{92}U$ isotope in $^{239}_{94}Pu$ which, instead, can be easily fissioned. The goal of such design is clearly to better exploit the primary source (precisely Uranium) to keep the economic feasibility of nuclear power. The fast energy spectrum also facilitates the transmutation of minor actinides in order to reduce the activation and decay time of spent fuel.

As a drawback, fast reactors are more concerned about stability of the nuclear reaction as compared to thermal reactor designs. In particular, old fast reactor designs suffered of the so-called *void effect*: a positive reactivity insertion (positive feedback on the chain reaction) in case of reduction of the coolant density. This effect led to a reduced operability of the reactor due to safety reasons.

ASTRID is a project of innovative Sodium-cooled Fast Breeder Reactor led by the CEA [64]. It is part of the Gen IV reactors whose objectives are the improvement of the safety and operability of nuclear reactors, as well as the improvement of the fuel-cycle for durable and economic solutions. One of the goals of ASTRID is the reduction of the above-noted void effect without compromising the performances of the reactor in terms of power density and burnup [65]. The solution proposed in ASTRID is the introduction of axial heterogeneities in order to maximize the neutron leakage from the fissile zones in case of reduction of the the Sodium density due, for example, to a temperature increase. An axial cut of the ASTRID core is shown in Fig. 9.1(b), while Fig. 9.1(a) shows its two-dimensional section. Remark the presence of the large Sodium plenum at the top of the reactor, and the absorbing protection behind him. In nominal condition, the plenum acts as a reflector and keeps neutrons inside the core, stabilizing the chain reaction. In 'accidental' transients, the temperature of the plenum increases determining a reduction of its density. In these conditions the plenum partially loses its reflective properties, increasing the leakages towards the absorbing neutron protection, thus reducing the reactivity insertion. The fertile layers increase the curvature of the flux at the upper fissile/plenum interface, further increasing the neutron leakage.

This solution totally differs from old reactor designs which are charac-



(a) Two-dimensional section. (b) Axial section.

Figure 9.1: Geometry of the ASTRID reactor.

terized by a more homogeneous axial composition. In these problems the classic two-dimensional approximation of the transport may introduce substantial errors and it should be avoided, or at least carefully tested with respect to a 3D reference calculation.

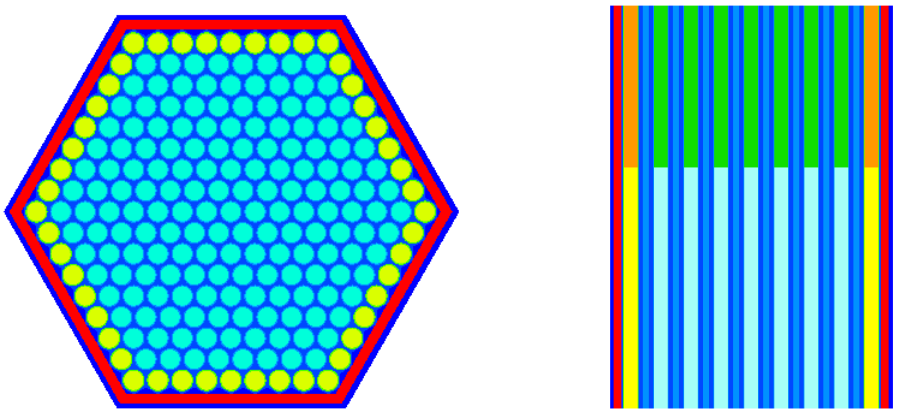
9.2 Two-level core analysis

The two-level approach is the classic method used for reactor analysis. In consists in studying the neutronic behavior of the reactor at two different scales:

Lattice level ($O(10cm)$) at this level the material heterogeneities introduce transport effects that have to be correctly accounted for. A fine-energy ($O(10^2) \rightarrow O(10^3)$ groups) transport solution with accurate geometry description and approximate boundary conditions (infinite lattice) are used at this level.

Core level ($O(1m)$) at this scale transport effects become less important and the neutron distribution assumes a diffusive behavior. Generally a coarse-group ($O(10)$ groups), coarse-mesh transport calculation of the whole core is done [66].

The homogeneous parameters used at the core level are computed through flux-weighted averages except for the rodded assembly for which an equivalence in reactivity is performed [67].



(a) Two-dimensional section. (b) Axial section.

Figure 9.2: Geometry of the CFV assembly. Different colors represent the coarse mesh used for the two-level scheme. Radially, the last row of fuel pins is kept separated from the inner pins. Axially, the fertile (upper) and the fissile (lower) zones are also separated.

9.2.1 Lattice calculation of a FBR assembly

The problem considered is representative of an assembly of the ASTRID CFV (French acronym for Low Void Effect Core) core. It is constituted by an infinite lattice configuration of a prism whose two-dimensional section is shown in Fig. 9.2(a). The axial heterogeneities have been modeled by assuming an infinite stack of layers of fertile and fissile materials with reflective boundary conditions. Fig. 9.2(b) shows an axial cut of the assembly.

A detailed solution of this problem has been obtained in order to apply the two-level equivalence and to compute the diffusive parameters used at the core level. The coarse mesh used for the energy representation is a 33 groups energy mesh. Spatially, the homogenization is done by separating the fuel in two zones: the peripheral zone including the last row of fuel pins, and the internal zone which includes the other fuel pins (different colors in Fig. 9.2(a)). Axially, values are homogenized by separating the fissile and the fertile zones as shown in Fig. 9.2(b).

Monte Carlo reference solution

The verification of the APOLLO3® scheme has been done by making comparisons with a continuous-in-energy Monte Carlo reference solution com-

puted with Tripoli4 [62]. The Monte Carlo method uses a statistical approach to simulate the path traveled by the neutrons and their interaction with the matter (e.g., fission reaction, scattering, etc.) without making any approximation of the physics of the interaction and of the geometry. About one week calculation on 6-cores Intel Xeon X5650 has been required to obtain a solution with acceptable statistical errors ($O(10^{-5})$ on reactivity and $O(10^{-3})$ for the energy and spatial distributions of reaction rates).

9.3 APOLLO3® numerical scheme of the FBR assembly

The detailed solution of the CFV assembly has been obtained using the 3D MOC solver of TDT developed throughout our research. The ECCO 1968 groups discretization of the energy is used for the multi-group calculation [68]. This energy mesh is conceived to correctly take into account the resonance effects for the whole energy domain (up to $20MeV$). The multi-group cross sections have been computed using the Subgroups method (see Sec. 2.2) of the AUTOP module of APOLLO3®. This method uses the 2D Collision Probability [7] solver of TDT to invert the transport operator for the quadrature points of the Probability Tables (see Sec. 2.1) in a fixed source problem. To take into account three-dimensional effects, the fixed source of the 2D Subgroups method is directly provided by the 3D MOC solution applying a 2D/3D equivalence. The mutual coupling between the MOC and the Subgroups solutions, and the nonlinearities introduced by the multi-group equivalence (see Sec. 2.2) require an iterative solution to converge on both multi-group fluxes and cross sections. The results of this calculation have been subject of a publication in an international conference [55].

9.3.1 Self-shielding spatial equivalence

The equivalence between the two-dimensional Subgroup method and the 3D MOC is done by partitioning the 3D geometry in N_S axial nodes (see Fig. 9.3(b)) and by integrating the slowing down equation (Eq. (2.17)) along the axial direction. To perform this operation we assume a homogeneous axial composition of the materials within each node. We obtain N_S modified equations which differ from the original in the definition of the external source:

$$q_{ext}^{\zeta}(\mathbf{r}_{\perp}) = \int_{\Delta z^{\zeta}} dz q_{ext}(\mathbf{r}) - \frac{\Delta J_g^{\zeta}(\mathbf{r}_{\perp})}{\Delta E_g}. \tag{9.1}$$

The first term in this equation is the integral of the external source in the ζth axial node of height Δh^{ζ}, while the term $\Delta J_g^{\zeta}/\Delta E_g$ identifies the group-averaged net axial current:

$$\Delta J_g^{\zeta}(\mathbf{r}_{\perp}) = \frac{1}{4\pi} \int_{4\pi} d\mathbf{\Omega}\mathbf{\Omega} \cdot \mathbf{n}(z^{\pm}) \psi_g(\mathbf{r}_{\perp}, z^{\pm}). \tag{9.2}$$

In the last equation we used $z^{\pm}$ to indicate the coordinates at the top (+) and the bottom (-) of the axial node, while $\mathbf{n}(z^{\pm})$ indicates the outgoing normal to the respective surfaces. The volume integral in Eq. (9.1) as well as the contribution of the net axial currents are computed using the values of the flux provided by the three-dimensional MOC solver. Remark that with this formulation we have neglected the energy dependence of the axial leakages within the energy group and used energy averaged values.

9.4 Results

The 3D Step-MOC solution has been computed for the geometry with two-dimensional section shown in Fig. 9.3(a), representing one twelve of the full complete assembly and composed of 115 radial regions. Axially, the geometry is divided in 30 planes with varying mesh size (see Fig. 9.3(b)). This last figure also shows the different configurations used for the self-shielding model. In configuration A, only two sets of multi-group cross sections are computed: one for the fertile zone, the other for the fissile zone. In configuration B, two additional self-shielding zones are added at the fertile/fissile interface. Concerning the representation of the scattering, a $P3$ approximation is used, which entails that 16 moments are required for each one of the 115×30 regions used for the spatial representation. The total number of unknowns of the problem is thus about 110 millions if the 1968-groups discretization of the energy is accounted for.

Concerning the trajectory discretization, the MOC calculation has been performed using a product quadrature formula composed of a 48 angles uniform set, and a 7 angles Gauss-Legendre set for, respectively, the azimuthal and the polar components. The trajectory spacing used is about $0.05cm$, which generates a total number of intersections of about 75 millions. Concerning the memory requirements of the tracking, the HSS storage has allowed a reduction of a factor 3.5 of the memory required to store the 75 millions identifiers of the crossed regions. The Chord Classification method, similarly, has allowed a reduction of the stored chord lengths by a factor 3.5.

The MOC algorithm is accelerated with a DP1 synthetic acceleration, which requires the calculation and storage of the coefficients coupling all the

50 millions unknown surface moments for all the 1968 energy groups. For this calculation the total number amounts to 3.7 billions. The solution of the DP1 system is done with a stabilized bi-conjugated gradient method [43, 10].

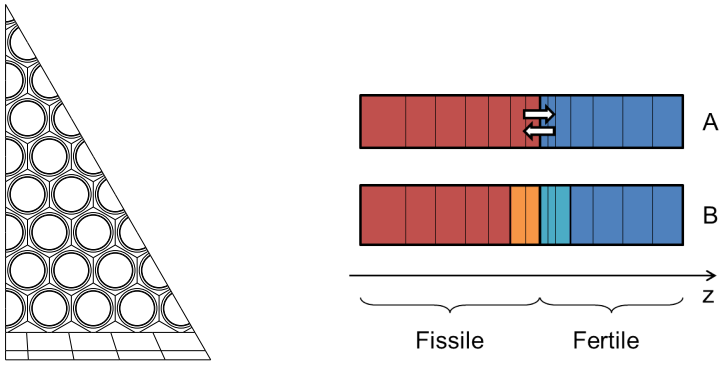
The serial execution of the complete calculation requires about 6 hours on a Xeon E5-2680@2.8 GHz. The Fig. 9.4(a) shows how the execution time is distributed among the different parts of the algorithm. The most time-consuming task is the MOC iteration which takes about 50% of the total execution time. Other time-consuming tasks are the construction of the DP1 coefficients (20%), the solution of the DP1 operator for the acceleration (11%), and the calculation of the CP coefficients (6%).

The majority of the algorithm has been parallelized using the methods described in this manuscript. Parallel directives are also used for the calculation of the CP coefficients, while about 13% of the algorithm is still serial (DP1 solution mainly). An ideal parallel execution of the algorithm would obtain a speed up following the law of Amdahl:

$$\text{Speed Up} = \frac{1}{0.13 + 0.87/Q}, \tag{9.3}$$

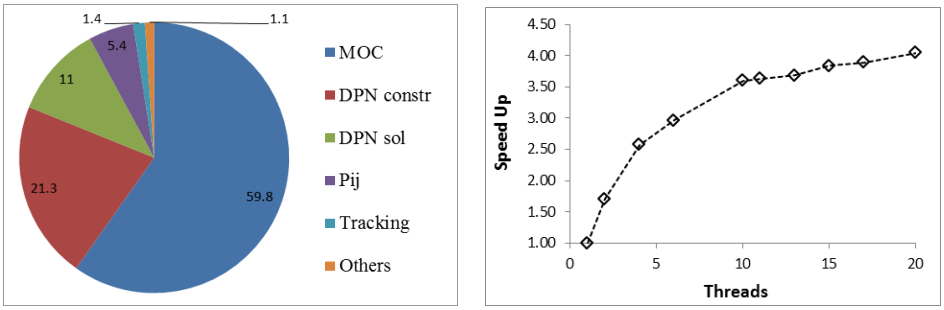
where 0.13 and 0.87 represent the fractions of, respectively, the serial and the parallelized part of the algorithm, while Q indicates the number of threads. According to this law, the maximum speed up achievable is of a factor 7.7. The Fig. 9.4(b) shows the speed up of the calculation for a varying number of threads up to 20. The figure shows that the parallel execution allows a reduction of the execution time with a speed up of about 4, but it is still far from the ideal case. The loss of efficiency is due to the parallel overheads discussed in Chapter 7. The minimum total calculation time si of $1h30$ and it is obtained for a 20-threaded execution applying the `SELFGUIDED` strategy for the sweep.

The Tab. 9.1 shows the convergence of the self-shielding models. In the `NOAUTOP` no self-shielding model is used and group-averaged cross sections are used. The absolute error on the reactivity with respect to the Monte Carlo reference solution is about $3500PCM$, while the application of the self-shielding model bring the error within the Monte-Carlo statistical error. Remark that no difference in the solution is obtained when the self-shielding model is refined, showing the convergence of the self-shielding model. Figs. 9.5 to 9.7 show the error on the total absorption reaction rate in the fuel with and without applying the self-shielding model. The error is computed by normalizing the APOLLO3® solution and the Tripoli4 solution to the total integrated value of absorption. In particular Figs. 9.6 and 9.5 show the axial variation of the error while Fig. 9.7 shows the error for the 33



(a) Radial mesh. (b) Axial mesh.

Figure 9.3: The figures show the mesh used for the calculations. In Fig. (a) we show the radial mesh of the hexagonal lattice with $2\pi/12$ symmetry. Fig. (b) represents the axial mesh: colored areas represent different self-shielding zones, while thin black lines represent the axial discretization used for the MOC calculation. Case A and B identify the two configurations used for the self-shielding calculation. The arrows are used to represent the axial leakages computed with Eq. (9.2) (omitted in configuration B for simplicity of representation).



(a) Serial profile. (b) Speed up.

Figure 9.4: The fig. (a) shows how the execution time is distributed among the different tasks for the calculation of the assembly in Fig. 9.3. The Fig. (b) shows the speed up obtained with a parallel execution for a number of threads up to 20.

	NOSS	A	B	B+Leak
$\Delta\rho$ (PCM)[a]	-3480	-1	-1	-1

Table 9.1: Error on reactivity for different self-shielding configurations (see Fig. 9.3(b)). NOSS: no self-shielding model is applied. A: configuration A with MOC sources and no leakages. B: configuration B with MOC sources and no axial leakages. B+Leak: configuration B with MOC sources and axial leakages. [a] Reference values are obtained by a Tripoli4 Monte-Carlo simulation which provides $\rho = 13872 \pm 4$ PCM in one week calculation using 6 cores on a Xeon X5650@2.67 GHz.

energy groups. Remark on these figures the strong effect of the self-shielding on both the energy and the axial variations of the error.

The axial distribution of the flux inside the fuel is shown in Fig. 9.8. The figure shows the comparison between the APOLLO3® and Tripoli4 fluxes for a thermal group and for a fast group. Results show a very good agreement between the two solutions. Remark the strong axial gradient of the flux which has required the 30-planes discretization of the axial direction.

In Figs. 9.9 to 9.11 we show the absolute error on the absorption and the fission reaction rates of the $^{235}_{92}U$, $^{238}_{92}U$, and $^{239}_{94}Pu$ isotopes in the four homogenization zones (fertile/fissile, peripheral/internal fuel pins). Comparisons with Tripoli4 are done by normalizing the distributions to the total absorption reaction rate.

In Fig. 9.12 we show the axial variation of the homogenized cross sections of the $^{238}_{92}U$ isotope for different energy groups. These cross sections represent the parameters that are used at the core-level. Remark that the homogenized cross sections for some energy groups have a non-negligible axial variation. This artificial variation derives from the homogenization process over the coarse energy mesh which is applied to impose the conservation of the reaction rates. By consequence, core-level calculations should take into account these axial variations to obtain conservative results.

9.5 Conclusions

The problem analyzed in this chapter is representative of the actual geometry of an assembly of the ASTRID reactor containing both radial and axial heterogeneities. This problem has been solved using a 3D fine-group MOC calculation coupled with a 2D Subgroups method, and by using a

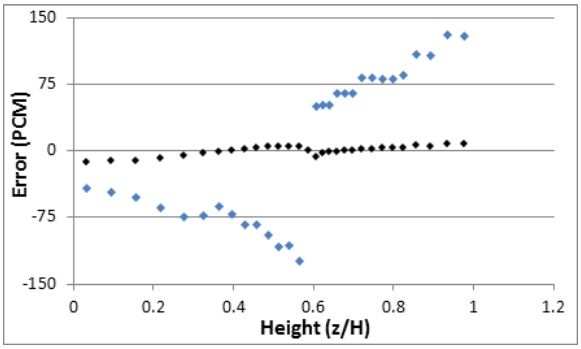


Figure 9.5: Axial variation in the fuel of the absolute error on the total absorption integrated over the energy domain. Cyan: `NOAUTOP`. Black: configuration A (Fig. 9.3(b)).

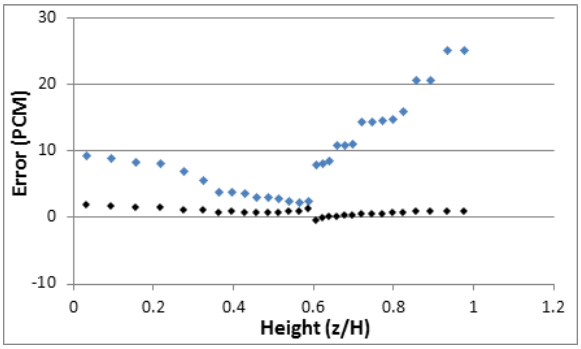


Figure 9.6: Axial variation in the fuel of the absolute error on the total absorption in the 16th energy group. Cyan: `NOAUTOP`. Black: configuration A (Fig. 9.3(b)).

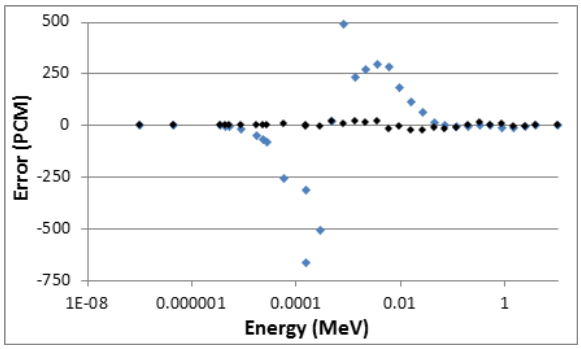


Figure 9.7: Absolute error of the total absorption in the fuel for the 33 energy groups. Values are axially integrated. Cyan: `NOAUTOP`. Black: configuration A (Fig. 9.3(b)).

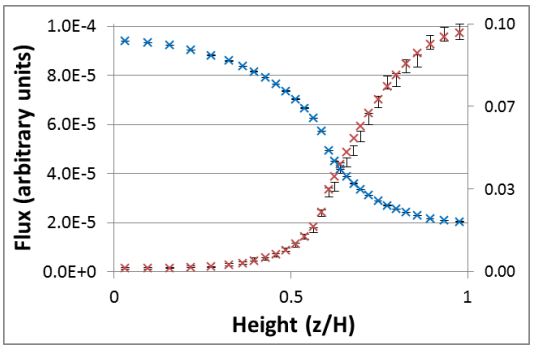
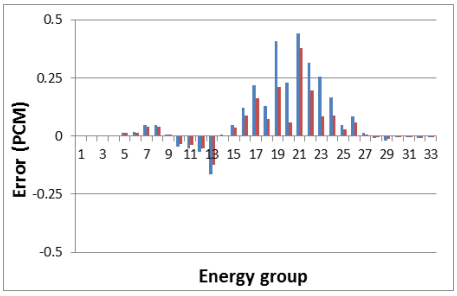
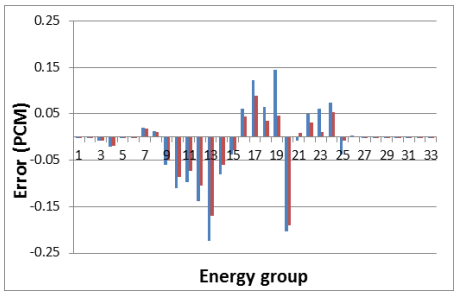


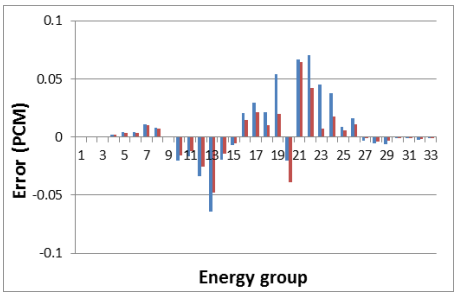
Figure 9.8: Axial distribution of the flux for a thermal (red) and a fast (cyan) group inside the fuel-pin. The figure also shows the reference Monte Carlo solution with associated error bars.



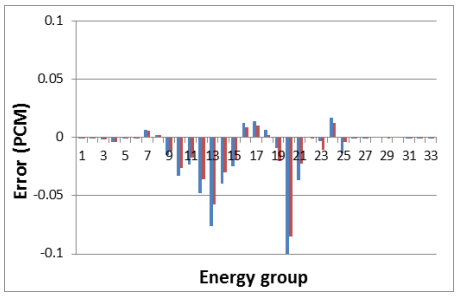
(a) Internal radial zone, fertile layer.



(b) Internal radial zone, fissile layer.

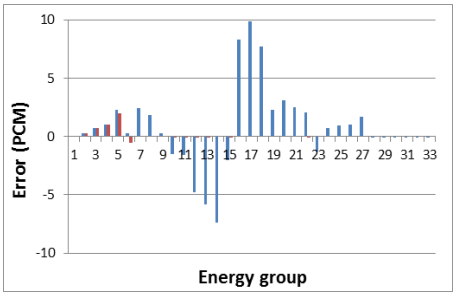


(c) Peripheral radial zone, fertile layer.

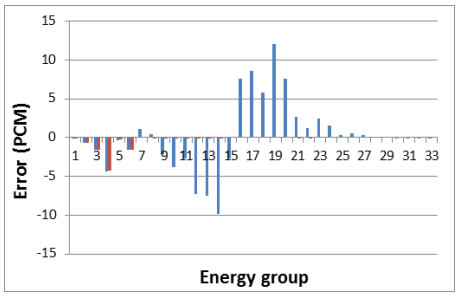


(d) Peripheral radial zone, fissile layer.

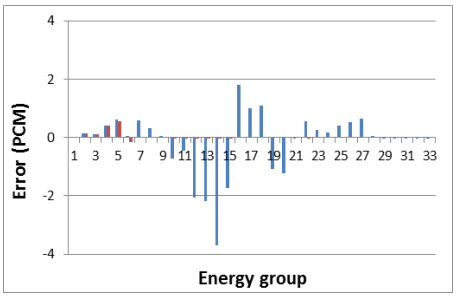
Figure 9.9: Absolute error on the energy groups of the absorption (cyan) and fission (red) reaction rates for isotope $^{235}_{92}U$.



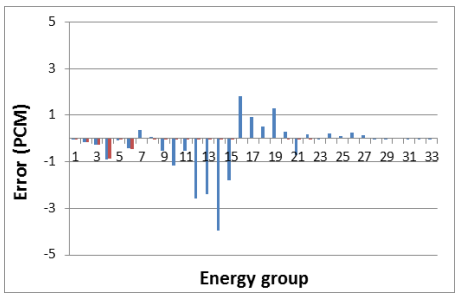
(a) Internal radial zone, fertile layer.



(b) Internal radial zone, fissile layer.

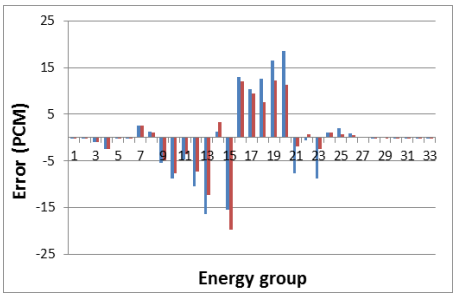


(c) Peripheral radial zone, fertile layer.

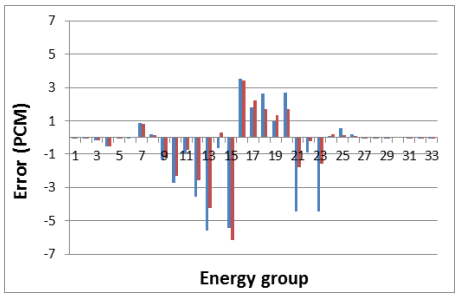


(d) Peripheral radial zone, fissile layer.

Figure 9.10: Absolute error on the energy groups of the absorption (cyan) and fission (red) reaction rates for isotope $^{238}_{92}U$.

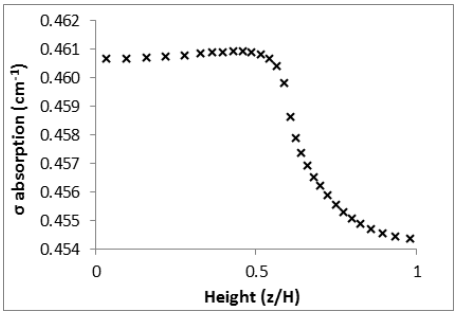


(a) Internal radial zone.

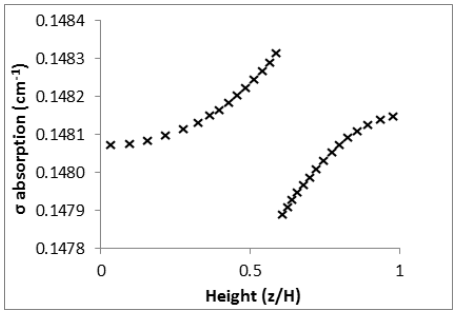


(b) Peripheral radial zone.

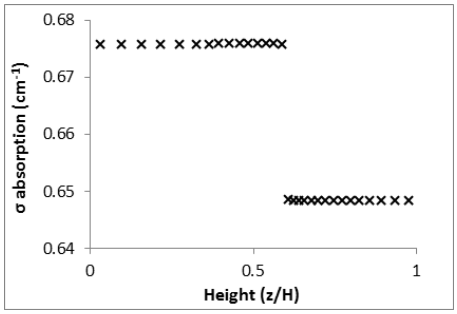
Figure 9.11: Absolute error on the energy groups of the absorption (cyan) and fission (red) reaction rates for isotope $^{239}_{94}Pu$.



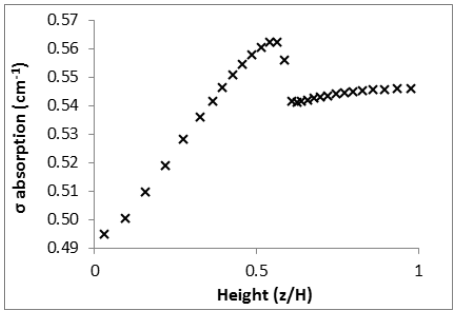
(a) Group 5.



(b) Group 10.



(c) Group 16.



(d) Group 29.

Figure 9.12: Axial variation of homogenized absorption cross sections of $^{238}_{92}U$ in the internal fuel pins.

continuous-in-energy Monte Carlo simulation as reference solution.

Comparisons between the APOLLO3® and Tripoli4 solutions have shown a very good agreement and allowed the verification of the 3D MOC solver and of the self-shielding model. From the computational point of view, the APOLLO3® scheme requires about $1h30'$ of calculation on a 20-threaded parallel execution, whereas an acceptable convergence of the Tripoli4 simulation requires about one week calculation on a 6-threaded execution, showing the computational advantages of the APOLLO3® scheme.

The relatively small execution time and the precision of the APOLLO3® scheme make it and adapted tool for the study of innovative reactor designs. Similar verification are currently done at the SPRC laboratory of the CEA of Cadarache, for example, by considering the actual geometry and composition of a full assembly.

The MOC calculation time can be further reduced by applying the polynomial expansion of the source discussed in Chapter 4. The results provided by the step approximation, in fact, show a regular axial variation of the solution which can be easily represented by polynomial functions. This option, however, has not been developed yet in APOLLO3® but it is a must for future works.

Chapter 10

Conclusions and perspectives

The objective of our research has been the implementation of a three-dimensional solver of the neutron transport equation based on the Method of Characteristics. This method solves the integral form of the transport equation along characteristics lines covering the problem geometry for the set of discrete directions belonging to an angular quadrature formula.

The majority of the methods discussed throughout this manuscript have been implemented in the nuclear reactor analysis code APOLLO3®, developed at the SERMA in CEA of Salcay. This is a new generation code designed for industrial calculations of nuclear reactors. In particular, the new methods have been implemented in the TDT solver, which provides the solution of the neutron transport equation using the multi-group discretization of the energy variable, and either a two-dimensional Method of Characteristics or a Collision Probability method for the treatment of the spatial and angular variables.

The first step towards the solution of a three-dimensional neutron transport has been the extension of the two-dimensional step-MOC approximation of TDT for the treatment of Cartesian axial geometries. These are geometries generated by the Cartesian product of a two-dimensional generic mesh and a mono-dimensional mesh. These represent, respectively, the section and the axis of the three-dimensional geometry. Although this description limits the application of the MOC to a reduced class of three-dimensional problems, it is sufficient to treat the majority of problems of reactor analysis.

The application of the MOC to these geometries has required the extension of the tracking strategies already developed for two-dimensional geometries. Special efforts have been done to extend the method of compound

trajectories to three-dimensional cases. This method allows an exact treatment of the so-called geometrical boundary conditions, that are conditions used to describe the symmetries of the geometry. This is an important step since a broad class of problems solved in reactor analysis is represented by infinite lattice calculations, for which the geometry is constituted by an infinite repetition of an elementary geometry. For these cases, we extended the tracking strategies in order to generate three-dimensional periodic trajectories.

To speed up the convergence of the MOC we also developed a three-dimensional synthetic acceleration method based on the DP_N spatial and angular representation of the neutron flux. This acceleration technique uses the difference between two MOC iterations as non-homogeneous source of an equivalent problem. The unknown of the synthetic problem is the additive correction factor that has to be applied to the MOC iteration to reduce the error with respect to its converged solution. This method has shown speed up of roughly a factor 10 with respect to the solution obtained using only free MOC iterations.

One of the main problems encountered in the application of the MOC to three-dimensional geometries is the large amount of resources required to store the intersections (chords) composing the trajectories. This is a relatively small problem in two-dimensional geometries for which the number of chords can be of the order of some hundreds of thousands, but it becomes critical in three-dimensional cases where it can easily increase by two or three orders of magnitude. The number of intersections also directly impacts the MOC execution time, since this is mostly determined by the operations required to solve the integral form of the transport equation for each chord. Hence, specialized methods have been developed to reduce both memory and time requirements of the MOC. These methods strongly exploit the regularities of Cartesian axial geometries. The Chord Classification method identifies and groups together chords that share the same length. In this way, a reduced number of chords has to be actually stored. In addition, this method allows the pre-calculation of transmission coefficients used for the trajectory sweep, avoiding to perform this calculation for each chord. The chord classification method has shown to be very effective in reducing the MOC memory requirements, also providing a good reduction of the execution time (of the order of some tens of percentage). An alternative storage method has been also implemented to further compress the tracking data. In this method trajectories are represented by the sequence of relative displacements along the Cartesian sz-planes generated by the two-dimensional trajectories used for tracking and the axial mesh. This

method allows a strong reduction of the memory needed for storing regions sequences (about factor 5/10), but it requires additional operations for the trajectory reconstruction (some tens of percentage of the time required for the trajectory sweep). Both Chord Classification and HSS storage methods have determined a strong reduction of the memory requirements of tracking data, allowing the treatment of relatively large cases on standard desktop machines.

To take full advantage of the parallel architecture of modern CPU, we applied parallel programming methods to the trajectory sweep. In this framework we tested different solutions. The parallelism over the tracking angles has the advantage of not requiring any additional operation and memory with respect to the serial algorithm, but is limited in degree of parallelism and efficiency because of the reduced number of tasks and of their non-uniform load profile. To improve the degree of parallelism, the sweep operation has been done over the trajectories, which has allowed an increase of the number of tasks that can be executed independently. As a drawback, this method requires additional memory to avoid concurrent I/O operations (race conditions). Also it requires additional operations for the so-called reduction operation, which gathers the partial contributions computed by each thread. An important part of the research has also been dedicated to balance the sweep operation to avoid thread idleness. For this purpose different scheduling strategies have been tested, and custom scheduling strategies have been implemented in order to take into account the actual load profile. These scheduling strategies allow a self-scheduling of the algorithm without any external tuning. In particular, a static strategy has been developed which provides good scaling with the minimum number of tasks. Static strategies with appropriate ordering of tasks also reduce the number of angles swept by each thread, which is advantageous for limiting the costs of the reduction operation and to reduce memory duplication. Parallel strategies have also been applied to speed up the calculation of the $\mathrm{DP_N}$ coefficients. In this case, data parallelism over the energy groups is an optimal strategy since it does not require any memory duplication, and it assures an uniform distribution of the load. The ensemble of the parallel strategies has allowed a speed up of a factor 4 for a 20-threaded calculation of a reactor assembly. The scaling is limited by the parallel overheads such as memory duplication and synchronizations.

Another subject tackled in this Ph.D. thesis has been the research of adapted quadrature formulas for three-dimensional axial geometries. The research has been focused on high order polar quadrature formulas with particular attention to quadrature sets containing the horizontal and the

vertical directions. This last choice derives from the fact that the computational cost of the sweep for these special directions is negligible with respect to that of standard directions. Results have shown that, even for heterogeneous cases, high order quadrature formulas have a better convergence rate than lower order ones. Differently, Bickley quadratures show poor convergence for heterogeneous cases, while they provide good results for quasi-2D cases. Results have also shown that the use of the horizontal and vertical directions adds a bias in the solution while keeping similar convergence rates. The origin of this bias has not been identified and requires further investigation.

The application of the MOC to physical cases requires the definition of multi-group cross sections that correctly take into account the self-shielding effects deriving from resonances. Part of the research has been dedicated to the coupling between the MOC and the Subgroup method for a on-line calculation of both multi-group cross sections and multi-group fluxes. The coupling has been done by applying a spatial equivalence between the three-dimensional MOC and the two-dimensional Collision Probability method used in the Subgroups method.

The verification of the step-MOC algorithm and of the self-shielding model has been done through comparisons with a reference calculation computed with the Tripoli4 continuous-in-energy Monte Carlo solver. The problem considered for this verification is a three-dimensional assembly in infinite lattice configuration representing a detail of the Gen IV sodium-cooled fast breeder reactor ASTRID. Results have shown that the approximated description of the space introduced by the 2D/3D self-shielding equivalence is sufficient to catch the main resonance effects. The effective multiplicative factor obtained with the step-MOC solution is within the Monte-Carlo statistical error (4 PCM). The main source of error appears to derive from the multi-group discretization, whereas spatial effects are correctly accounted for.

Thanks to the developments done during our research, the TDT solver of APOLLO3® is now capable of solving the neutron transport equation in three-dimensional axial geometries with reasonable computational requirements (few GB of memory and some hours of calculation), and small errors with respect to Monte Carlo reference solutions. These latter generally require far larger execution times (days of calculations) which make the 3D MOC still advantageous in an industrial contest.

The methods developed throughout our research have been subject of three publications in international conferences [63, 49, 55], and of one publication in an international journal [45].

Because of limited timing of our research, some of the methods discussed in this manuscript have not been implemented and tested in the APOLLO3® version of TDT. A development which seems fundamental to speed up of the MOC solver is the polynomial expansion of the source along the axial direction. In fact, most of the problems encountered in industrial calculations are characterized by mild heterogeneities along the axial direction. By consequence, the axial variation of the solution is a regular function which can be faithfully described by a polynomial representation, allowing a reduction of meshes required for the spatial convergence of the numerical solution. A similar representation is also possible for the DP_N approximation used for the acceleration of the MOC solver but its derivation has not been explicitly carried out.

Concerning the parallelism, alternative strategies have been also proposed to further improve the scaling of the MOC for a large number of threads. These go from a domain-decomposition method with exact boundary conditions for the calculation of optically large problems, to the application of hybrid Gauss-Seidel/Gauss-Jacobi algorithms for the parallelization of the multi-group problem. This latter can be determining for the speed up of cases with a fine-group discretization of the energy.

Bibliography

[1] J. Askew, A characteristics formulation of the neutron transport equation in complicated geometries, Tech. rep., UK Atomic Energy Establishment (1972).

[2] M. Halsall, CACTUS, a characteristics solution to the neutron transport equations in complicated geometries, Tech. rep., UKAEA Atomic Energy Establishment, Winfrith (United Kingdom) (1980).

[3] R. Ferrer, J. Rhodes, K. Smith, Linear source approximation in CASMO5, in: PHYSOR , Knoxville (TN), 2012.

[4] W. Boyd, S. Shaner, L. Li, B. Forget, K. Smith, The OpenMOC method of characteristics neutral particle transport code, Annals of Nuclear Energy 68 (2014) 43–52.

[5] R. Roy, The cyclic characteristic method, in: International conference Physics of Nuclear Science and Technology, 1998.

[6] R. Sanchez, I. Zmijarevic, M. Coste-Delclaux, E. Masiello, S. Santandrea, E. Martinolli, L. Villate, N. Schwartz, N. Guler, APOLLO2 year 2010, Nucl. Eng. and Technology 42 (2010) 474–499.

[7] P. Mosca, Conception et développement d'un mailleur énergétique adaptatif pour la génération des bibliotéques multigroupes des codes de transport, Ph.D. thesis, Paris-Sud 11 (2009).

[8] A. Abramowitz, I. Stegun, Handbook of Mathematical Functions, Dover Publications, 1965.

[9] E. E. Lewis, W. F. Miller, Computational Methods of Neutron Transport Theory, 1985.

[10] Y. Saad, Numerical methods for large eigenvalue problems, 2011.

[11] G. B. Arfken, H. J. Weber, F. E. Harris, Mathematical Methods for Physicists, Accademic Press, 2012.

[12] K. Lathrop, Ray effects in discrete ordinates equations, Nuclear Science and Engineering 32 (1968) 357.

[13] R. Sanchez, J. Ragusa, On the construction of Galerking angular quadratures, Nuclear Science and Engineering 169 (2011) 133–154.

[14] C. D. Ahrens, Derivation of new 3D dicrete ordinate equations, in: PHYSOR , Knoxville (TN), 2012.

[15] R. Sanchez, Prospects in deterministic three-dimensional whole-core transport calculations, Nuclear Engineering and Technology 44 (2012) 113–150.

[16] G. Rimpault, M. Grimstone, Validation of new sub-group algorithms for resonance self shielding in heterogeneous structures, in: Topical Meeting on Advances in Nuclear En-gineering Computation and Radiation Shielding, Santa Fe, New Mexico, April 9-13,, 1989.

[17] P. Ribon, J.-C. Sublet, M. Coste-Delclaux, CALENDF-2002: User manual, Technical Report CEA-R-6020, CEA (2003).

[18] P. Ribon, J.-M. Maillard, Les tables de probabilite: application au traitement des sections efficaces pour la neutronique, Technical Report CEA-N-2485, CEA (1986).

[19] M. Herman, ENDF-6 data formats and procedures for the evaluated nuclear data file ENDF-VII, Cross Section Evaluation Working Group (2005).

[20] L. Lei-Mao, Subgroup method in the APOLLO3 selfshielding calculations, Technical Report RT/11-5242, CEA (2011).

[21] M. L. Williams, Correction of multigroup cross sections for resolved resonance interference in mixed absorbers, Nuclear Science and Engineering 83 (1983) 9–37.

[22] J. Y. Cho, H. G. Joo, Solution of the C5G7MOX benchmark three-dimensional extension problems by the decart direct whole core calculation code, Progress in Nuclear Energy 48 (5) (2006) 456 – 466.

[23] J. Cho, K. Kim, C. Lee, S. Zee, H. Joo, Axial SPN and radial MOC coupled whole core transport calculation, Journal of Nuclear Science and Technology 44 (2007) 1156–1171.

[24] F. Févotte, B. Lathuilière, Micado: Parallel implementation of a 2D-1D iterative algorithm for the 3D neutron transport problem in prismatic geometries, in: M&C, Sun Valley (ID), 2013.

[25] I. Zmijarevic, Résolution de l'équation de transport par des méthodes nodales et des caractéristiques dans les domaines à trois dimensions, Ph.D. thesis, Université de Provence Aix-Marseille I (1998).

[26] R. Lenain, E. Masiello, F. Damian, R. Sanchez, Domain-decomposition method for 2D and 3D transport calculations using hybrid MPI/OpenMP parallelism, in: M&C, SNA + Monte Carlo, Nashville (TN), 2015.

[27] I. Suslov, MCCG3D - 3D discrete-ordinates transport code for unstructured grid. state of the art and future development, in: Proceedings of Seminar "Neutronics-96", Obninsk, Russia, 1998, p. 162.

[28] G. Wu, R. Roy, A new characteristics algorithm for 3D transport calculations, Annals of Nuclear Energy 30 (1) (2003) 1 – 16.

[29] B. Kochunas, B. Collins, D. Jabaay, T. J. Downar, W. R. Martin, Overview of development and design of MPACT: Michigan parallel characteristics transport code, American Nuclear Society - ANS; La Grange Park (United States), 2013.

[30] Z. Liu, H. Wu, L. Cao, Q. Chen, Y. Li, A new three-dimensional method of characteristics for the neutron transport calculation, Annals of Nuclear Energy 38 (2011) 447–454.

[31] J. B. Taylor, D. Knott, A. J. Baratta, A method of characteristics solution to the oecd/nea 3D neutron transport benchmark problem, in: M&C, SNA, Monterey (CA), 2007.

[32] J. F. Fevotte, Techniques de traçage pour la méthode des caractéristiques appliquée à la résolution de l' équation du transport des neutrons en domaines multi-dimensionnels, Ph.D. thesis, Paris-Sud 11 (2009).

[33] S. Santandrea, J. Jaboulay, P. Bellier, F. Fevotte, H. Golfier, Improvement and validation of the linear surface characteristics scheme, Annals of Nuclear Energy 36 (2009) 46–59.

[34] R. Ferrer, J. Rhodes, Extension of linear source MOC method to anisotropic scattering in CASMO5, in: PHYSOR , Kyoto (JP), 2014.

[35] E. Masiello, R. Clemente, S. Santandrea, Higher-order method of characteristic for 2-D unstructured meshes, in: M&C, Saratoga Springs, New York, 2009.

[36] R. M. Ferrer, Y. Y. Azmy, A robust arbitrarily high order transport method of the characteristic type for unstructured tetrahedral grids, in: M&C, Saratoga Springs, New York, 2009.

[37] X. M. Chai, K. Wang, The linear source approximation in three dimension characteristics method, in: M&C, Saratoga Springs, New York, 2009.

[38] G. Wu, R. Roy, Acceleration techniques for trajectory-based deterministic 3d transport solvers, Annals of Nuclear Energy 30 (5) (2003) 567 – 583.

[39] H. Khalil, Effectiveness of a consistently formulated diffusion synthetic acceleration differencing approach, Nuclear Science and Engineering 98 (1988) 226–243.

[40] I. R. Suslov, An algebraic collapsing acceleration in long characteristic transport theory, in: Proc. 11th Symp. Atomic Energy Research, Csopak, Hungary, 2001, p. 162.

[41] R. Le Tellier, A. Hébert, An improved algebraic collapsing acceleration with general boundary conditions for the characteristics method, Nuclear Science and Engineering 156 (2007) 121–138.

[42] L. Li, K. Smith, B. Forget, A low order non linear transport acceleration scheme for the method of the characteristic, in: PHYSOR , Kyoto (JP), 2014.

[43] S. Santandrea, R. Sanchez, Analysis and improvement of the DPN acceleration technique for the method of characteristics in unstructured meshes, Annals of Nuclear Energy 32 (2005) 163–193.

[44] S. Santandrea, R. Sanchez, Acceleration techniques for the characteristic method in unstructured meshes, Ann. Nucl. Energy 29 (2002) 323–352.

[45] D. Sciannandrone, S. Santandrea, R. Sanchez, Optimized tracking strategies for step MOC calculations in extruded 3D axial geometries, Annals of Nuclear Energy.

[46] R. Sanchez, L. Mao, S. Santandrea, Treatment of boundary conditions in trajectory-based deterministic transport methods, Nucl. Sci. Eng. 140 (2002) 23–50.

[47] R. Le Tellier, G. Marleau, M. Dahmani, A. Hébert, Improvements of the reactivity devices modeling for the advanced {CANDU} reactor, Annals of Nuclear Energy 35 (5) (2008) 868 – 876.

[48] I. Suslov, Improvements in the long characteristics method and their efficiency for deep penetration calculations, Progress in Nuclear Energy 39 (2001) 223 – 242.

[49] D. Sciannandrone, S. Santandrea, Tracking strategies in 3D axial geometries for a MOC solver, in: SNA + MC, Paris, 2013.

[50] U. Drepper, What every programmer should know about memory (November 2007).
URL http://www.akkadia.org/drepper/cpumemory.pdf

[51] OpenMP Architecture Review Board, OpenMP application program interface version 3.0 (2008).

[52] L. Naymeh, Analyse et développement d'un schéma de discrétisation numérique de l'équation du transport des neutrons en géométrie tridimensionelle, Ph.D. thesis, Paris-Sud 11 (2014).

[53] W. Boyd, K. Smith, B. Forget, Parallel performance results for the OpenMOC method of characteristic code on multi-core platforms, in: PHYSOR , Kyoto (JP), 2014.

[54] M. Dahmani, G. J. Wu, R. Roy, J. Koclas, Development and parallelization of the three-dimensional characteristics solver MCI of DRAGON, in: Physor, Seoul (KR), 2002.

[55] D. Sciannandrone, S. Santandrea, R. Sanchez, L. Lei-Mao, J.-F. Vidal, P. Archier, J.-M. Palau, Coupled fine-group three-dimensional flux

calculation and subgroups method for a fbr hexagonal assembly with the APOLLO3® core physics analysis code, in: M&C, SNA + Monte Carlo, Nashville (TN), 2015.

[56] A. Leonard, C. McDaniel, Optimal polar angles and weights for the characteristics method, Trans. Am. Nucl. Soc. 73 (1995) 172.

[57] A. Yamamoto, M. Tabuchi, N. Sugimura, Derivation of optimum polar angle quadrature set for the method of characteristics based on approximation error for the bickley function, Journal of Nuclear Science and Technology 44 (2007) 129–136.

[58] P. Davis, P. Rabinowitz, Methods of numerical integration, Academic Press, 1975.

[59] C. Clenshaw, A. Curtis, A method for numerical integration on an automatic computer, Numerische Mathematik 2 (1960) 197–205.

[60] L. N. Trefethen, Is gauss quadrature better than Clenshaw-Curtis?, SIAM Rev. 50 (2008) 67–87.

[61] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes 3rd Edition: The Art of Scientific Computing, Cambridge University Press, 2007.

[62] J. Both, et al., A survey of TRIPOLI-4, in: Proceedings of the 8th International Conference on Radiation Shielding, Arlington, Texas, 1994.

[63] D. Sciannandrone, S. Santandrea, R. Sanchez, Angular quadrature formulas for step-MOC calculations in three-dimensional axial geometries, in: PHYTRA3, Tètouan, Morocco, 2014.

[64] L. C. Pierre, J.-F. Sauvage, J.-P. Serpantie, Sodium-cooled fast reactors: the astrid plant project, in: ICAPP, Nice (FR), 2011.

[65] P. Sciora, D. Blanchet, L. Buiron, B. Fontaine, M. Vanier, F. Varaine, C. Venard, Low voided effect core design applied on 2400 MWth SFR reactor, in: ICAPP, Nice (FR), 2011.

[66] J.-Y. Moller, J.-J. Lautard, MINARET, a deterministic neutron transport solver for nuclear core calculations, in: M&C, Rio de Janeiro (BR), 2011.

[67] G. Rimpault, D. Plisson, J. Tommasi, R. Jacqmin, The ERANOS code and data system for fast reactor neutronic analyses, in: PHYSOR, Seoul (SKR), 2002.

[68] G. Rimpault, Y. Peneliau, J. Vidal, S. Mirotta, A. Gandini, In depth uncertainty estimation of the neutron compuational tools, in: PHYSOR, Kyoto (JP), 2014.