An axial polynomial expansion and acceleration of the characteristics method for the solution of the Neutron Transport Equation

Laurent Graziano

université PARIS-SACLAY

NNT: 2018SACLS389

Thèse de doctorat

An axial polynomial expansion and acceleration of the characteristics method for the solution of the Neutron Transport Equation

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Université Paris-Sud

École doctorale n° 576 PHENIICS
Spécialité de doctorat: Énergie nucléaire

Thèse présentée et soutenue à Saclay, le 16 octobre 2018, par

Laurent Graziano

Composition du jury:

Dr. Richard SANCHEZ Former CEA research director	Rapporteur
Prof. Sandra DULLA Politecnico di Torino	Rapporteur
Prof. Jean C. RAGUSA Texas A&M	Examinateur
Prof. Danny LATHOUWERS TU Delft	Examinateur
Prof. Pierre DESESQUELLES Université Paris-Sud	Président du jury
Dr. Alain MAZZOLO CEA/DEN	Directeur de thèse
Dr. Simone SANTANDREA CEA/DEN	Encadrant
Dr. Daniele SCIANNANDRONE CEA/DEN	Invité

Contents

Abstract

The purpose of this PhD is the implementation of an axial polynomial approximation in a three-dimensional Method Of Characteristics (MOC) based solver. The context of the work is the solution of the steady state Neutron Transport Equation (NTE) for critical systems, and the practical implementation has been realized in the Two/Three Dimensional Transport (TDT) solver, as a part of the APOLLO3® project. A three-dimensional MOC solver for *3D extruded* geometries has been implemented in this code during a previous PhD project, relying on a piecewise constant approximation for the neutrons fluxes and sources. The developments presented in the following represent the natural continuation of this work. Three-dimensional neutron transport MOC solvers are able to produce accurate results for complex geometries. Although accurate, the computational cost associated to this kind of solvers is very important. An axial polynomial representation of the neutron angular fluxes has been used to lighten this computational burden.

The work realized during this thesis can be considered divided in three major parts: *transport*, *acceleration* and *others*. The first part is constituted by the implementation of the chosen polynomial approximation in the transmission and balance equations typical of the method of characteristics. This part was also characterized by the computation of a set of numerical coefficients necessary to obtain a stable algorithm. During the second part, we modified and implemented the solution of the equations of the DP_N synthetic acceleration. This method was already used for the acceleration of both inner and outer iteration in TDT for the two and three dimensional solvers at the beginning of this work. The introduction of a polynomial approximation required several equation manipulations and associated numerical developments. In the last part of this work we have looked for the solutions of a series of different issues associated to the first two parts. Firstly, we had to deal with numerical instabilities associated to a poor numerical spatial or angular discretization, for both the transport and the acceleration methods. Secondly, we have conceived different methods to reduce the memory footprint of the acceleration coefficients. The approach that we have eventually chosen relies on a non-linear least squares fitting of the cross sections dependence of such coefficients. The traditional approach consists in storing one set of coefficients per each energy group. The fit method allows replacing this information with a set of coefficients computed during the regression procedure that are used to reconstruct the acceleration matrices on-the-fly. This procedure of course adds some computational cost to the method, but we believe that the reduction in terms of memory makes it worth it.

In conclusion, our work has focused on applying a simple polynomial approximation in order to reduce the computational cost and memory footprint associated to a MOC solver used to compute the neutron fluxes in three dimensional extruded geometries. Even if this does not constitute a radical improvement, the high order approximation that we have introduced allows a reduction in terms of memory and computational times of a factor between 2 and 5, depending on the case. We think that these results will constitute a fertile ground for further improvements.

1. Introduction

Nuclear reactor physics, sometimes referred to as *Neutronics*, deals with one of the most important and fascinating subjects of nuclear engineering: the behaviour of the neutron population inside the reactor core. Being able to understand and simulate how neutrons move and interact is a necessary condition to design and operate nuclear reactors.

Nuclear energy is facing important challenges and struggling to keep up a good image in the public opinion, in particular after the Fukushima Daiichi accident. Safety criteria were revised and became stricter. Meanwhile, in the nuclear community, interests for some new reactor types are starting to rise again. Reactors belonging to the so-called *generation IV* (Gen-IV) should be able to deliver better performances with better safety and security. The idea to develop fast breeder reactors, for example, which should be able to produce energy for a longer period of time while reducing the amount of radioactive waste produced, is very appealing. The idea of fast breeder reactor was already proposed in the past and test facilities in different parts of the world were constructed. Unfortunately, this type of technology comes with bigger challenges both in simulation, operation and materials. Nowadays, these kinds of reactors are not ready yet to be deployed.

However, thanks to the advancing in all technological fields, fast breeder reactor reactors could at the present moment have a second chance to became reality. For this to happen, of course they should prove to satisfy all the *post Fukushima* safety criteria and to be economically viable. These non-trivial requirements need considerable investments and efforts. One of the aspects that requires an improvement is the simulation tools used in the nuclear domain. In fact thanks to the rapid increase of computing power experienced in the recent period, along with the availability of powerful parallel computers, it is possible to aim at the development of better simulation tools. Neutron simulation has always been a difficult task, due to the presence of a variety of unknowns that requires a very fine representation. Depending on the problem, approximations of different precision level have been introduced in order to make a simulation affordable. However, by increasingly exploiting the fast evolution of the computational methods, it is becoming possible to relieve some approximations and aim at increasing the accuracy of the neutronic simulations.

The object of this work is the improvement of one of the simulation tool under development at the CEA of Saclay. The Two and Three Dimensional Transport (TDT) code, in the APOLLO3® project, delivers a deterministic solution of the neutron transport equation using the Method Of Characteristics (MOC). The code is able to treat 3D realistic geometries solving a direct transport problem, without homogenization. An axial polynomial transport method and acceleration methods have been developed during this PhD work, in order to offer a more effective treatment of typical reactor geometries. The angular flux is represented with a set of polynomial functions. With a polynomial representation, the number of axial meshes needed to represent the flux gradients is strongly reduced, when comparing to the use of a constant approximation. This translates in significant advantages both in terms of memory and computational time.

Part I

Background

2. General nuclear properties

Every atomic nucleus is constituted by protons and neutrons. The number of protons, referred to as *atomic number* (Z), defines a chemical element. The number of neutrons defines an *isotope*, which indicates a family of nuclei sharing the same chemical properties, but different nuclear behaviour. The sum of the number of protons and neutrons is referred to as *mass number* (A).

The measured masses of the known element nuclei do not exactly coincide with the sum of the masses of the protons and neutrons constituting them. This effect is known as *mass defect*. This "missing mass" corresponds, in accordance to Einstein's equation $E = m\,c^2$, to the binding energy between the nucleus particles, released at the moment of the nucleus formation. The higher the difference between the mass of the nucleus and the sum of the masses of the fission products, the higher the energy released under a fission event.

The measured energy corresponding to the mass defect is shown in Fig.1a. Without approaching the difficult task of going into the details of this phenomenon, we can assess from this figure the basic principle of nuclear fission: breaking heavy nuclei into lighter products will release energy.

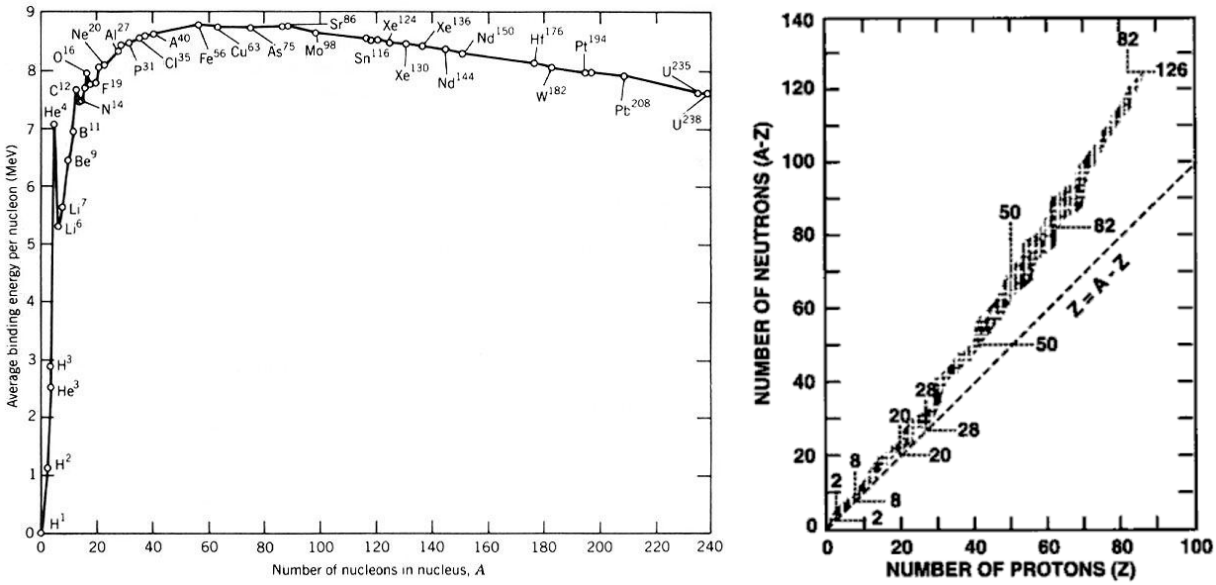
In nuclear reactors fissions are caused by neutron interacting with the fissionable isotopes. The energy released during a fission generally comes with the production of secondary particles, among which, a number of neutrons larger than one. As Fig. 1b shows, the number of neutrons needed to compensate for the electrostatic force between protons increases non-linearly with the atomic number. Therefore, when a heavy nucleus is broken into two lighter ones, a certain amount of neutrons is released. Since this number is larger than one, a self-sustained nuclear reaction is possible.

The energy and the neutrons released during the fission process represent the cornerstones of the nuclear energy production system.

2.1 Cross sections

Fission is one of the possible reactions following a particle collision. To each reaction corresponds a microscopic cross section σ_j, which indicates the likelihood of the reaction j to happen. The microscopic cross section measure units are *barns* ($10^{-24}\,cm^2$). Cross sections can be interpreted as the area seen by the incident neutron and they strongly depend on the neutron energy. Microscopic cross sections are tabulated values obtained with laboratory experiments and completed by mathematical models, in order to have a continuous representation across the energy domain. As an example, we report in Fig. 2 the microscopic cross section for the ^{235}U isotope.

The macroscopic cross sections read $\Sigma_j = \sigma_j\,N_i$, where N_i indicates the concentration of the isotope i expressed in $atoms/cm^3$. The units of measure of the macroscopic cross sections are cm^{-1}. The inverse of the macroscopic cross section is generally referred to as *mean free*



(a) (b)

Figure 1 – Binding energy as a function of the mass number (a) and neutrons to protons ratio (b).

path, which represents the average distance travelled by a neutron between two collisions.

2.2 The Neutron Transport Equation

The Neutron Transport Equation (NTE), or linear neutron Boltzmann equation, is the most used mathematical tool to represent the neutron behaviour inside a nuclear reactor. The present work focuses on a steady state treatment of the Boltzmann equation. The steady state approach is of course not representative of transient problems such as start, shut down, accidents scenarios and any kind of power fluctuation. For nominal conditions, however, a steady state approach is sufficiently representative and results less expensive in terms of computational time.

Since the number of neutrons in reactor systems is very large, they can be studied not as individual particles, but as a continuous-like function, whose values will not be representative of a point of the phase space, but rather of the averages of the quantities in a small interval around this point. The unknown of the neutron transport equation is the neutron angular flux, $\psi(\vec{r},\vec{\Omega},E)\ \left[\frac{neutrons}{cm^2\,s}\right]$, which indicates the number of neutrons crossing a surface orthogonal to the neutrons motion direction, per unit time. The NTE can be obtained applying a particle balance in a small element of volume $d\vec{r}$, in a point of the domain defined by the position $\vec{r}$ inside a domain D, for neutrons travelling with energy E and direction $\vec{\Omega}$, and it reads:

$$\begin{aligned}\Big(\vec{\Omega}\cdot\vec{\nabla}+\Sigma(\vec{r},E)\Big)\,\psi(\vec{r},\vec{\Omega},E) = \qquad\qquad (2.1)\\ +\int_0^\infty dE'\oint\frac{d\vec{\Omega}'}{4\pi}\,\Sigma_s(\vec{r},E'\to E,\vec{\Omega}\cdot\vec{\Omega}')\,\psi(\vec{r},\vec{\Omega}',E') \qquad for\ \ \vec{r}\in D,\ \vec{\Omega}\in S_{4\pi}\\ +\sum_{i=1}^{N_f(\vec{r})}\chi_i(E)\int_0^\infty dE'\,\nu_i(E')\,\Sigma_{f,i}(\vec{r},E')\oint\frac{d\vec{\Omega}'}{4\pi}\,\psi(\vec{r},\vec{\Omega}',E')+S(\vec{r},\vec{\Omega},E),\end{aligned}$$

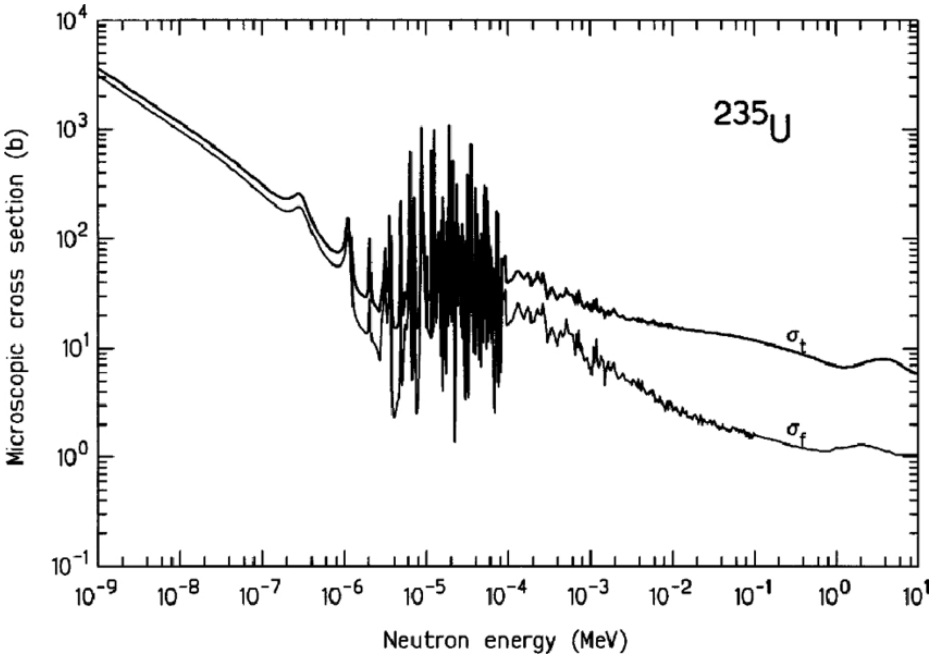


Figure 2 – Total and fission microscopic cross section for ^{235}U as a function of the energy [1].

where:

$\psi(\vec{r}, \vec{\Omega}, E) = v\, n(\vec{r}, \vec{\Omega}, E)$ is the neutron angular flux resulting as the product between the neutron velocity (v) and concentration (n).

$\vec{\Omega} \cdot \vec{\nabla}\, \psi(\vec{r}, \vec{\Omega}, E)$ the integral of this quantity represents the angular flux variation due to neutrons entering and exiting from the element of volume $d\vec{r}$.

$\Sigma(\vec{r}, E)\, \psi(\vec{r}, \vec{\Omega}, E)$ is the removal term due to neutron capture, fission or transfer.

$\oint \frac{d\vec{\Omega}}{4\pi}$ stands for the integration over the unit sphere.

$\int_0^\infty dE' \oint \frac{d\vec{\Omega}'}{4\pi}\, \Sigma_s(\vec{r}, E' \to E, \vec{\Omega} \cdot \vec{\Omega}')\, \psi(\vec{r}, \vec{\Omega}', E')$ is the transfer term.

$\sum_{i=1}^{N_f(\vec{r})} \chi_i(E) \int_0^\infty dE'\, \nu_i(E')\, \Sigma_{f,i}(\vec{r}, E') \oint \frac{d\vec{\Omega}'}{4\pi}\, \psi(\vec{r}, \vec{\Omega}', E')$ is the fission term.

$S(\vec{r}, \vec{\Omega}, E)$ is the external source term, which takes into account neutrons produced by processes independent of the neutron flux, such as spontaneous decay.

Going through each term we explicitly report the meaning of each symbol for clarity:

$\Sigma(\vec{r}, E)$ is the total cross section.

$\Sigma_s(\vec{r}, E' \to E, \vec{\Omega} \cdot \vec{\Omega}')$ is the transfer cross section.

$N_f(\vec{r})$ is the number of fissile isotopes.

$\chi_i(E)$ is the fission emission spectra, for isotope i.

$\nu_i(E')$ is the average number of emitted neutrons for isotope i.

$\Sigma_{f,i}(\vec{r}, E')$ is the fission cross section for isotope i.

Equation (2.1) is valid for a *sub-critical* system, that is when a self-sustained fission chain cannot exists. When the number of neutrons emitted during fissions is sufficient for the fission chain to be self-sustained, we will say that we are in a *critical* condition. When approaching criticality, which is the common operating state of a nuclear reactor, the external source term is generally negligible when compared to the number of neutrons produced by fissions and, in order to be able to compute a physical solution, equation (2.1) is studied as an eigenvalue problem, reading:

$$\begin{aligned} &\left(\vec{\Omega} \cdot \vec{\nabla} + \Sigma(\vec{r}, E)\right) \psi(\vec{r}, \vec{\Omega}, E) = \\ &+ \int_0^\infty dE' \oint \frac{d\vec{\Omega}'}{4\pi} \, \Sigma_s(\vec{r}, E' \to E, \vec{\Omega} \cdot \vec{\Omega}') \, \psi(\vec{r}, \vec{\Omega}', E') \qquad for \ \vec{r} \in D, \ \vec{\Omega} \in S_{4\pi} \\ &+ \frac{1}{k_{eff}} \sum_{i=1}^{N_f(\vec{r})} \chi_i(E) \int_0^\infty dE' \, \nu_i(E') \, \Sigma_{f,i}(\vec{r}, E') \oint \frac{d\vec{\Omega}'}{4\pi} \, \psi(\vec{r}, \vec{\Omega}', E'), \end{aligned} \tag{2.2}$$

where k_{eff} is the effective multiplication factor.

Boundary Conditions

To complete the mathematical representation of our problem we also need to impose a set of boundary conditions. The boundary conditions impose the values of the entering angular flux at the domain boundary ∂D and for the set of the entering directions defined by $S_{2\pi}^-$:

$$\psi(\vec{r}, \vec{\Omega}, E) = \psi_{in} \qquad \vec{r} \in \partial D, \ \vec{\Omega} \in S_{2\pi}^-. \tag{2.3}$$

Some of the usual boundary conditions imposed may be *vacuum* ($\psi_{in} = 0$) or *reflection*, *rotation* and *translation*, depending on the system symmetries. When the computational domain presents some sort of symmetries, it is sufficient to compute the solution only on the smallest domain from which the whole system can be obtained by applying the appropriate geometrical movement. For these kinds of so-called *geometrical* boundary conditions we will say that:

$$\psi(\vec{r}, \vec{\Omega}, E) = \psi(G(\vec{r}, \vec{\Omega}), E) \qquad \vec{r} \in \partial D, \ \vec{\Omega} \in S_{2\pi}^-,$$

which is to say that the entering angular flux from a point $\vec{r}$ with direction $\vec{\Omega}$ will be imposed equal to the exiting angular flux in the point $G(\vec{r}, \vec{\Omega})$, where G represents the geometrical movement associated with the kind of symmetry.

2.2.1 Classical deterministic approximations

We briefly describe in the following some of the most important approximations applied to our problem, described by Eq.(2.2). Approximations are needed in deterministic methods to deal with the spatial, angular and energy dependence.

Multi-group approximation

The energy domain in nuclear reactor physics ranges from about 10^7 eV, the maximal energy of neutrons emitted during fission, to about 10^{-5} eV. Such a vast domain is divided in deterministic codes in a set of intervals, applying the so-called *multi-group* approximation. As for every other kind of discretization, the width of the group interval is a compromise choice between the desired precision and the relative cost. Laboratory measured cross sections are obtained as point-wise values, close enough to each other to be used to feed theoretical models which deliver values representative of the continuous behaviour. Monte Carlo based solvers use this fine representation and are therefore able to simulate in the best possible way the particles energy distribution. For deterministic codes, on the other hand, per-group constant cross sections are used.

The energy dependence E will be replaced with a superscript g, indicating that we are dealing with the group number g, over a total number of groups N_g. Denoting the upper and lower energy bounds for each group respectively as E_{g-1} and E_g, we define the average per-group angular flux value as:

$$\psi^g(\vec{r},\vec{\Omega}) = \frac{1}{\Delta E_g}\int_{E_g}^{E_{g-1}} dE\ \psi(\vec{r},\vec{\Omega},E),$$

where $\Delta E_g = E_{g-1} - E_g$, is the group width. Remark that, as customary, the group counting starts from the highest energy value. By applying this integration to each terms of the neutron transport equation we obtain the multi-group equivalent of Eq.(2.2), which reads:

$$\begin{aligned}\left(\vec{\Omega}\cdot\vec{\nabla} + \Sigma^g(\vec{r},\vec{\Omega})\right)\psi^g(\vec{r},\vec{\Omega}) = \\ + \sum_{g'=1}^{N_g}\oint\frac{d\vec{\Omega}'}{4\pi}\ \Sigma_s^{g'\to g}(\vec{r},\vec{\Omega}\cdot\vec{\Omega}')\ \psi^{g'}(\vec{r},\vec{\Omega}') \\ + \frac{1}{k_{eff}}\sum_{i=1}^{N_f(\vec{r})}\chi_i^g\sum_{g'=1}^{N_g}\ \nu_i^{g'}\ \Sigma_{f,i}^{g'}(\vec{r},\vec{\Omega})\oint\frac{d\vec{\Omega}'}{4\pi}\ \psi^{g'}(\vec{r},\vec{\Omega}'),\end{aligned} \tag{2.4}$$

where the multi-group cross sections can be defined as follows, in order to preserve each reaction rate:

$$\Sigma^g(\vec{r},\vec{\Omega}) = \frac{\int_{E_g}^{E_{g-1}} dE\ \Sigma(\vec{r},E)\ \psi(\vec{r},\vec{\Omega},E)}{\int_{E_g}^{E_{g-1}} dE\ \psi(\vec{r},\vec{\Omega},E)}, \tag{2.5}$$

$$\Sigma_s^{g'\to g}(\vec{r},\vec{\Omega}\cdot\vec{\Omega}',\vec{\Omega}') = \frac{\int_{E_{g'}}^{E_{g'-1}} dE'\ \int_{E_g}^{E_{g-1}} dE\,\Sigma_s(\vec{r},E'\to E,\vec{\Omega}\cdot\vec{\Omega}')\ \psi(\vec{r},\vec{\Omega}',E')}{\int_{E_{g'}}^{E_{g'-1}} dE'\ \psi(\vec{r},\vec{\Omega}',E')}, \tag{2.6}$$

$$\chi_i^g\ \nu_i^{g'}\ \Sigma_{f,i}^{g'}(\vec{r},\vec{\Omega}) = \frac{\int_{E_g}^{E_{g-1}} dE\,\chi_i(E)\ \int_{E_{g'}}^{E_{g'-1}} dE'\,\nu_i(E')\Sigma_{f,i}(\vec{r},\vec{\Omega},E')\ \psi(\vec{r},\vec{\Omega},E')}{\int_{E_{g'}}^{E_{g'-1}} dE'\ \psi(\vec{r},\vec{\Omega},E')}. \tag{2.7}$$

Sticking to this formalism would result in a prohibitive memory size of the problem, since we would need to store group and angle dependent cross sections values. The angular dependency coming from the averaging process is in actual applications always neglected, and will not figure in the following. Moreover, these formulas show that following this procedure would require using the continuous-in-energy neutron flux as weight function in the integrals. This is of course impossible since the flux is the desired solution of the

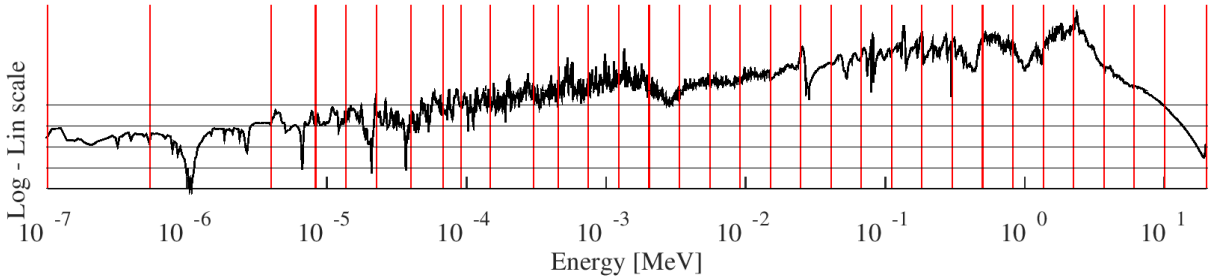


Figure 3 – Example of a 33 energy groups energy mesh (in red) superposed to a fission reaction rate spectrum for a fast reactor.

problem. From these considerations we can see that alternative ways are necessary to obtain the flux used as weight function. The choice of the energy discretization strongly influences this aspect: if the group size is small enough, the variation of the cross section within the group is negligible. In this case the multi-group cross section is almost independent of the neutron flux. A particular care has to be taken when dealing with the resonance domain. Here the cross sections variation is so important that a prohibitive number of groups would be required. To circumvent this problem, *self-shielding* methods are used. Basic principles of the self-shielding will be recalled later.

The energy dependence will be considered discretized in groups in the rest of this manuscript. If the group number is not explicitly reported, we are implicitly assuming that the equations are valid for a generic group g. Figure 3 shows an example of a 33 group energy mesh. This particular subdivision will also be presented in the results section with some complementary information.

Legendre polynomials expansion

The angular dependence of the transfer term of Eq.(2.2) is treated with a classical expansion over the Legendre polynomials. The scattering cross section is written as [2]:

$$\Sigma_s^{g'\to g}(\vec{r},\vec{\Omega}\cdot\vec{\Omega}') \simeq \frac{1}{4\pi}\sum_{k=0}^{K}\Sigma_{s,k}^{g'\to g}(\vec{r})P_k(\vec{\Omega}\cdot\vec{\Omega}'),$$

where K is the anisotropy order of the Legendre expansion and $P_k(\vec{\Omega}\cdot\vec{\Omega}')$ is the Legendre polynomial of order k:

$$P_0(x) = 1$$
$$P_k(x) = \frac{1}{2^k k!}\frac{d^k}{dx^k}(x^2-1)^k \quad for\ k = 1, 2, ...$$

The Legendre polynomials are then replaced with the real spherical harmonics:

$$P_k(\vec{\Omega}\cdot\vec{\Omega}') = \sum_{l=-k}^{l=k} A_k^l(\vec{\Omega})A_k^l(\vec{\Omega}'),$$

where the definition of the real spherical harmonics $A_k^l(\vec{\Omega})$ used in TDT is reported in [3], and reads:

$$A_k^l(\vec{\Omega}) = A_k^l(\mu,\varphi) = \begin{cases} \alpha_k^l\ P_k^l\ (\mu)\ cos(\ l\ \varphi) & l \geq 0 \\ \alpha_k^{|l|}\ P_k^{|l|}(\mu)\ sin(|l|\varphi) & l < 0, \end{cases}$$

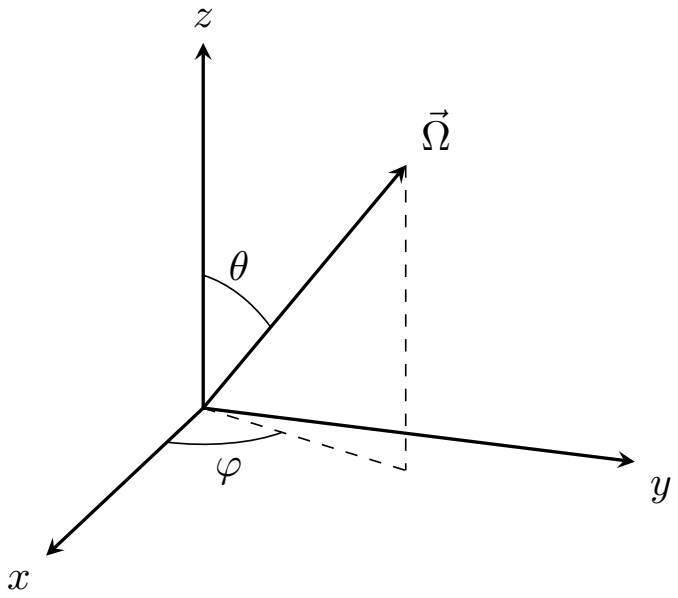


Figure 4 – Azimuthal and polar coordinates definition

where $\mu = cos(\theta)$ and the definitions of the azimuthal and polar components of the vector $\vec{\Omega}$ are given in Fig. 4 for clarity.

Replacing the transfer cross section expansion in the multi-group neutron transport equation, Eq.(2.4), we obtain:

$$\left(\vec{\Omega}\cdot\vec{\nabla}+\Sigma^{g}(\vec{r})\right)\psi^{g}(\vec{r},\vec{\Omega}) = \sum_{k=0}^{k=K}\sum_{l=-k}^{l=k} A_k^l(\vec{\Omega})\sum_{g'=1}^{N_g}\Sigma_{s,k}^{g'\to g}(\vec{r})\oint\frac{d\vec{\Omega}'}{4\pi}\; A_k^l(\vec{\Omega}')\psi^{g'}(\vec{r},\vec{\Omega}') \qquad (2.8)$$
$$+\frac{1}{k_{eff}}\sum_{i=1}^{N_f(\vec{r})}\chi_i^g\sum_{g'=1}^{N_g}\;\nu_i^{g'}\;\Sigma_{f,i}^{g'}(\vec{r})\oint\frac{d\vec{\Omega}'}{4\pi}\;\psi^{g'}(\vec{r},\vec{\Omega}').$$

From now on, we use a simplified notation for the spherical harmonics indexes (k,l), replacing them with a single index n:

$$\sum_{k=0}^{k=K}\sum_{l=-k}^{l=k} A_k^l(\vec{\Omega}) = \sum_{n=1}^{N_m} A_n(\vec{\Omega}), \qquad (2.9)$$

where $N_m = (K+1)^2$ is the total number of angular moments. As a consequence of the transfer cross section expansion, we obtain the classical definition of the angular flux moments:

$$\Phi^{g,n}(\vec{r}) = \oint\frac{d\vec{\Omega}}{4\pi}\; A_n(\vec{\Omega})\,\psi^{g}(\vec{r},\vec{\Omega}), \qquad (2.10)$$

where the 0-th order moment is referred to as *scalar flux*, and reads:

$$\Phi^{g,0} = \oint\frac{d\vec{\Omega}}{4\pi}\,\psi^{g}(\vec{r},\vec{\Omega})$$

We can see from the definition of the flux moments that Eq.(2.8) can be written in a more compact form:

$$\left(\vec{\Omega}\cdot\vec{\nabla}+\Sigma^{g}(\vec{r})\right)\psi^{g}(\vec{r},\vec{\Omega}) = q^{g}(\vec{r},\vec{\Omega}) = \sum_{n=1}^{N_m} A_n(\vec{\Omega})\,q^{g,n}(\vec{r}), \qquad (2.11)$$

where:

$$
\begin{aligned}
q^{g,n}(\vec{r}) &= q^{g,n}_{scatt}(\vec{r}) + \frac{1}{k_{eff}} q^{g,n}_{fiss}(\vec{r}) \\
q^{g,n}_{scatt}(\vec{r}) &= \sum_{g'=1}^{N_g} \Sigma^{g' \to g}_{s,n}(\vec{r}) \; \Phi^{g',n}(\vec{r}) \\
q^{g,n}_{fiss}(\vec{r}) &= \begin{cases} \sum_{i=1}^{N_f(\vec{r})} \chi^g_i \sum_{g'=1}^{N_g} \nu^{g'}_i \; \Sigma^{g'}_{f,i}(\vec{r}) \; \Phi^{g',0}(\vec{r}) & for \; n = 0 \\ 0 & for \; n > 0 \end{cases}
\end{aligned}
\tag{2.12}
$$

We also report here the following definition of the *fission integral*, that will be useful later:

$$
||\mathcal{F} \, \Phi(\vec{r})|| = \int_D d\vec{r} \sum_{i=1}^{N_f(\vec{r})} \sum_{g'=1}^{N_g} \nu^{g'}_i \; \Sigma^{g'}_{f,i}(\vec{r}) \oint \frac{d\vec{\Omega}'}{4\pi} \; \psi^{g'}(\vec{r}, \vec{\Omega}'),
\tag{2.13}
$$

which is generally used as a normalisation factor.

S_N approximation

The angular dependence is treated with the standard S_N approach. The continuous dependence of the angular variable is replaced with a set of discrete directions. The angular integrals will be numerically computed as:

$$
\oint \frac{d\vec{\Omega}}{4\pi} \, f(\vec{\Omega}) \simeq \sum_{k=0}^{N} w(\vec{\Omega}_k) \, f(\vec{\Omega}_k),
\tag{2.14}
$$

where $f(\vec{\Omega})$ is a generic function of $\vec{\Omega}$, $w(\vec{\Omega}_k)$ is the weight related to the chosen quadrature formula and N is the chosen number of directions. Figure 5 shows a set of discrete ordinates on a sphere octant. In the following we will use the analytic notation for angular integrals, assuming however that every integral is numerically computed with this approximation. This approach is typical of method of characteristics based solvers.

Space discretization

Another classical approximation used in deterministic solvers consists in dividing the computational domain in a set of regions and considering constant certain properties in each region. This approximation is generally applied to cross sections, which can be written replacing the continuous position dependence $\vec{r}$, with the region index:

$$
\Sigma^g(\vec{r}) \simeq \Sigma^g_r,
$$

Remark that if isotopic depletion were not present and the temperature profile over the region were plate, this would not be an approximation. Since this is not true, the approximation introduces errors. In a similar way, other functions can be considered constant over the regions volumes, depending on the chosen approximations. The size (and number) of the computational regions has to be chosen as a trade-off between the desired precision and the computational resources engaged. Remark also that the constant approximation is not the only one possible: an expansion over a suitable basis is also a possible choice. Once again, the type and order of the expansion will be strictly linked to the number of

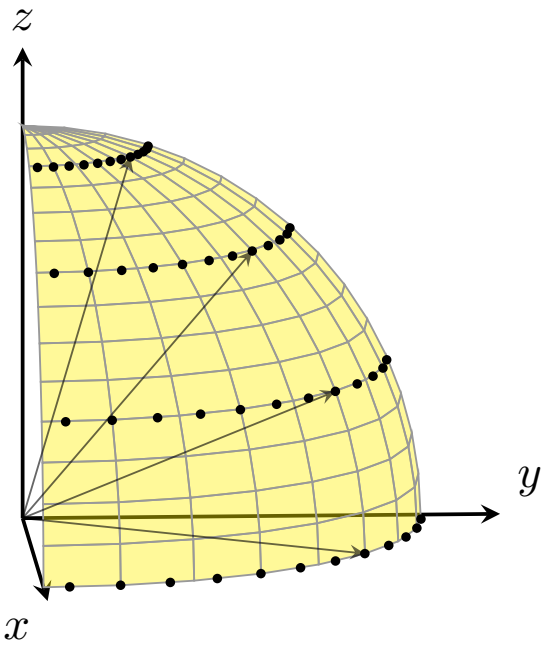


Figure 5 – Example of a set of discrete ordinates.

computational regions, and both aspects will affect the computational time and memory usage.

From now on r will indicate the region number index and every function appearing with the subscript r will be considered constant in the region.

Two-step calculations

The neutron transport equation faithfully describes the neutrons behaviour in a nuclear reactor. However, an industrial-size nuclear reactor is a large and complicated system. A precise solution of the neutron transport equation with a very fine discretization of the reactor core would result in a number of unknowns too large, even for the best available machines. For experimental facilities, with smaller size and different constraints, a direct transport solution is becoming nowadays possible, using continuous Monte Carlo or deterministic methods. In the industrial field, the general computational approach is to break down the problem in two steps. First, the so-called *lattice* calculations are performed on small but representative core sub-systems, generally fuel cells, assemblies or clusters, using fine discretizations both in space and in energy. This phase requires the application of a fictive boundary condition, such as reflection. These sub-systems calculations will then be representative of infinite systems constituted by the repetition of identical geometrical motifs. Such approximation is acceptable to describe real systems only far away from boundaries. The neutron flux obtained with this strategy is then used to compute cross sections representative of larger regions (*homogenization*) and wider energy ranges (*collapsing*), using equivalence methods that preserve the fine reaction rates values [4]. Attentive care has to be applied in lattice calculations to estimate the neutron flux in particular zones which may not be well modelled with the infinite medium condition, such as boundary or rodded assemblies. Also axial and radial reflectors need a particular treatment, since they do not contain fissile material and cannot be computed alone using a critical model. The reflectors cross sections are generally produced by computing a larger portion of the system containing also fissile assemblies, or 1D simplified models.

These *homogenized* and *collapsed* cross sections are then used for a full core calculation,

which is the second step of the traditional scheme, and it is performed either using again the transport theory, but with a lower number of groups, or lower order approximation such as diffusion theory.

This work can be seen as a contribution focused mainly on the lattice level. The purpose here is, in fact, to deliver a better 3D transport solution for geometries typical of nuclear reactor assemblies or clusters. The traditional two-step scheme, which is in practice the only one used at industrial-level, relies only on 2D or 1D calculations at the lattice level, accepting the risk of not representing the 3D effects accurately. This condition is acceptable when the geometry is regular axially and the composition not too heterogeneous, but becomes quite inaccurate for heterogeneous axial systems, such as partially inserted control rod configurations.

2.3 APOLLO3®

During this thesis, I have worked in the TDT code, within the APOLLO3® project. This project is focused on the developing of a French deterministic code, following the previous version, APOLLO2®. As for most deterministic codes, a lattice and a core code environment are present. TDT is a method of characteristics and collision probability lattice solver based on long trajectories, able to treat 2D unstructured and 3D extruded geometries.

2.3.1 Self-shielding in APOLLO3

As anticipated in the previous section, every deterministic method uses the multi-group approximation to treat the energy dependence. To obtain per-group constant cross sections values, an equivalence method in necessary. Self-shielding is a key point for deterministic methods and it is a broad and complicated topic. A correct representation of the flux in the resonance zone is crucial, since the neutron flux will suffer a variation proportional to the cross section variation, but in opposite direction. Moreover, in a group within the resonant domain of a resonant isotope, many isotopes may feature resonances very close to each other or even overlapping and the neutron flux is influenced by the combined effect of the different isotopes.

The objective of a self-shielding method is to compute the collapsed cross sections presented in Eqs.(2.5) through (2.7). The same procedure applies for the different terms of these equations, so we will take as an example only the first one, used to compute the total cross section Σ. As we have already mentioned, we neglect here the angular dependency that would be obtained using the angular flux, and we directly write the equations using the scalar flux value:

$$\Sigma^g(\vec{r}) = \frac{\int_{E_g}^{E_{g-1}} dE\ \Sigma(\vec{r},E)\ \Phi(\vec{r},E)}{\int_{E_g}^{E_{g-1}} dE\ \Phi(\vec{r},E)}, \tag{2.15}$$

The unknown here is the flux $\Phi(\vec{r},E)$. In order to treat numerically this problem, each macro-group g is finely discretized in a series of micro-group. However, in order to avoid the use of a heavy notation with a double group index, the flux $\Phi(\vec{r},E)$ in the micro-groups is indicated with a continuous dependence in energy. Remark that we are dealing with a circular problem, since the energy integral of the flux in a macro-group coincide with the original unknown of our problem defined by Eq.(2.10). As a consequence, it is impossible to obtain a really accurate value for $\Phi(\vec{r},E)$, if not iterating.

The purpose of the self-shielding method is to obtain a suitable approximation of the flux spectrum inside the macro-group, in order to compute multi-group cross sections that preserve the reaction rates values. If the flux approximation is obtained with a very simplified model, the self-shielded cross sections will faithfully preserve the reaction rates only for non-resonant isotopes or for resonant isotopes but outside their resonance domain. Unfortunately, this approximation is not sufficient, since resonances are present in several isotopes and for energy intervals that are important for fast neutrons slowing down.

Generally, isotopes are treated one at a time, while using already condensed or guess values for other isotopes cross sections. An iteration over all the resonant isotopes is performed until a stable set of cross sections is obtained, using the procedure called *Bondarenko iterations*. Since the energy discretization causes a considerable increase of the problem complexity when compared to the multi-group discretization, some drastic approximation has to be applied to the angular and spatial treatment in order to make the computation affordable. Some of the most common approximations consist in considering only isotropic scattering and using only small sub-domains, instead of the real computational geometries.

Several self-shielding methods are available in APOLLO3, but during this work the self-shielding methods were never object of any developments. The results of this work are obtained using the *Tone* method. For actual explanation about the self-shielding method, I refer to authors expert on the subject: [5] for a wide overview of different self-shielding method used in different codes and depending on the treated reactor spectrum and [6] for the *Tone* method in APOLLO3.

In a few words, the *Tone* method is based on a homogeneous-heterogeneous equivalence. The flux obtained with this method is the solution of an infinite medium problem, where the cross section has been replaced with an equivalent cross section [6]. The solution that we would have in a infinite and homogeneous problem would be:

$$\Sigma(u)\,\Phi(u) = q(u), \tag{2.16}$$

where $u = ln(\frac{E}{E_0})$ is the lethargy and $q(u)$ indicates the neutron source. If the micro-group discretization is fine enough we can safely apply the *narrow resonance* approximation, which implies that $q(u) \simeq \Sigma_p$. With this assumption we can write Eq.(2.16) for a resonant isotope x, as:

$$\Phi(u) = \frac{C_x}{\sigma_x(u) + \sigma^g_{b,x}},$$

where:

$C_x = \dfrac{\Sigma_p}{N_x}$ is a constant and N_x is the concentration of the isotope x,

$\sigma^g_{b,x} = \dfrac{1}{N_x}\displaystyle\sum_{y\neq x}\Sigma^g_y$ is the background cross section for the isotope x.

We consider next a heterogeneous system composed by a set of homogeneous regions and we write the flux in the region i with the *collision probability* formalism:

$$V_i\,\Sigma_i(u)\,\Phi_i(u) = \sum_j P_{ij}(u)\,q_j(u)\,V_j,$$

where V is the region volume and:

$P_{ij} = \dfrac{1}{V_i}\displaystyle\int_i d\vec{r}\int_j d\vec{r}\,'\,\Sigma(\vec{r}\,')\frac{e^{-\tau(\vec{r}\,',\vec{r})}}{4\pi|\vec{r}-\vec{r}\,'|^2}$ represents the probability for a neutron born in region j to have a collison in region i.

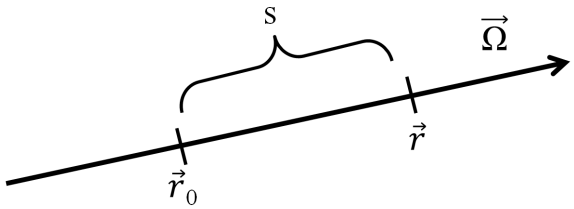


Figure 6 – Characteristic line and related variable definitions

Applying reciprocity ($P_{ij}(u)\Sigma_i(u)V_i = P_{ji}(u)\Sigma_j(u)V_j$), conservation ($\sum_j P_{ij}(u) = 1$), Tone's approximation ($P_{ij}(u) = f_i(u)^g P_{ij}^g$) and the narrow resonance approximation we obtain [6]:

$$\Phi_i(u) = \frac{\sum_j P_{ij}^g \Sigma_{p,j} V_j}{\sum_j P_{ij}^g \Sigma_j(u) V_j}. \tag{2.17}$$

Since we are doing the Bondarenko iterations, we can write the cross section for the region j as:

$$\Sigma_j(u) = N_{x,j}\sigma_x(u) + \sum_{y \neq x} \Sigma_{y,j}^g.$$

Replacing this in Eq.(2.17), we obtain:

$$\Phi_i(u) = \frac{D^g}{\sigma_x(u) + \sigma_{0,x,i}^g}, \tag{2.18}$$

where:

$$D^g = \frac{\sum_j P_{ij}^g \Sigma_{p,j} V_j}{\sum_j P_{ij}^g N_{x,j} V_j} \quad \text{is a constant,}$$

$$\sigma_{0,x,i}^g = \frac{\sum_j P_{ij}^g V_j \sum_{y \neq x} \Sigma_{y,j}^g}{\sum_j P_{ij}^g N_{x,j} V_j} \quad \text{is the heterogeneous equivalent background cross section for the isotope } x \text{ in the region } i.$$

Equation (2.18) is used to compute the unknown flux of Eq.(2.15) which acts as weighting function and the probability tables as quadrature formulas [6].

2.4 The method of characteristics

This work is entirely situated in the Method Of Characteristics (MOC) framework. We therefore recall here the MOC basics concept and formulas. The method of characteristics is a very well-known method used to solve linear partial differential equations. The idea is to integrate the equation along a characteristic line, in order to reduce the problem to a ordinary differential equation, for which it is easier to obtain the solution.

2.4.1 Integral form of the neutron transport equation

To obtain the characteristic form of the transport equation, we express the first term of Eq.(2.11) as:

$$\vec{\Omega} \cdot \vec{\nabla}\, \psi(\vec{r}, \vec{\Omega}) = \frac{\partial \psi(\vec{r}, \vec{\Omega})}{\partial x}\Omega_x + \frac{\partial \psi(\vec{r}, \vec{\Omega})}{\partial y}\Omega_y + \frac{\partial \psi(\vec{r}, \vec{\Omega})}{\partial z}\Omega_z.$$

On the other hand, the derivative with respect to the parametric variable s reads:

$$\frac{d\psi(\vec{r},\vec{\Omega})}{ds} = \frac{\partial\psi(\vec{r},\vec{\Omega})}{\partial x}\frac{dx}{ds} + \frac{\partial\psi(\vec{r},\vec{\Omega})}{\partial y}\frac{dy}{ds} + \frac{\partial\psi(\vec{r},\vec{\Omega})}{\partial z}\frac{dz}{ds}.$$

We can then identify the following terms:

$$\begin{aligned}
&\frac{dx}{ds} = \Omega_x \quad \rightarrow \quad x = x_0 + s\,\Omega_x \\
&\frac{dy}{ds} = \Omega_y \quad \rightarrow \quad y = y_0 + s\,\Omega_y \quad \rightarrow \quad \vec{r} = \vec{r}_0 + s\,\vec{\Omega}, \\
&\frac{dz}{ds} = \Omega_z \quad \rightarrow \quad z = z_0 + s\,\Omega_z
\end{aligned}$$

where Figure 6 shows the meaning of this transformation. This means that we can write Eq.(2.11) as a function of s, obtaining:

$$\frac{d\psi(\vec{r}_0 + s\,\vec{\Omega},\vec{\Omega})}{ds} + \Sigma(\vec{r}_0 + s\,\vec{\Omega})\ \psi(\vec{r}_0 + s\,\vec{\Omega},\vec{\Omega}) = q(\vec{r}_0 + s\,\vec{\Omega},\vec{\Omega}),$$

and we write it using the following abuse of notation $\vec{r}_0 + s\,\vec{\Omega} = s$, for simplicity. We also drop the dependency in $\vec{\Omega}$, getting:

$$\frac{d\psi(s)}{ds} + \Sigma(s)\ \psi(s) = q(s). \tag{2.19}$$

We start by solving the associated homogeneous problem:

$$\frac{d\psi_0(s)}{ds} + \Sigma(s)\ \psi_0(s) = 0 \qquad \rightarrow \qquad \psi_0(s) = \psi_0(0)e^{-\int_0^s ds'\ \Sigma(s')}$$

The solution of the problem will then be the sum of the associated homogeneous problem and a particular solution ψ_p of the same form:

$$\psi(s) = \psi_0(0)e^{-\int_0^s ds'\ \Sigma(s')} + \psi_p(s)e^{-\int_0^s ds'\ \Sigma(s')}$$

We replace the preceding equation in Eq.(2.19) and we obtain the particular solution ψ_p:

$$\frac{d\psi_p(s)}{ds}e^{-\int_0^{s'} ds'\ \Sigma(s')} = q(s) \qquad \rightarrow \qquad \psi_p(s) = \int_0^s ds'\ q(s')e^{\int_0^{s'} ds''\ \Sigma(s'')}$$

Putting all back together and using again the explicit notation we obtain:

$$\begin{aligned}
\psi(\vec{r},\vec{\Omega}) =& \psi(\vec{r} - s\,\vec{\Omega},\vec{\Omega})\ e^{-\int_0^s ds'\ \Sigma(\vec{r}-s'\,\vec{\Omega})} \\
&+ e^{-\int_0^s ds'\ \Sigma(\vec{r}-s'\,\vec{\Omega})} \int_0^s ds'\ q(\vec{r} - s'\,\vec{\Omega})\, e^{\int_0^{s'} ds''\ \Sigma(\vec{r}-s''\,\vec{\Omega})}
\end{aligned} \tag{2.20}$$

This equation is generally interpreted and used knowing the value of the angular flux in a given point $\vec{r}_0 = \vec{r} - s\,\vec{\Omega}$ and computing how the flux changes along a line as a consequence of removal and production. We will see in the next chapter how the equation is used in the code and its numerical equivalent.

2.4.2 Applications of the method of characteristics in neutronics

The method of characteristics is one of the most widespread methods used to solve the mono-group neutron transport equation for the reactor lattice calculations. This method is focused on the use of the integral transport equation, Eq.(2.20), to compute the angular flux across characteristic lines across the whole computational domain.

The first application of the method of characteristics for neutron transport calculations goes back to the seventies. Following Askew's work [7], this method was introduced in the English code CACTUS [8]. The method of characteristics showed the possibility to potentially treat every kind of geometry which can be represented in a Cartesian frame reference, since it does not require any particular regularity. It also allows an arbitrary anisotropy treatment and it is able to correctly catch the streaming effects typical of high absorbing media. The streaming term $\vec{\Omega} \cdot \vec{\nabla}\psi$ of the transport equation is treated exactly using the MOC formalism, in contrast to what happens using for example the finite difference approach, making it appealing for neutron transport calculations [9]. Many applications of the method of characteristics were developed starting from the 1980s. To cite some examples: Alcouffe and Larsen in 1981 [10], Filippone, Woolf, and Lavigne in 1981 [11], Suslov in MCCG3D code in 1993, [12], Knott and Edenius in CASMO-4 code in 1993 [13], Cho and Hong, CRX code in 1996 [14], Roy, DRAGON V3 code [15] and others.

Approximately until this period methods development was focused on two dimensional geometries. Most of these methods were based on *cyclic tracking*. The idea of *cyclic tracking* is that choosing particular tracking angles and applying the geometrical movement associated to the geometry symmetries to the characteristic lines, a trajectory will eventually return at the starting point after a certain length. This method allows an exact treatment of boundary conditions. An alternative approach consists in replacing the geometrical motions acting on the trajectory, with an approximate *albedo-like* condition. In this case the trajectory is terminated when a border is reached. This second choice implies a simpler tracking strategy, but a worst boundary conditions representation, when exact geometrical movements are to be applied. Trajectories lengths following this approach are considerably reduced, but boundary information must be stored to simulate the appropriate boundary conditions.

In the French lattice code APOLLO2, the MOC solver named TDT was developed in the late 1990s [16]. Cyclic trajectories and albedo-like boundary conditions were investigated and eventually the exact boundary conditions path was chosen as a privileged approach, even if the other choice remains operative [17]. The later version, APOLLO3, inherits these same features. The TDT solver was implemented also in APOLLO3 and upgraded ever since.

Trajectory-based geometry dicretization

As anticipated, the basis of the method of characteristics is to exploit the integral form of the neutron transport equation, by following a neutron path along a straight line. Each trajectory is considered representative of a portion of the domain neighbouring such a line. To have a full representation of the computational domain a set of parallel trajectories will be necessary for each angular direction. A cross section area is associated to each trajectory, acting as an integration weight. The approximation will then be the more representative of the real geometry, the closer the trajectories are to each other. Figure 7 shows a graphical representation of a simple but irregular two dimensional domain discretized using a set of parallel trajectories.

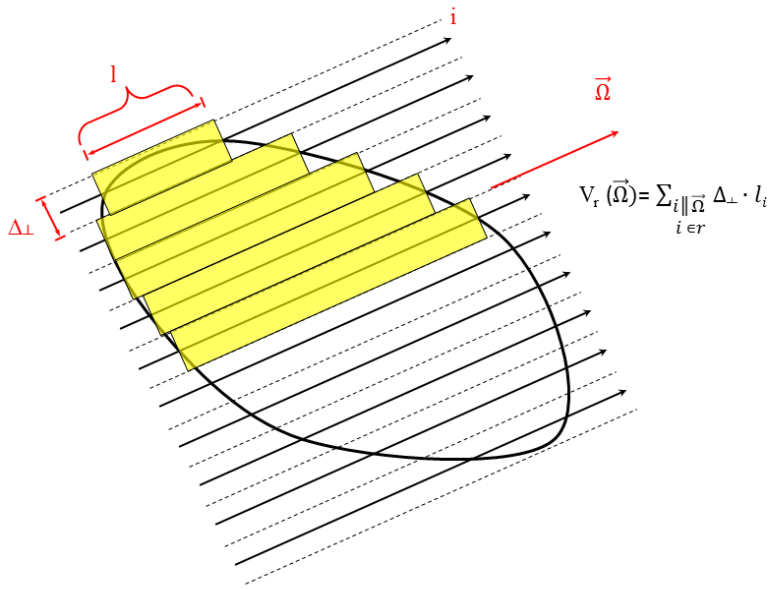


Figure 7 – Two dimensional trajectory-based discretization. In yellow the volume associated to each trajectory.

We can easily deduce from this that the numerical quantities computed with the trajectory-based discretization inherit an angular dependence. For each region it is possible to identify an analytic volume ($V_{r,a}$) or an angular volume ($V_r(\vec{\Omega})$). Averaging the angular volumes for a region over all the angles it is also possible to compute a numerical volume (V_r). For clarity, we report the formulas used to compute the angular and numerical volumes, respectively:

$$V_r(\vec{\Omega}) = \Delta_\perp(\vec{\Omega}) \sum_{\substack{t \parallel \vec{\Omega} \\ t \cap r}} l \qquad\qquad V_r = \oint \frac{d\vec{\Omega}}{4\pi} V_r(\vec{\Omega}),$$

where the sum is performed for all the trajectories parallel to the direction $\vec{\Omega}$ that cross the region r, $\Delta_\perp$ is the perpendicular integration weight associated to the trajectory-based discretization and l is the chord length and their definition is graphically represented in Fig. 7.

For a very refined tracking, the numerical quantities tend to the analytic values. However, in real applications they are always different. For small regions in particular, the numerical and analytic values can be very different and this effect must be taken into account. A possible approach to overcome this difference consists in the use of *normalized chords*. Following this approach, the length of each chord crossing a region is multiplied by the ratio of the analytic and angular volumes. As a drawback, modifying the chords lengths, the model is not any more representative of the physical distances travelled by the particles. Even if chord normalization is widely present in the literature (e.g. [5],[18] and others) and if the exact volume integration has some advantages, in TDT the preferred choice is to not normalize chords, in order to obtain the best possible physical representation of particles transport across the domain.

Long and *Short* characteristics

Two main families of MOC based methods are used to simulate neutron transport along straight lines: the *Long* characteristics and the *Short* characteristics, sometimes referred to

as SMOC. The TDT solver is based on the long characteristics approach: a set of trajectories covers the whole domain from boundary to boundary, and the particles are followed from the trajectory starting point, until the end of the line. In the short characteristics approach, the integral transport equation is used to simulate neutrons crossing each region from boundary to boundary, but for each region, the surface and volume fluxes are represented with approximated functions. In this method, fine tracking information in each region are used to compute a set of coefficients, which allow to propagate the surface fluxes from a boundary to another, without the need to treat one chord at a time. To obtain a good representation of the solution, volume and surface fluxes are generally represented through a polynomial expansion. Depending on the solution regularity and on the polynomial basis and order, the method can deliver very accurate results. The main difference between the two methods consists in how the characteristics discretization is used: in the short characteristics approach, the trajectories are used only in the coefficients computation phase. In a second phase the coefficients are used to compute the flux transmission across the domain, from surface to surface. Since the iterative strategy imposes the fluxes transmission to be repeated several times in a calculation, and since surface fluxes transmission is much cheaper than using the fine trajectory-based transmission, the short characteristics method generally presents a smaller computational cost in comparison to the long characteristics companion, but requires a higher memory storage. As a drawback, the short characteristics method entails some geometrical limitations. Until now, only plane surfaces are used to compute the surface flux expansion. As a consequence, a possible way to apply this method to generic geometries is the use of triangular (or tetrahedral) meshes. In this case, the flux on each surface can be correctly expanded using the chosen polynomial basis. However, this approach results in a large number of meshes in typical Light Water Reactors geometries. An example of the use of this approach can be found in Genesis or THOR codes [19], [20].

An alternative approach is presented in the IDT code, allowing the treatment of heterogeneous cartesian cells [21]. Thanks to this method, it is possible to treat Cartesian cells that contain a finite number of concentric circles. Even if this approach can be applied to the majority of reactor geometries, it is still not sufficient to compute a more complex geometry without using a high number of meshes to subdivide irregular components in order to represent them with plane surfaces. A second limitation of the method comes from the memory requirements: the coefficients needed are energy, angle, polynomial order and surface (or region, depending on the coefficients type) dependent. This leads to a large memory storage, which can be addressed in different ways when the geometry becomes too large, like storage on file or on-the-fly computation without storage with the use of tabulated values [22]. Another interesting feature introduced in this work allows to recognize equal cells (same geometry and same cross sections), and to compute the coefficients only once for each cell type.

The strong point of the long characteristics method is that, at least theoretically, arbitrary region shapes can be treated. The only difficulty consists in being able to identify the intersection points between a trajectory and a region boundary. In the actual implementation of TDT only lines and arcs segments are recognized as region boundaries. Still, this allows an exact representation of typical reactor geometries.

Numerical form of the integral transport equation

As anticipated in Sec. 2.4.1, the integral transport equation is the basis of the MOC based solver to simulate neutron transport. The analytic expression reported in section

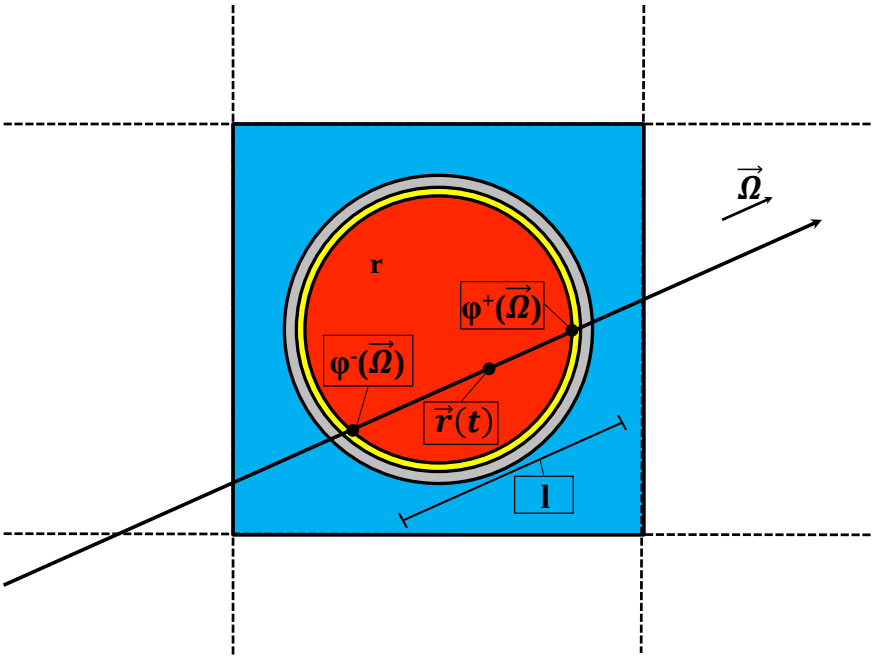


Figure 8 – Graphical representation of entering and exiting fluxes for a region along a trajectory in a simple reactor cell geometry.

2.4.1 is not however the same as the one used in actual calculations. Since the domain is discretized into a set of homogeneous regions, the cross sections are constant along a given chord. Moreover, the integral transmission equation is applied every time a trajectory enters a new region. Given this condition, the cross sections are constant between the two integration boundaries. Equation (2.20) then simplifies, becoming:

$$\psi^+(\vec{\Omega}) = \psi^-(\vec{\Omega})\, e^{-\Sigma_r\, l} + \int_0^l dt\; q\left(\vec{r}(t), \vec{\Omega}\right)\; e^{-\Sigma_r (l-t)}, \tag{2.21}$$

where the two flux values $\psi(\vec{r} - s\,\vec{\Omega})$ and $\psi(\vec{r})$ are replaced with ψ^- and ψ^+, the entering and exiting fluxes along a trajectory, respectively. We have also used t to indicate the local coordinate along the trajectory. The notations used are detailed in Figure 8. The term $q\left(\vec{r}(t), \vec{\Omega}\right)$ is also generally treated using different sort of approximations, depending on the chosen method. This will be discussed later, in particular in Chapters 3, 4 and 5.

3. The 3D MOC in APOLLO3

The two-dimensional method of characteristics has been used for a long time for the solution of the neutron transport equation in lattice calculations. The transition to three-dimensional lattice calculations had to wait for the upscale of computer architectures. Given the fast computing power increase experienced in the last decades, the 3D solution becomes more and more appealing. The method of characteristics can be applied to any type of 3D geometries, at least theoretically. In the TDT module of the French lattice code, APOLLO2, the MOC solution was available only for 2D geometries. In the APOLLO3 code, under development at CEA, it was decided to also implement a 3D MOC solver. In order to attain interesting performances, a parallel treatment has been adopted. The method of characteristics is well suited for a parallel treatment, since the solution along each trajectory can be computed independently. However, an efficient parallel strategy requires the use of a series of precautions to avoid drastic efficiency reduction. The extension of the MOC solver to 3D extruded geometries in the APOLLO3 code has been realized in the last years during a PhD work [23], which was also the object of several publications [24], [25], [26]. We recall in this chapter some elements developed during this former work, which constitutes the basis of this manuscript subject.

3.1 The Step approximation

The so-called *Step Characteristics* (SC) approximation is the most widespread way to numerically approach the MOC based solution. The idea, very popular in numerical algorithms, is to approach the desired function (here the neutron flux) with a set of constant terms. This kind of approximation is acceptable only when such a function presents small gradients in the considered domain. This translates into the reasonable concept that the finer the spatial discretization introduced, the better the solution obtained. This approximation was chosen in the first implementation of the 3D method of characteristics in TDT.

The fundamental approximation introduced when using the Step method can be expressed as:

$$\psi(\vec{r},\vec{\Omega}) \simeq \psi_r(\vec{\Omega}),$$

where the index r represents a constant value per region. This entails that also the source terms reported in Eqs.(2.12), become constant in each computational region.

Step characteristics equations

Two main equations are necessary to define the MOC transport solution: a balance and a transmission equation.

Integrating Eq.(2.11) over the volume of each region, we can obtain a balance equation to

compute the region averaged angular flux, $\psi_r(\vec{\Omega})$:

$$\frac{1}{V_r}\int_r d\vec{r}\left(\vec{\Omega}\cdot\vec{\nabla}\,\psi(\vec{r},\vec{\Omega})\right)+\frac{\Sigma_r}{V_r}\int_r d\vec{r}\;\psi(\vec{r},\vec{\Omega})=\frac{1}{V_r}\int_r d\vec{r}\;q(\vec{r},\vec{\Omega}). \tag{3.1}$$

Applying the divergence theorem to the first term we get:

$$\Sigma_r\psi_r(\vec{\Omega})=q_r(\vec{\Omega})-\frac{1}{V_r}\int_{\partial r} d\vec{r}_s\;\vec{\Omega}\cdot\hat{n}\;\psi(\vec{r}_s,\vec{\Omega}),$$

where $\hat{n}$ is the outward normal at $\vec{r}_s$. The surface integrals over the region boundary ∂r are obtained using the trajectory-based discretization, and hence decomposed as:

$$\begin{aligned}
&\frac{1}{V_r}\int_{\partial r} d\vec{r}_s\;\vec{\Omega}\cdot\hat{n}\;\psi(\vec{r}_s,\vec{\Omega})=\\
&\frac{1}{V_r}\int_{\partial r^+} d\vec{r}_s{}^+\;\vec{\Omega}\cdot\hat{n}^+\;\left[\psi(\vec{r}_s{}^+,\vec{\Omega})-\psi(\vec{r}_s{}^-,\vec{\Omega})\right]=\\
&\frac{\Delta_\perp(\vec{\Omega})}{V_r}\sum_{\substack{t\parallel\vec{\Omega}\\ t\cap r}}\left[\psi_t^+(\vec{\Omega})-\psi_t^-(\vec{\Omega})\right]=\Delta J_r(\vec{\Omega}).
\end{aligned} \tag{3.2}$$

Here the sum is performed over all the trajectories crossing the region boundary and $\psi_t^{\pm}(\vec{\Omega})$ are the exiting/entering angular fluxes along the trajectory. $\Delta_\perp(\vec{\Omega})$ represents the integration weight associated to the set of trajectories parallel to the $\vec{\Omega}$ direction. The integration weight generally coincides with the area associated to a trajectory. The chosen strategy for the 3D MOC in TDT is to use a per-angle constant trajectory spacing, in order to obtain constant weights, that can be factorized, as explained in [25]. The balance equation can finally be expressed as:

$$\Sigma_r\psi_r(\vec{\Omega})=q_r(\vec{\Omega})-\Delta J_r(\vec{\Omega}). \tag{3.3}$$

The difference between the region contribution of the exiting and entering angular fluxes is the so-called *current term* $\Delta J_r(\vec{\Omega})$ and it is computed with the *transmission* equation. A numerical transmission equation can be obtained starting from Eq.(2.21) and applying the constant per-region source approximation:

$$\psi^+(t,\vec{\Omega})-\psi^-(t,\vec{\Omega})=\left(\frac{q_r(\vec{\Omega})}{\Sigma_r}-\psi^-(t,\vec{\Omega})\right)\left(1-e^{-\Sigma_r l}\right). \tag{3.4}$$

Eq.(2.21) has been reformulated in this way for computational reasons: directly computing the difference between exiting and entering fluxes allows to minimize the number of floating-point operations, and to have better results for regions presenting a small value of the total cross section. As it is customary in MOC applications, $1-e^{-\Sigma_r l}$ is pre-tabulated in terms of $\tau=\Sigma_r l$. This strategy allows both an efficient evaluation using a linear interpolation which only demands one floating-point operation, and also a correct representation for vanishing $\Sigma_r l$ values, using Taylor's expansions.

3.1.1 Iterative strategy

The solution of the transport equation via the method of characteristics is obtained in several steps. Several nested iteration loops are used, as schematically represented by Algorithm 1. At the beginning the flux is initialized in each fissile region. The outermost

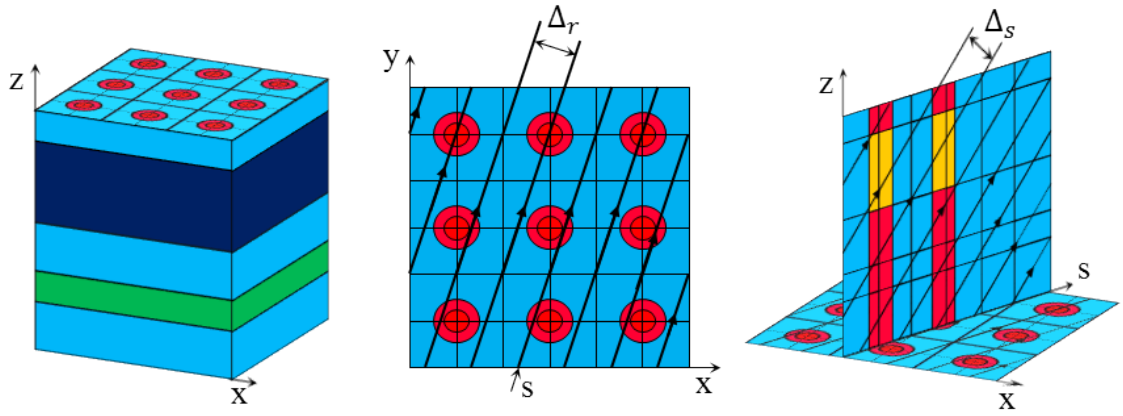


Figure 9 – Left: a simple example of a 3D extruded geometry. Center: a two-dimensional tracking associated to this geometry. Right: a visual interpretation of a set of three-dimensional trajectories, uplifted from the two-dimensional tracking. Image taken from reference [23].

iteration loop starts initializing the fission source term of Eq.(2.12) for each energy group, using the initial angular flux moments guess. Then, starting from the highest energy group, a transfer contribution coming from other groups is added to the source term. This defines the second iteration loop. The third level is constituted by the transfer within the same group. This final contribution allows to define the source term of the balance equation (Eq.(3.3)). The second and last term of this equation is computed using the trajectory sweep. The transmission equation (Eq.(3.4)) is solved for each chord of each 3D trajectory. The balance equation is used to compute the average angular flux in each region, which is used in Eq.(2.10) to estimate a new value for the moments of the flux for the considered energy group. This new evaluation is used to update the transfer term within the same group, until a converged value is obtained. This inner-most level of iteration is generally referred to as *inner iterations*. At this point it is possible to update the transfer term coming from the group just computed, and passing to the solution of the successive energy level. The procedure is repeated until the fluxes have converged in each group. At this point the new set of fluxes is used to compute again the fission source for all the energy groups, and a new *outer iteration* starts. The new eigenvalue estimation is also computed using the fission integrals of two consecutive *outer iterations*, in a classical power method. The procedure is repeated until the convergence of the angular flux moments in each group and for each region is attained [27] .

3.1.2 Chord classification method

One of the weak points of 3D geometries MOC treatment is its memory needs. As Eq.(3.4) shows, the solution of the transmission equation requires the knowledge of the chord lengths. In a typical reactor geometry the number of 3D chords can easily reach values between 10^6 and 10^9. In a classical unstructured 2D domain each chord is different from each other. As a consequence, the information must be directly stored and retrieved during the trajectory sweep. A different approach can be followed when dealing with 3D extruded geometries. As Figure 9 shows, the 3D tracking is based on the two-dimensional footprint. The intrinsic regularity of the extruded geometries allows avoiding the direct storage of all the 3D chords. The *Chord Classification Method* (CCM), introduced in [23], takes advantage of this aspect. Figure 9 also shows how, defining s the local coordinate along a 2D trajectory and z the axial coordinate, all the 3D trajectories lying on the same two-dimensional footprint belong to a vertical plane that we can define as *s-z plane*. Figure 10 represents a set of parallel trajectories for a given polar angle lying on a generic 2D line.

Initialize $\lambda = \frac{1}{k_{eff}}$ and the flux moments in fissile regions

Outer iterations:^{o}

while *the fission integral is not converged* **do**

Compute the fission source contribution for each group:

$$q_n^{g,ext} = \begin{cases} \frac{1}{k_{eff}} \sum_{i=1}^{N_f(\vec{r})} \chi_i^g \sum_{g'=1}^{N_g} \nu_i^{g'} \ \Sigma_{f,i}^{g'} \ \Phi_r^{g',0} & for\ n = 0 \\ 0 & for\ n > 0 \end{cases} \quad \text{(Eq.(2.12))}$$

Thermal iterations:

Start iterating on groups, starting from the highest energy

while *The flux in each group is not converged* **do**

Compute the mono-group solution:

Update the transfer contribution from the other groups:

$q_n^{g,ext} = q_n^{g,ext} + \sum_{g' \neq g} \Sigma_{s,n}^{g' \to g} \ \Phi_r^{g',n}$ (Eq.(2.12))

Inner iterations:^{i}

while *The flux in the g group is not converged* **do**

Compute $q_n^g = q_n^{g,ext} + \Sigma_{s,n}^{g \to g} \ \Phi_r^{g,n}$

Trajectories sweep, to compute:

$\Delta J_r(\vec{\Omega})$ (Eq.(3.2))

Compute the flux moments (Eq.(3.3) + (2.10))

end

end

Update the fission integral $\to \mathcal{F} \, \Phi_r^{o+1}$ Eq.(2.13)

Update the eigenvalue $\lambda^{o+1} = \lambda^o \frac{||\mathcal{F} \, \Phi_r^o||}{||\mathcal{F} \, \Phi_r^{o+1}||}$

end

Algorithm 1: Simplified iterative scheme

In this situation horizontal lines associated to each z-plane and vertical lines, associated to the two-dimensional tracking, generate purely Cartesian meshes. It is easy to see here that all the chords crossing two vertical surfaces belonging to the same z plane and lying on the same 2D chord, share the same length. Since they cross the same region, they also share the same optical length. This set of chords is said to belong to the same *class*. For a given 2D chord of length $l_{i,2D}$, and a given polar angle θ, the length of each 3D chord belonging to the same class is easy to obtain. The same applies to the situation where a set of chords crosses the same two successive horizontal surfaces. In this case, the information needed to retrieve the chords lengths are the plane height (Δz), and again the polar angle (θ). These two kinds of chords are defined respectively as *V-chords* and *H-chords* and their 3D lengths can be obtained with the following expressions:

$$l^V_{i,3D} = \frac{l_{i,2D}}{sin\ \theta}, \qquad\qquad l^H_{i,3D} = \frac{\Delta z}{cos\ \theta}.$$

The third and last possibility is constituted by *Mixed* chords, which are chords entering a region through a horizontal surface and exiting through a vertical one, or vice-versa. For this kind of chords it is harder to recognize a regular pattern. Some attempts have been made to classify a portion of these chords, but finally it was found that the simplest and most efficient strategy is to store their length in a *non-recognized* chord structure. This structure is composed by a series of vectors, one per each trajectory, containing the three-dimensional lengths in the same order as they are encountered along the trajectory.

Two benefits arise from the CCM: the amount of tracking information to be stored is drastically reduced and the number of floating-point operations to be performed to compute the transmission coefficients ($\beta = 1 - e^{-\Sigma_r l}$), is also reduced. During the tracking phase, the chord type must be identified in order to store only *non-recognized* chords lengths. This information must be available during the transport sweep to compute the *non-recognized* chords β coefficients, while for *classified* chords the value is pre-computed.

3.1.3 Hit surfaces sequence

The CCM method implementation requires to recognize the chord type during the transmission sweep, in order to retrieve the related pre-computed β coefficient. The idea of the method is to exploit the regularity on the *s-z plane*, which allows to keep track of the movement along the 2D footprint and in the axial planes, while sweeping the 3D trajectory. Knowing the starting axial layer and 2D chord, the type of surface that can be encountered next is either horizontal or vertical: a horizontal crossing will lead to a change of the axial layer, leaving the 2D footprint unchanged, while a vertical impact will result in passing from the present 2D chord to the next one in the 2D trajectory, leaving the axial layer unchanged. These simple considerations are the basis of another method implemented in [23], in order to exploit these regularities to further compact the tracking information and at the same time recognize the chord type during the transport sweep. This method is named *Hit Surfaces Sequence* (HSS). For each 3D trajectory, a sequence of integer values is stored. An integer in this sequence defines the number of successive surfaces of the same type crossed. If the chord type is *Mixed*, a 1 is stored. On the other hand if a sequence of at least two successive vertical or horizontal surfaces is encountered, a number >1 defines the number of impacts of the same type. Passing from a sequence of horizontal surfaces to a sequence of vertical ones, implies the presence of a *M-chord* between the two. Vertical and horizontal surfaces sequences are distinguished by the integer sign. If long sequences of *V-chords* or *H-chords* are present, this method strongly reduces the tracking storage size. Some tracking information is needed to complete the description: while sweeping the trajectory, the axial direction (*up* or *down*) must be known, in order to increase or decrease the

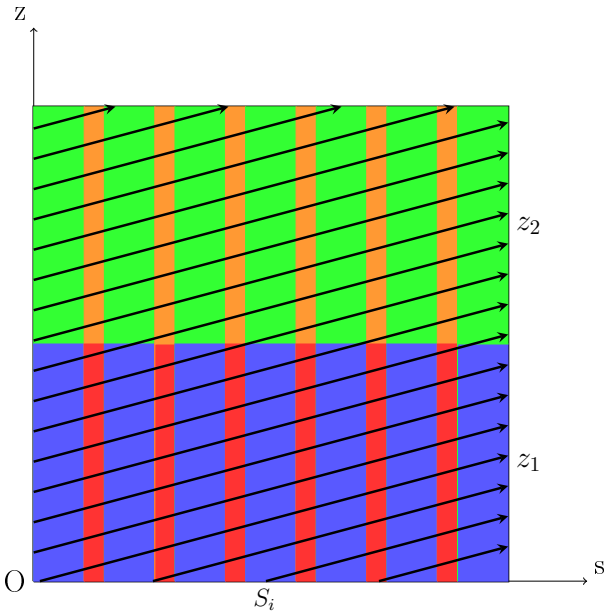


Figure 10 – A set of three-dimensional trajectories belonging to a *s-z plane*. The different colors of the two axial layers indicate different materials.

axial plane counter. As far as the 2D trajectory is concerned, the sweeping direction (*forward* or *backward*) must be known, in order to increase or decrease the 2D chord counter. Figure 11 gives an example of the information relative to two simple trajectories, written using the HSS method. A more exhaustive explanation about the CCM and HSS methods is given in [23]. We however decided to give here some elements, because the polynomial method developed in this work inherits completely the CCM and the HSS method, and benefits from both of them.

3.1.4 Parallel Strategy

The transport sweep is one of the most computationally intensive parts of the MOC solver, in particular for 3D geometries. An OpenMP-based parallel algorithm has been proposed and implemented in [23] in order to reduce the computational cost of the trajectory sweep operation, and upgraded in later code versions.

In the method of characteristics, each trajectory must be swept starting from the beginning until the end, and this operation must be performed sequentially. On the other hand, several trajectories can be swept independently by different threads, since the flux along a trajectory does not depend on the solution along the others. Once all the trajectories are swept, the flux angular moments in each region can be computed with the help of Eq.(3.3). The trajectories sweep contributes to this equation computing the difference between exiting and entering flux per each chord, and cumulating this difference in a structure that is region and angle dependent, referred to as $\Delta J_r(\vec{\Omega})$. The delicate part of the parallel treatment of the trajectory sweep mainly resides in this cumulation. Even if the flux along different trajectories can be computed independently, in the end only one $\Delta J_r(\vec{\Omega})$ term must be obtained, representative of the whole domain. As usual in parallel methods, particular care must be paid to avoid different threads attempting to modify the same value, at the same time. This situation is generally referred to as *race condition*, and represents an important drawback that can strongly decrease the performances of a parallel algorithm.

To avoid *race conditions* the proposed strategy in [23] and later enhanced in [26], consists in having private copies of $\Delta J_r(\vec{\Omega})$ for each thread. Race conditions during the trajectory

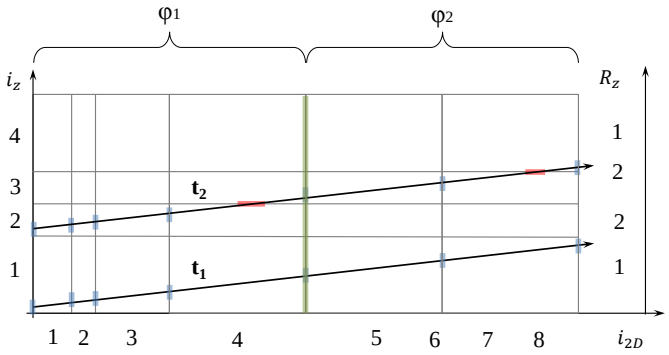


Figure 11 – Example of HSS descriptions for two simple trajectories. Two trajectories are crossing here vertical (blue) and horizontal (red) lines. On the vertical axis is indicated the axial plane number. On the right side the axial plane succession represents a reflective boundary condition, while on the left an open boundary. On the horizontal axis, the 2D chords counter. The green vertical line represents a direction change due to a horizontal boundary being reached. The information stored by the HSS method are: $HSS_1 = \{-7\}$, $HSS_2 = \{-4, 1, -2, 1, -1\}$. This image is taken from [23] and slightly modified.

sweep are eliminated and the *reduce* [1] operation on a shared $\Delta J_r(\vec{\Omega})$ is postponed after the trajectory sweep. This is a typical approach used in parallel algorithms, making use of so-called *private* and *shared* copies. The drawback of this approach consists in the memory needs: private copies of large structure are not only a problem for evident limitations in memory size, but also because duplicating large structure can result in a decrease in computational performances due to the higher memory access cost. To overcome this issue the trade-off solution used consists in the duplication of smaller portions of the $\Delta J_r(\vec{\Omega})$ term. In fact, each trajectory can only cover a subset of the directions considered by the angular quadrature formula. If no geometrical boundary conditions are used, each trajectory can only access to $\Delta J_r(\vec{\Omega})$ of a single angle.

When geometrical boundary conditions are considered, the number of angles encountered by the same trajectory increases, as we can see in Figure 12. This particularity leads to the definitions of private copies that we can call $\Delta J_r(\vec{\Omega}_{connected})$. In this case, the *reduction* from the private copies per group of connected angles to the global copy for the totality of the angles, cannot be performed any more at the end of the trajectory sweep, but each thread must *reduce* every time it passes from a sub-set of angles, to another. We remark that, however large in memory size, these private auxiliary arrays are not group dependent. As a consequence, even imagining being able to treat a domain with a million of regions, we would need about 100 Megabytes per thread. This rough estimation is meant to show that the use of thread private variables for the $\Delta J_r(\vec{\Omega})$ terms does not constitute a major memory bottleneck when applying the MOC solution.

To complete the parallel treatment of the transport sweep, a load balancing technique has been implemented, allowing to minimize the number of reductions from private to shared copies, and to distribute the amount of work among threads as equally as possible. In a few words, the cost of each trajectory (*weight*) is priorly estimated, and the trajectories are divided in *packages* of different sizes. Before dividing the trajectories in packages, they are sorted in such a way that the trajectories belonging to the same package will likely belong to the same sub-set of connected angles. This choice is made in order to minimize the

[1] The *reduction* operation usually refers to the situation when several quantities computed in parallel by independent threads must be summed or combined together.

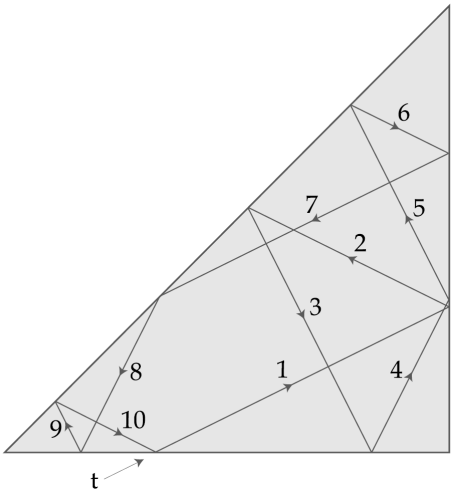


Figure 12 – Example of a cyclic track. Image taken from [17].

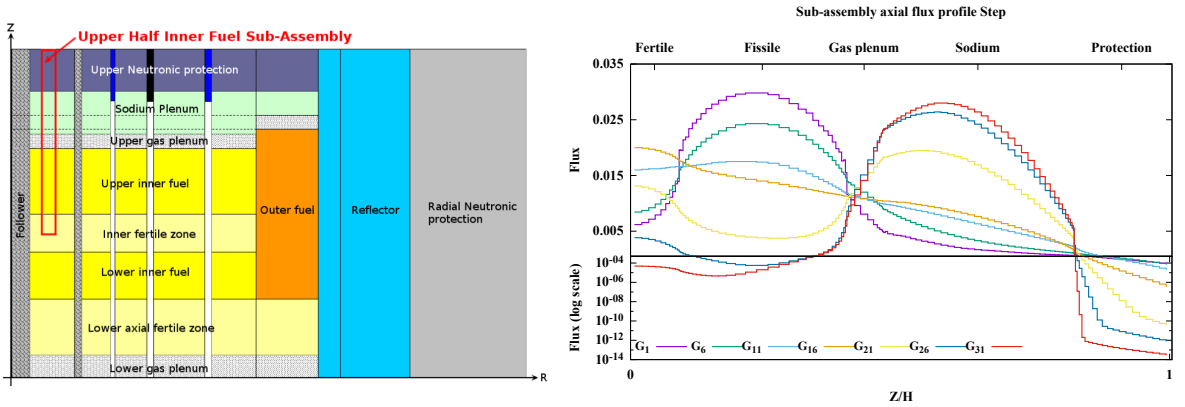
number of reductions to be performed. In this way, the thread will be able to sweep several trajectories before a reduction operation. Finally, the packages of trajectories are divided among threads. To minimize the risk to have a large package of trajectories remaining to be computed by a single thread, while the others have already finished their work, several strategies have been tested. The one eventually chosen is to order the packages by decreasing weight, after having them divided in a geometrical series fashion. Starting from the computationally heavier jobs, until the lightest, a package is assigned to a thread using a *dynamic* logic. When a thread has completed a task, a new set of trajectories with a lower associated computational cost is assigned to it. The procedure is repeated until all trajectories are swept. This approach has proven to be effective in terms of parallel efficiency and not particularly memory demanding. Dividing the charge into subsets and ordering them in decreasing computational cost is generally referred to as a *greedy algorithm*, and it is quite widespread among parallel strategies.

3.1.5 Example of results on ASTRID sub-assembly

The SC method has been validated on a heterogeneous three-dimensional assembly of the innovative Sodium-cooled Fast Breeder Reactor ASTRID (Advanced Sodium Technological Reactor for Industrial Demonstration), designed at the CEA in France. An exhaustive description of the reactor will be given later, in a dedicated section. Here we justify the choice of this benchmark by the fact that this reactor features a quite heterogeneous axial design in comparison to conventional light water reactors.

The left part of Fig.13 shows a simplified view of the ASTRID reactor. The main difference in comparison to traditional designs is due to the fact that the reactor core is divided, also in the axial direction, in different layers: fertile layers, fissile layers, sodium plenum and some others. This additional complexity makes this case a very good candidate for three-dimensional MOC solver verification. In traditional reactors, a 2D calculation gives in fact an acceptable approximation of the neutron flux far away from the top and bottom reactor boundaries. In this case, on the other hand, a two-dimensional calculation results extremely inaccurate. For this reason an assembly of this reactor was chosen to verify the accuracy obtained with the 3D MOC solver, and, at first, with the SC approximation.

The right part of Fig.13 shows an example of several axial flux profiles for a fuel pin, corresponding to different energy groups. The left side of this figure also roughly shows the axial location of the fuel assembly in the reactor. From this example we can see how



(a) Graphical representation of the axial computational domain.

(b) Axial flux profile for a fuel pin. G stands for the group number.

Figure 13 – Axial flux profile obtained with the SC method for an assembly of the ASTRID reactor. The SC calculation uses ~ 100 axial meshes. The flux are collapsed on 33 energy groups and the flux in each group is normalized, in order to present here only the flux gradients.

the flux gradients are quite steep, but also quite regular. This example also shows that a per-layer constant solution leads to the use of a very high number of axial meshes, which strongly affect the memory needs of the computation: not only the multi-group fluxes need to be duplicated for each axial plane, but also a series of auxiliary quantities of the 3D MOC solver. In particular the acceleration coefficients, which are very memory demanding, are strongly affected by the chosen axial discretization.

From these considerations arose the idea to try to express the solution using a polynomial basis. The motivations and the original ideas of this development in the TDT solver were already laid down in [23], which represents the starting point for our developments.

The introduction of this manuscript ends here. We pass now to the description of the polynomial method implemented. We have inherited the formulation given in [23] for the polynomial transport equations, which consists in the *transmission* and angular *balance* equations. Several elements had to be modified and we realized this only when confronted to the practical method implementation. In a second time, we developed and the implemented the synthetic acceleration method, that allowed us to obtain a complete and effective polynomial MOC solver.

Part II

New Developments

4. High Order MOC framework

A growing interest for three dimensional neutron transport calculations has characterized the last decades. Fed by the fast and continuous computer technology advances, 3D full core transport calculations are becoming a possible long term target for neutronic simulations. Current reactor calculations mainly rely on two-dimensional transport solutions, followed by the homogenization and collapsing of macroscopic cross sections, to be used in low order 3D methods. Even if this approach has been sufficient to design and operate nuclear reactors, there are some particular cases for which a more precise solution accounting for three dimensional transport effects may sound appealing. For the time being, a whole core transport solution in a reasonable computational time is still futuristic. However, smaller portions of the reactor core featuring particular conditions, like partially inserted control rods mainly affecting the power profile in the surrounding assemblies, can be now computed with direct transport methods without cross sections homogenization and collapsing. Safety analysis also could profit of more effective 3D calculations, as well as experimental facility simulations, for which the computational domain is generally reduced in comparison to industrial size power reactors.

Aiming at larger three dimensional computational domains could probably be accomplished only by increasing computational power, but high order methods often come to help in these situations. A large variety of high order methods based on long or short characteristics have been proposed and/or developed. We give in this section a small overview of some of them.

The first big family of high order methods regroups 2D MOC solvers using linear volume flux expansion in the x-y directions. Among these we cite the works in [28], [29], [30], [31]. In TDT a linear source expansion has also been introduced in [32] and [33], which performs a linear interpolation of surface fluxes along the trajectories. An arbitrarily high order method has also been tested in TDT for a volume flux expansion up to the fourth order [34] and a linear version has been implemented in the IDT code [21]. Since the typical distances in the radial plane are very small, the number of regions is very high due to the presence of very heterogeneous materials. This inevitably leads to a high number of unknowns, which further increases if a high order method is adopted. The aim of a high order method is to decrease the number of meshes needed, since the spatial gradients are already well represented by the high order expansion. But if a great number of meshes is imposed by the material description, a high order method may be not efficient. Literature examples can be found showing that a radial expansion above the linear term is more memory demanding than the linear counterpart, with a negative impact or a negligible gain in computational time [34].

The second family of methods is constituted by 3D MOC-based solvers with a high order flux representation. The methods belonging to this category are much more heterogeneous than their two-dimensional counterparts. Different approaches have in fact been adopted to address the three-dimensional nature of the problem. The method of characteristics can be used to directly treat 3D geometries. However, these kinds of computations are still

very expensive, both in terms of memory and of computational time. The mainstream approach is the so-called *fusion* method, where a 2D MOC traditional solution is coupled with mono-dimensional axial transport or SP_3 approximations. These kinds of methods are mainly chosen because the computational cost that they demand is fairly lower than a direct solution with a 3D MOC approach. Both the 3D MOC, and the 2D+1D *fusion* methods can benefit from a high order flux representation. For *fusion* methods these kinds of flux expansion are common. On the other hand, for 3D MOC solvers high order flux approximations are still not widespread.

The first approach that we want to cite is the OpenMOC solver developed at the MIT. The OpenMOC code is intended, as TDT, to directly solve 3D problems with the long characteristics method. Moreover, it addresses the high order problem using a similar approach to the one we have adopted, assuming a quadratic source expansion only in the axial direction [35], [36], [37].

The authors of [38] implemented a linear source expansion in x-y-z for a 3D MOC transport method. We believe this work to be a preliminary stage since only isotropic scattering was considered and no acceleration technique was proposed. Even if the results showed better performances when compared to the constant approximation, we were not able to find further developments.

The developers of the Genesis code adopted a different approach [19]. A two-dimensional conventional MOC solution is obtained in the radial plane, while the so-called LEAF (*Legendre polynomial Expansion of the Angular Flux*) method is used for the treatment of the solution along the axial meshes. The method exploits the regularity derived by the use of extruded geometries to identify *characteristics planes*. Classical 2D lines are drawn, and, in the normal direction, axial slices lying on the 2D chords are identified. Figure 14, taken from [19], gives a visual interpretation of the concept. The surface fluxes on the vertical and horizontal surfaces are expanded using Legendre polynomial. The scalar flux and the emission density are expanded linearly in the radial plane, and up to the second order in the axial direction. The transmission between characteristics planes is computed through a set of coefficients. These are obtained using a sort of two-dimensional fine tracking on the plane, in a short characteristics fashion. The method results in a very large number of coefficients to be stored or computed on-the-fly. In order to avoid both possibilities, an interesting tabulation strategy was adopted.

The DeCART [39] and MPACT [40] codes also adopt a 2D MOC approach coupled with a 1D axial diffusion solution. For a better approximation of the axial flux an expansion over Legendre polynomials is used in both methods. The nTRACER code [41] also employs a two-dimensional modular MOC solution for each plane in which the domain is

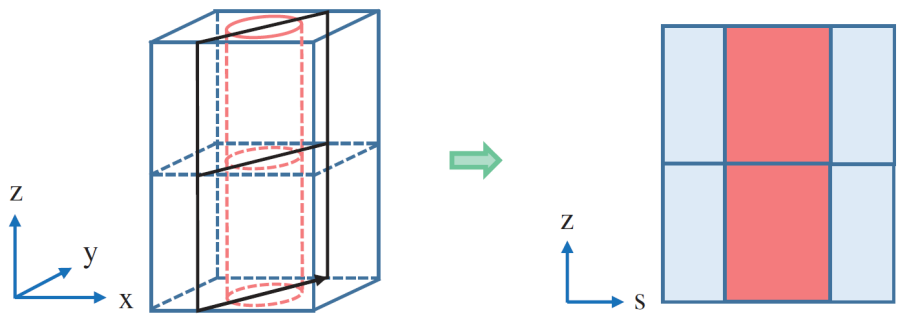


Figure 14 – Characteristic plane used in the Genesis code. Image source [19]

discretized, and the solution in each axial layer is coupled with the neighbours through a region-dependent leakage term. A nodal approach based on the SP_3 approximation is used to solve the axial problem.

Finally, the Split Cell method explained in [42] and [43] adopts a high order flux representation (linear or exponential) of the flux spatial dependence for the solution of the 3D transport equation for tetrahedral meshes. A similar approach is followed by the authors of [20], where the solution for the same 3D transport problem for tetrahedral meshes is addressed. In this work previous stability problems limiting the expansion used to the first order were overcome, and the authors present an *Arbitrarily High-Order Transport Method* for the representation of the flux spatial dependencies.

This quick overview of some of the existing methods allows identifying a clear trend. The interest in 3D calculations is rapidly growing, since they are becoming more and more feasible. Some interest is also growing for 3D whole core direct calculations. The method of characteristics is certainly a valid candidate for this task. The mainstream approach is constituted by *fusion* methods, which allow an interesting solution with reasonable computational costs. Full 3D MOC solutions, though in theory more precise, are much more expensive, and remain nowadays impracticable for very large systems, unless using very powerful parallel machines.

This work is focused on the development of a polynomial approximation in the 3D MOC solver implemented in TDT, in order to decrease the computational cost and the memory needs of the method. Conscious that this will not solve every limitation associated to 3D MOC solvers, we hope anyway that this method will allow obtaining precise solutions of larger 3D domains, when compared to the use of the SC approximation.

4.1 High order generic MOC formulation

We would like to start our technical discussion mentioning the work presented in [44], which was very useful for our developments. The author of this manuscript reports a synthetic formulation for generic high order MOC equations. We borrow some of his notations and equations for some first considerations.

The spatial dependence of the angular flux can be approximated as:

$$\psi(\vec{r},\vec{\Omega}) \sim \ \vec{f}(\vec{r}) \cdot \vec{\psi}_r(\vec{\Omega}\,), \tag{4.1}$$

$$\vec{f}(\vec{r}) = \{f_p(\vec{r}),\ p = 0, N_p\}, \qquad \vec{\psi}_r(\vec{\Omega}) = \{\psi_{r,p}(\vec{\Omega}),\ p = 0, N_p\},$$

where r is the region index, $\vec{f}$ and $\vec{\psi}_r$ are the expansion functions and the associated flux coefficients, respectively, N_p is the chosen degree for the expansion. Replacing this definition in Eq.(2.10), an alternative version of the angular flux moments is obtained:

$$\Phi^n(\vec{r}) = \vec{f}(\vec{r}) \cdot \vec{\Phi}^n_r, \qquad \vec{\Phi}^n_r = \oint \frac{d\vec{\Omega}}{4\pi}\ A_n(\vec{\Omega})\ \vec{\psi}_r(\vec{\Omega}). \tag{4.2}$$

Given these definitions, and since cross sections are considered homogeneous, the source

term defined by the set of Eqs.(2.12) can be synthetically re-written as:

$$q(\vec{r},\vec{\Omega}) = \sum_{n=1}^{N_m} A_n(\vec{\Omega})\, q^n(\vec{r}),$$

$$q^n(\vec{r}) \sim \vec{f}(\vec{r}) \cdot \vec{q}_r^{\,n}, \qquad \vec{q}_r^{\,n} = \vec{q}_{r,scatt}^{\,n} + \frac{1}{k_{eff}} \vec{q}_{r,fiss}^{\,n}, \tag{4.3}$$

where:

$$\vec{q}_{r,scatt}^{\,n} = \sum_{g'=1}^{N_g} \Sigma_{s,n,r}^{g' \to g}\ \vec{\Phi}_r^{\,g',n}, \qquad \vec{q}_{r,fiss}^{\,n} = \begin{cases} \sum_{i=1}^{N_f(\vec{r})} \chi_i \sum_{g'=1}^{N_g} \nu_i^{g'}\ \Sigma_{f,i,r}^{g'}\ \vec{\Phi}_r^{\,g',0} & for\ n=0 \\ 0 & for\ n>0. \end{cases}$$

As for any other MOC based formulation, both a transmission and a balance equation are necessary. If we express the source term of Eq.(2.21) using the expansion reported in the set of Eqs.(4.3) and defining $\vec{q}_r(\vec{\Omega}) = \sum_{n=1}^{N_m} A_n(\vec{\Omega})\ \vec{q}_r^{\,n}$, we obtain:

$$\psi^+(\vec{\Omega}) = \psi^-(\vec{\Omega})\, e^{-\Sigma_r l} + \int_0^l dt'\ \vec{f}\,[\vec{r}(t')\,] \cdot \vec{q}_r(\vec{\Omega})\ e^{-\Sigma_r (l-t')}. \tag{4.4}$$

This can also be written following the notation of [44], as:

$$\psi^+(\vec{\Omega}) = \psi^-(\vec{\Omega})\, e^{-\Sigma_r l} + \vec{E}_t \cdot \vec{q}_r(\vec{\Omega}),$$

where:

$$\vec{E}_t = \int_0^l dt'\ \vec{f}\,[\vec{r}(t')\,]\ e^{-\Sigma_r (l-t')}$$

To update the source terms we need to compute the flux expansion coefficients of Eq.(4.2), which means that we require to compute the expansion coefficients $\vec{\psi}(\vec{\Omega})$. Proceeding with the classical Galerkin approach, we write the *spatial moments* of the angular flux as:

$$'\vec{\psi}_r(\vec{\Omega}\,) = \frac{1}{V_r} \int_r d\vec{r}\ \vec{f}(\vec{r})\, \psi(\vec{r},\vec{\Omega}), \tag{4.5}$$

where the symbol " $'$ " is used to differentiate the spatial moments $'\vec{\psi}_r$, from the expansion coefficients $\vec{\psi}_r$. Using the expansion of Eq.(4.1) to express $\psi(\vec{r},\vec{\Omega})$ we obtain the expansion coefficients as a function of the moments, defining the Galerkin mass matrix $\bar{\bar{\mathcal{M}}}$ as:

$$\bar{\bar{\mathcal{M}}} = \frac{1}{V_r} \int_r d\vec{r}\, \vec{f}(\vec{r}) \otimes \vec{f}(\vec{r}) \qquad \vec{\psi}_r(\vec{\Omega}) = \bar{\bar{\mathcal{M}}}^{-1}\ '\vec{\psi}_r(\vec{\Omega}). \tag{4.6}$$

Note that $'\vec{\psi}_r \neq \vec{\psi}_r$, except if $\bar{\bar{\mathcal{M}}}$ equals to the identity matrix, i.e., when the expansion functions are region-wise orthonormalized. The author of [44] defines two possible alternatives that can be used to compute $\vec{\psi}_r(\vec{\Omega})$: using the integral or the differential form of the neutron transport equation.

Integral approach

In the integral approach, the spatial moments of the angular flux are expressed coherently with the trajectory-based space discretization, as:

$$'\vec{\psi}_r(\vec{\Omega}) = \frac{1}{V_r}\int_r d\vec{r}\;\vec{f}(\vec{r})\,\psi(\vec{r},\vec{\Omega}) = \frac{1}{V_r}\int_{\partial r_\perp} d_2 r_\perp \int_0^l dt\;\vec{f}\;[\vec{r}(t)\,]\;\psi\left[\vec{r}(t),\vec{\Omega}\right].$$

Here the volumetric integral on the region r has been decomposed as an integral over the surface perpendicular to the trajectories direction, denoted by $d_2 r_\perp$, times the integral along the trajectory local coordinate t. The angular flux along the trajectory is computed adapting Eq.(4.4) to compute the flux in a generic point t along the trajectory:

$$'\vec{\psi}_r(\vec{\Omega}\,) = \frac{1}{V_r}\int_{\partial r_\perp} d_2 r_\perp\;\vec{I}_t\,\psi^-(\vec{\Omega}) + \bar{\bar{C}}_r\,\vec{q}_r(\vec{\Omega}), \tag{4.7}$$

where:

$$\begin{aligned}
\vec{I}_t &= \int_0^l dt\;\vec{f}\;[\vec{r}(t)\,]\;\;e^{-\Sigma_r t},\\
\bar{\bar{C}}_r &= \frac{1}{V_r}\int_{\partial r_\perp} d_2 r_\perp\,\bar{\bar{C}}_t,\\
\bar{\bar{C}}_t &= \int_0^l dt\;\vec{f}\;[\vec{r}(t)\,]\otimes\int_0^t dt'\;\vec{f}\,[\vec{r}(t')\,]\;e^{-\Sigma_r(t-t')}.
\end{aligned}$$

Once the spatial angular moments $'\vec{\psi}_r(\vec{\Omega})$ are obtained, the expansion coefficients are retrieved applying Eq.(4.6).

Differential approach

The second possible way is to project the differential form of the neutron transport equation, Eq.(2.11), over the chosen expansion function $\vec{f}(\vec{r})$, obtaining an equation similar to (3.1):

$$\frac{1}{V_r}\int_r d\vec{r}\,\vec{f}(\vec{r})\left(\vec{\Omega}\cdot\vec{\nabla}\psi(\vec{r},\vec{\Omega})\right) + \frac{\Sigma_r}{V_r}\int_r d\vec{r}\,\vec{f}(\vec{r})\,\psi(\vec{r},\vec{\Omega}) = \frac{1}{V_r}\int_r d\vec{r}\,\vec{f}(\vec{r})\,q(\vec{r},\vec{\Omega}). \tag{4.8}$$

The first term can be written considering only one component p of the $\vec{f}(\vec{r})$ vector at a time, obtaining:

$$f_p(\vec{r})\left(\vec{\Omega}\cdot\vec{\nabla}\psi(\vec{r},\vec{\Omega})\right) = \vec{\nabla}\cdot\left[f_p(\vec{r})\,\vec{\Omega}\,\psi(\vec{r},\vec{\Omega})\right] - \left(\vec{\nabla}\,f_p(\vec{r})\right)\cdot\vec{\Omega}\,\psi(\vec{r},\vec{\Omega}). \tag{4.9}$$

We introduce now the following matrices:

$$\bar{\bar{\mathcal{J}}} = \vec{\nabla}\otimes\vec{f}(\vec{r}) \qquad\qquad \textit{and} \qquad\qquad \bar{\bar{\mathcal{G}}} = \vec{\Omega}\otimes\vec{f}(\vec{r}),$$

that allow us to write first term of Eq.(4.8), using Eq.(4.9) and a vectorial notation, as:

$$\frac{1}{V_r}\int_r d\vec{r}\,\vec{f}(\vec{r})\left(\vec{\Omega}\cdot\vec{\nabla}\psi(\vec{r},\vec{\Omega})\right) = \underbrace{\frac{1}{V_r}\int_r d\vec{r}\;\vec{\nabla}\;\bar{\bar{\mathcal{G}}}\,\psi(\vec{r},\vec{\Omega})}_{A} - \underbrace{\frac{1}{V_r}\int_r d\vec{r}\;\vec{\Omega}\;\bar{\bar{\mathcal{J}}}\,\psi(\vec{r},\vec{\Omega})}_{B}. \tag{4.10}$$

The divergence theorem is used for the term (A):

$$\begin{aligned} A) \rightarrow \frac{1}{V_r}\int_r d\vec{r}\ \vec{\nabla}\ \bar{\bar{\mathcal{G}}}\,\psi(\vec{r},\vec{\Omega}) &= \frac{1}{V_r}\int_{\partial r} d\vec{r}_s\ \vec{\Omega}\cdot\hat{n}\,\vec{f}(\vec{r}_s)\,\psi(\vec{r}_s,\vec{\Omega}) \\ &= \frac{1}{V_r}\int_{\partial r^+} d\vec{r}_s^{\,+}\ \vec{\Omega}\cdot\hat{n}^+\ \left[\vec{f}(\vec{r}_s^{\,+})\,\psi(\vec{r}_s^{\,+},\vec{\Omega}) - \vec{f}(\vec{r}_s^{\,-})\,\psi(\vec{r}_s^{\,-},\vec{\Omega})\right] \\ &= \frac{1}{V_r}\int_{\partial r_\perp} d_2 r_\perp\ \left[\vec{f}\left(\vec{r}^{\,+}\right)\,\psi^+(\vec{\Omega}) - \vec{f}\left(\vec{r}^{\,-}\right)\,\psi^-(\vec{\Omega})\right]. \end{aligned} \tag{4.11}$$

Here ∂r^+ denotes the exiting surface for the region r in direction $\vec{\Omega}$. $\vec{r}_s^{\,\pm}$ and $\vec{r}^{\,\pm}$ are the position vector on the exiting/entering surface.

The term (B) of Eq.(4.10) can be re-written using Eq.(4.1), obtaining:

$$B) \rightarrow \frac{1}{V_r}\int_r d\vec{r}\ \vec{\Omega}\ \bar{\bar{\mathcal{J}}}\,\psi(\vec{r},\vec{\Omega}) = \bar{\bar{\mathcal{M}}}_1(\vec{\Omega})\,\vec{\psi}_r(\vec{\Omega}) \tag{4.12}$$

$$\text{where:} \qquad \bar{\bar{\mathcal{M}}}_1(\vec{\Omega}) = \frac{1}{V_r}\int_r d\vec{r}\ \vec{\Omega}\ \left(\bar{\bar{\mathcal{J}}}\otimes\vec{f}(\vec{r})\right)$$

Putting this back together, using again Eq.(4.1) to express $\psi(\vec{r},\vec{\Omega})$ in the second term of Eq.(4.8) and Eq.(4.3) for $q(\vec{r},\vec{\Omega})$ in the third term, we get:

$$\left[\Sigma_r\,\bar{\bar{\mathcal{M}}} - \bar{\bar{\mathcal{M}}}_1(\vec{\Omega})\right]\ \vec{\psi}_r(\vec{\Omega}) = \bar{\bar{\mathcal{M}}}\,\vec{q}_r(\vec{\Omega}) - \frac{1}{V_r}\int_{\partial r_\perp} d_2 r_\perp\ \left[\vec{f}\left(\vec{r}^{\,+}\right)\,\psi^+(\vec{\Omega}) - \vec{f}\left(\vec{r}^{\,-}\right)\,\psi^-(\vec{\Omega})\right], \tag{4.13}$$

where the $\bar{\bar{\mathcal{M}}}$ matrix is the one in Eq.(4.6).

Both the *integral* and the *differential* approach have to compute the flux entering/exiting along the trajectories to obtain the surface integrals. In both cases Eq.(4.4) is used to accomplish this task. The main drawback of the integral approach is that it requires the storage of the matrices $\bar{\bar{C}}_r$. Since these are region, polynomial order and group dependent, this could result in a very large size.

Therefore, we believe the differential approach to be more efficient both in terms of memory requirements, and of number of floating point operations. For these reasons we choose to follow this path in our developments. We will refer to this chapter in the following, since our method is a particular case of the one just described.

5. Axial Polynomial transport MOC

Most common nuclear reactor designs feature a very heterogeneous two-dimensional geometry, but a quite regular axial pattern. In the radial plane, in fact, few centimeters may separate fissile, highly absorbing or moderating materials. To enhance heat transfer and to decrease the peak temperature, the fuel pin diameter must be the smallest possible. This leads to a very heterogeneous and complex geometry, which must be finely represented by simulation tools in order to obtain an acceptable precision. In the axial direction, on the contrary, the regularity is much higher. The fuel rods generally feature the same geometry from top to bottom of the reactor. The compositions inside the fuel pins may vary depending on the position, but even in this case the difference is not as important as in the radial plane. Axial heterogeneities may also be represented by spacer grids or partially inserted control rods. Even in this case the distance between two different materials is of the order of at least $\sim 10\ cm$, whereas on the radial plane the average distance between completely different materials is of the order of $1\ cm$. Figure 15 shows a simplified representation of a light water reactor core and of a fuel assembly.

Even if the material composition can be more regular in the axial direction, in comparison to the radial plane, the results obtained with the SC method during the work in [23] clearly show that a fine axial discretization is necessary in order to properly represent the flux axial gradients. As anticipated in the previous chapter, following the idea proposed in [23], we try to represent the flux axial behaviour using a set of polynomial functions. If the chosen polynomial basis is suitable for the representation of the flux gradients, this approach should result in a strong reduction of the number of the axial meshes. We proceed now with a detailed description of the axial polynomial method that we have developed.

5.1 Flux and source expansion

Choice of the polynomial basis

The polynomial basis that has been chosen to express the angular flux axial behaviour makes use of the following local coordinate:

$$\tilde{z}_r = \left(\frac{z_r - \bar{z}_r}{\Delta z_r / 2} \right) \in [-1, +1] \,,$$

where Δz_r is the height of the region r, z_r is the absolute axial coordinate and $\bar{z}_r$ is the value of the axial coordinate at the region center. With this local coordinate we define:

$$\vec{P}(\tilde{z}_r) = \{ (\tilde{z}_r)^p \,, \quad p = 0, N_p \}, \tag{5.1}$$

where N_p is the chosen order for the polynomial expansion. Thanks to this definition $(\tilde{z}_r)^p \in [-1, 1]$ for every region, independently of the region height. In this way, when passing from a region to the next one through a horizontal border, the polynomial value

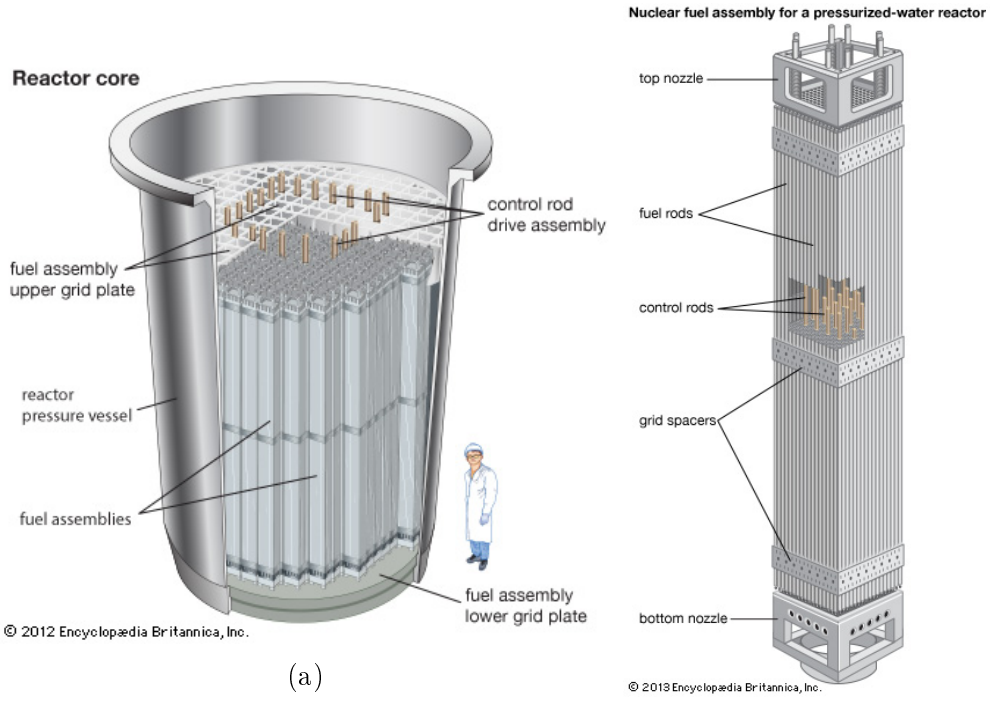


Figure 15 – Simplified view of a LWR core and of a fuel assembly.

only changes its sign, switching from $+1$ to -1 (only for odd terms). This property allows avoiding a certain number of operations during the trajectory sweep, as it will be shown later.

Angular flux expansion

We assume now that the spatial dependence of the angular flux $\psi(\vec{r},\vec{\Omega})$ can be expressed as:

$$\psi(\vec{r},\vec{\Omega}) \simeq \sum_{p=0}^{N_p} P_p(\tilde{z}_r)\,\psi_{r,p}(\vec{\Omega}) = \vec{P}(\tilde{z}_r)\cdot\vec{\psi}_r(\vec{\Omega}), \tag{5.2}$$

where $\vec{\psi}_r(\vec{\Omega})$ is a region-wise constant vector of dimension N_p+1. This expression is the equivalent of Eq.(4.1). The same procedure described in section 4.1 is repeated here, obtaining the following angular flux moments definition:

$$\Phi^n(\vec{r}) = \vec{P}(\tilde{z}_r)\cdot\vec{\Phi}^n_r, \qquad \vec{\Phi}^n_r = \oint \frac{d\vec{\Omega}}{4\pi}\, A_n(\vec{\Omega})\,\vec{\psi}_r(\vec{\Omega}), \tag{5.3}$$

and the following expression for the source term:

$$q(\vec{r},\vec{\Omega}) = \sum_{n=1}^{N_m} A_n(\vec{\Omega})\, q^n(\vec{r}), \tag{5.4}$$

$$q^n(\vec{r}) \sim \vec{P}(\tilde{z}_r)\cdot\vec{q}^n_r, \qquad \vec{q}^n_r = \vec{q}^n_{r,scatt} + \frac{1}{k_{eff}}\vec{q}^n_{r,fiss} \tag{5.5}$$

where:

$$\vec{q}^{\,n}_{r,scatt} = \sum_{g'=1}^{N_g} \Sigma^{g'\to g}_{s,n,r}\ \vec{\Phi}^{\,g',n}_{r}, \qquad \vec{q}^{\,n}_{r,fiss} = \begin{cases} \sum_{i=1}^{N_f} \chi_i \sum_{g'=1}^{N_g} \nu_i^{g'}\ \Sigma^{g'}_{f,i,r}\ \vec{\Phi}^{\,g',0}_{r} & for\ n=0 \\ 0 & for\ n>0. \end{cases} \tag{5.6}$$

Polynomial source terms in vectorial notation

We also define a vectorial notation in order to express the angular and spatial expansions in a more compact form. This notation will help to lighten the notation. Replacing Eq.(5.5) in Eq.(5.4) we obtain:

$$q(\vec{r},\vec{\Omega}) = \sum_{n=1}^{N_m} A_n(\vec{\Omega}) \sum_{p=0}^{N_p} P_p(\tilde{z}_r)\ q^n_{r,p}. \tag{5.7}$$

We introduce now a new notation:

$$\vec{\boldsymbol{\mathcal{Z}}}(\tilde{z},\vec{\Omega}) = \{A_0(\vec{\Omega})P_0(\tilde{z}), A_1(\vec{\Omega})P_0(\tilde{z}), ..., A_0(\vec{\Omega})P_1(\tilde{z}), A_1(\vec{\Omega})P_1(\tilde{z}), ...\}, \tag{5.8}$$

$$\vec{\boldsymbol{q}} = \{q^n_p\} = \{\underbrace{q^0_0, q^1_0, q^2_0, ...}_{p=0}, \underbrace{q^0_1, q^1_1, q^2_1, ...}_{p=1}, \underbrace{q^0_2, q^1_2, q^2_2, ...}_{p=2}, ...\}. \tag{5.9}$$

Here and in the following a bold letter will indicate a vector or a matrix, where the angular and spatial dimensions are collapsed in a single dimension, equal to $N_m \times (N_p+1)$. To avoid misunderstandings in the following we will use:

$\vec{x}$	mono-dimensional vector (space or angle)
$\bar{\bar{\mathcal{B}}}$	matrix associated to a mono-dimensional space (space or angle)
$\vec{\boldsymbol{x}}$	bi-dimensional vector (space and angle)
$\boldsymbol{\mathcal{B}}$	matrix associated to a bi-dimensional space (space and angle)

With this new notation we can re-write Eq.(5.7) as:

$$q(\vec{r},\vec{\Omega}) = \vec{\boldsymbol{\mathcal{Z}}}(\tilde{z},\vec{\Omega}) \cdot \vec{\boldsymbol{q}}_r. \tag{5.10}$$

In the same way we can write the angular flux moments defined by (5.3) using this vectorial form, as: $\vec{\boldsymbol{\Phi}}_r = \{\Phi^n_{r,p}\}$, which has also dimensions equal to $N_m \times (N_p+1)$.

To comply with the iterative solution algorithm, the self-scattering term is generally separated from the transfer from other groups, while the latter is regrouped with the fission source in an *external* term [45]. With this purpose, it will be useful for the next sections to reformulate the source term and, in particular, the self-scattering using the vector notation defined in Eqs.(5.8) and (5.9). Equation (5.5) then becomes:

$$\begin{aligned} &\vec{\boldsymbol{q}}_r = \vec{\boldsymbol{q}}_{\,r,scatt} + \frac{1}{k_{eff}} \vec{\boldsymbol{q}}_{\,r,fiss} = \vec{\boldsymbol{q}}^{\,in}_r + \vec{\boldsymbol{q}}^{\,ext}_r, \\ &\vec{\boldsymbol{q}}^{\,in}_r = \boldsymbol{\Sigma}^g_{r,s}\, \vec{\boldsymbol{\Phi}}_r \quad with \quad \boldsymbol{\Sigma}^g_{r,s} = \underbrace{\bar{\bar{I}}_d}_{\#(N_p+1)^2} \otimes \underbrace{\bar{\bar{\Sigma}}^{g\to g}_{r,s}}_{\#N_m^2}, \end{aligned} \tag{5.11}$$

where $\vec{\boldsymbol{q}}^{\,ext}_r$ gathers scattering coming from other groups and fission, $\bar{\bar{I}}_d$ is an identity matrix and # indicates the total matrices size. As a result, $\boldsymbol{\Sigma}^g_{r,s}$ is a diagonal matrix acting only on angular moments.

We introduce also a slightly different formulation useful in certain cases, obtained by simply interchanging the sums in Eq.(5.7):

$$q(\vec{r},\vec{\Omega}) = \vec{P}(\tilde{z}_r)\cdot\vec{q}_r(\vec{\Omega}) \qquad \text{where:} \qquad q_{r,p}(\vec{\Omega}) = \sum_{n=1}^{N_m} A_n(\vec{\Omega})\, q^n_{r,p} \tag{5.12}$$

5.2 Angular balance equation

In order to update the source terms defined by Eq.(5.6), we need to compute the flux moments defined by Eq.(5.3). A possible way to obtain the angular flux expansion coefficients $\vec{\psi}_r(\vec{\Omega})$ that this equation requires is to follow a similar procedure to the one reported in section 4.1, and labelled as *differential* approach.

We start by defining the polynomial moments of the angular flux as:

$$'\vec{\psi}_r(\vec{\Omega}) = \frac{1}{V_r}\int_r d\vec{r}\ \vec{P}(\tilde{z}_r)\,\psi(\vec{r},\vec{\Omega}). \tag{5.13}$$

Once again, we project the differential form of the neutron transport equation, Eq.(2.11), over the chosen expansion functions $\vec{P}(\tilde{z}_r)$, obtaining:

$$\frac{1}{V_r}\int_r d\vec{r}\ \vec{P}(\tilde{z}_r)\left(\vec{\Omega}\cdot\vec{\nabla}\psi(\vec{r},\vec{\Omega})\right) + \frac{\Sigma_r}{V_r}\int_r d\vec{r}\ \vec{P}(\tilde{z}_r)\,\psi(\vec{r},\vec{\Omega}) = \frac{1}{V_r}\int_r d\vec{r}\ \vec{P}(\tilde{z}_r)\,q(\vec{r},\vec{\Omega}). \tag{5.14}$$

The first term is modified in the same way as in Eq.(4.11) and Eq.(4.12):

$$\begin{aligned}&\frac{1}{V_r}\int_r d\vec{r}\ \vec{P}(\tilde{z}_r)\left(\vec{\Omega}\cdot\vec{\nabla}\psi(\vec{r},\vec{\Omega})\right) = \\ &\frac{1}{V_r}\int_{\partial r_\perp} d_2r_\perp\left[\vec{P}\left(\tilde{z}_r^+\right)\ \psi^+(\vec{\Omega}) - \vec{P}\left(\tilde{z}_r^-\right)\ \psi^-(\vec{\Omega})\right] + \frac{1}{V_r}\int_r d\vec{r}\ \vec{\Omega}\ \bar{\bar{\mathcal{J}}}\,\psi(\vec{r},\vec{\Omega}),\end{aligned}$$

Since the only non-zero component of $\bar{\bar{\mathcal{J}}} = \vec{\nabla}\otimes\vec{P}(\tilde{z}_r)$ is the one along z, this term can be simplified in comparison with the general three-dimensional case:

$$\vec{\Omega}\ \bar{\bar{\mathcal{J}}}\,\psi(\vec{r},\vec{\Omega}) = \frac{\partial\vec{P}(\tilde{z})}{\partial z}\,\Omega_z\ \psi(\vec{r},\vec{\Omega}).$$

We then use the following property:

$$\frac{\partial P_p}{\partial z} = \frac{p}{(\Delta z_r/2)}P_{p-1},$$

$$\text{and we obtain:} \qquad \frac{\partial P_p(\tilde{z})}{\partial z}\,\Omega_z\ \psi(\vec{r},\vec{\Omega}) = \mu\frac{p}{(\Delta z_r/2)}P_{p-1}(\tilde{z})\ \psi(\vec{r},\vec{\Omega}),$$

where $\mu = cos(\theta)$. Substituting back into Eq.(5.14), using the definition of the polynomial moments of the angular flux given in Eq.(5.13) and the source expansion of Eq.(5.12), we obtain a balance for the p-th component of $'\vec{\psi}_r(\vec{\Omega})$:

$$\begin{aligned}\Sigma_r\ '\psi_{r,p}(\vec{\Omega}) = &\left(\bar{\bar{\mathcal{P}}}\ \vec{q}_r(\vec{\Omega})\right)_p \\ &-\frac{1}{V_r}\int_{\partial r_\perp} d_2r_\perp\left[P_p\left(\tilde{z}_r^+\right)\ \psi^+(\vec{\Omega}) - P_p\left(\tilde{z}_r^-\right)\ \psi^-(\vec{\Omega})\right] + \mu\frac{p}{(\Delta z_r/2)}\ '\psi_{r,p-1}(\vec{\Omega}),\end{aligned}$$

where the $\bar{\bar{\mathcal{P}}}$ matrix is the equivalent of Eq.(4.6), and reads:

$$\bar{\bar{\mathcal{P}}} = \frac{1}{V_r} \int_r d\vec{r}\ \vec{P}(\tilde{z}_r) \otimes \vec{P}(\tilde{z}_r). \tag{5.15}$$

This balance equation can be solved starting from the $p = 0$ order, for which the term $'\psi_{r,p-1}(\vec{\Omega})$ is zero, until $p = N_p$, as a triangular system of equations. It can also be re-written in a vector form as:

$$\Sigma_r\ '\vec{\psi}_r(\vec{\Omega}) = \bar{\bar{\mathcal{P}}}\ \vec{q}_r(\vec{\Omega}) - \Delta\vec{J}_r(\vec{\Omega}) + \mu\ \bar{\bar{\mathcal{C}}}_r\ '\vec{\psi}_r(\vec{\Omega}), \tag{5.16}$$

where:

$$\begin{aligned}\Delta\vec{J}_r(\vec{\Omega}) &= \frac{1}{V_r} \int_{\partial r_\perp} d_2 r_\perp \left[\vec{P}\left(\tilde{z}_r^+\right)\ \psi^+(\vec{\Omega}) - \vec{P}\left(\tilde{z}_r^-\right)\ \psi^-(\vec{\Omega})\right] \\ &\simeq \frac{\Delta_\perp(\vec{\Omega})}{V_r} \sum_{\substack{t \parallel \vec{\Omega} \\ t \cap r}} \left[\vec{P}\left(\tilde{z}_r^+\right)\ \psi^+(t,\vec{\Omega}) - \vec{P}\left(\tilde{z}_r^-\right)\ \psi^-(t,\vec{\Omega})\right].\end{aligned}$$

$\Delta\vec{J}_r(\vec{\Omega})$ is the polynomial equivalent of the SC version defined by Eq.(3.2). The integration weight $\Delta_\perp(\vec{\Omega})$, and the sum over the trajectories parallel to the direction $\vec{\Omega}$ share the same meaning, while the $\bar{\bar{\mathcal{C}}}_r$ matrix reads:

$$\bar{\bar{\mathcal{C}}}_r = \frac{2}{\Delta z_r} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & \ddots & \ddots \end{bmatrix}. \tag{5.17}$$

Equation (5.16) can be used to compute the polynomial moments of the angular flux $'\vec{\psi}_r(\vec{\Omega})$ and the coefficients can be eventually retrieved by using the equivalent of Eq.(4.6):

$$\vec{\psi}_r(\vec{\Omega}\,) = \bar{\bar{\mathcal{P}}}^{-1}\ '\vec{\psi}_r(\vec{\Omega}\,). \tag{5.18}$$

5.2.1 Particle Conservation - First observations

As the author of [44] underlines in appendix B of his work, particle conservation requires a careful approach when using the balance equation derived in the previous section. As described in section 4.1, two alternative ways to obtain a balance equation are possible. However, both options make use of the integral form of the neutron transport equation to compute the entering/exiting angular fluxes along the trajectories, used to compute the surface integral of the type:

$$\frac{1}{V_r} \int_{\partial r_\perp} d_2 r_\perp\ \vec{I}_t\, \psi^-(\vec{\Omega}) \qquad \text{for the integral form} \qquad Eq.(4.7)$$

and

$$\frac{1}{V_r} \int_{\partial r_\perp} d_2 r_\perp \left[\vec{f}\left(\vec{r}^{\,+}\right)\ \psi^+(\vec{\Omega}) - \vec{f}\left(\vec{r}^{\,-}\right)\ \psi^-(\vec{\Omega})\right] \qquad \text{for the differential form} \qquad Eq.(4.11)$$

The integral approach is inherently coherent with the MOC space approximation while computing the angular fluxes along the trajectories. On the other hand, the differential

approach is coherent with the trajectory-based space discretization only if some precautions are taken.

In particular, the $\bar{\bar{\mathcal{P}}}$ matrix of Eq.(5.16) must be computed using the same space discretization introduced by the method of characteristics. If this is not the case, numerical instabilities arise, that can lead to a divergent solution. When a fine tracking is used the trajectory-based discretizations are close enough to the analytical counterparts, and the method results stable. On the other hand, if the volume approximations are not precise, which is often the case for small regions that suffer from poor discretization, the method has proven to be highly unstable. In an early stage of our work [46] we did not understand this detail, and tried instead some acrobatic stabilization techniques. Only after a careful reading of [44], we finally obtained a stable version.

The proof of this fact is postponed to Appendix A. However the only important result can be resumed in a slightly different version of Eq.(5.16), featuring a numerical version of the $\bar{\bar{\mathcal{P}}}$ matrix.

About the $\bar{\bar{\mathcal{P}}}$ matrix

The numerical matrix to be used in the angular balance equation to avoid numerical instabilities is obtained starting from the definition of $\bar{\bar{\mathcal{P}}}$ of Eq.(5.15), and writing $\tilde{z}_r$ as a function of the local coordinate along the trajectory t as:

$$\tilde{z}_r = \frac{z_r^{in} - \bar{z}_r + \mu\ t}{\Delta z_r/2} \qquad \mu = cos(\theta) \qquad t \in [0, l], \tag{5.19}$$

where z_r^{in} is the axial coordinate value at the trajectory entering point and $\bar{z}_r$ is the average value in the region r.

The (pp') element of the numerical counterpart of Eq.(5.15) then reads:

$$\begin{aligned}
\bar{\bar{\mathcal{P}}}_{r,pp'}(\vec{\Omega}) =& \frac{1}{V_r} \int_r d\vec{r}\ P_p(\tilde{z}_r)\ P_{p'}(\tilde{z}_r) \\
\simeq& \frac{1}{V_r(\vec{\Omega})} \int_{\partial r_\perp} d_2 r_\perp \int_0^l dt \left[\frac{z_r^{in} - \bar{z}_r + \mu\ t}{\Delta z_r/2} \right]^{p+p'} \\
=& \sum_{k,k'=0}^{p,p'} c_{p,k}\ c_{p',k'}\ \frac{\Delta_\perp(\vec{\Omega})}{V_r(\vec{\Omega})} \sum_{\substack{t \| \vec{\Omega} \\ t \cap r}} \left[\frac{z_r^{in} - \bar{z}_r}{\Delta z_r/2} \right]^{p+p'-k-k'} \mu^{k+k'} \left(\frac{2}{\Delta z_r} \right)^{k+k'} \frac{l^{(k+k'+1)}}{k+k'+1},
\end{aligned} \tag{5.20}$$

where $c_{p,k}$ are the binomial coefficient:

$$c_{p,k} = \frac{p!}{k!(p-k)!}.$$

The parametric angle dependency of Eq.(5.20) comes from the use of the angle-dependent integration. In the limit of analytic integration this dependence disappears. The first element of the $\bar{\bar{\mathcal{P}}}_r(\vec{\Omega})$ matrix coincides with the numerical angular volume $V_r(\vec{\Omega})$. Henceforth, the same angular volume has to be used as a denominator. Remark that computing these numerical matrices requires to store a number of square matrices with the size of the used polynomial degree, times the number of the angles, times the number of computational regions.

$\bar{\bar{\mathcal{P}}}$ Analytic limit

It is worth underlying that analytically, the $\bar{\bar{\mathcal{P}}}_r(\vec{\Omega})$ matrix does not depend on the angle:

$$\begin{aligned}\bar{\bar{\mathcal{P}}}_r(\vec{\Omega}) &= \frac{1}{V_r}\int_r d\vec{r}\ \vec{P}(\tilde{z})\otimes\vec{P}(\tilde{z}) \\ &= \frac{1}{\Delta z_r}\int_{\bar{z}-\frac{\Delta z_r}{2}}^{\bar{z}+\frac{\Delta z_r}{2}} dz\left(\frac{z_r-\bar{z}_r}{\Delta z_r/2}\right)^{p+p'} = \bar{\bar{\mathcal{P}}}_a,\end{aligned}$$

where the analytic matrix reads:

$$\bar{\bar{\mathcal{P}}}_{a,pp'} = \frac{1}{\Delta z_r}\int_{\bar{z}-\frac{\Delta z_r}{2}}^{\bar{z}+\frac{\Delta z_r}{2}} dz\left(\frac{z_r-\bar{z}_r}{\Delta z_r/2}\right)^{p+p'} = \begin{cases}\frac{1}{p+p'+1} & for\ p+p'\ even\\ 0 & for\ p+p'\ odd\end{cases} \tag{5.21}$$

For $N_p = 3$, for example:

$$\bar{\bar{\mathcal{P}}}_{a,pp'} = \begin{bmatrix} 1 & 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{5} \\ \frac{1}{3} & 0 & \frac{1}{5} & 0 \\ 0 & \frac{1}{5} & 0 & \frac{1}{7} \end{bmatrix}.$$

Stable angular balance equation

Replacing the $\bar{\bar{\mathcal{P}}}_r(\vec{\Omega})$ definition of Eq.(5.20) in the angular balance equation previously obtained, we get:

$$\Sigma_r\ {}'\vec{\psi}_r(\vec{\Omega}) = \bar{\bar{\mathcal{P}}}_r(\vec{\Omega})\ \vec{q}_r(\vec{\Omega}) - \Delta\vec{J}_r(\vec{\Omega}) + \mu\ \bar{\bar{\mathcal{C}}}_r\ {}'\vec{\psi}_r(\vec{\Omega}). \tag{5.22}$$

The $\Delta\vec{J}_r$ term also requires a slight modification as a consequence of the reasoning reported in Appendix A. Once again the use of the numerical angular volume $V_r(\vec{\Omega})$ is made necessary:

$$\Delta\vec{J}_r(\vec{\Omega}) \simeq \frac{\Delta_\perp(\vec{\Omega})}{V_r(\vec{\Omega})}\sum_{\substack{t\parallel\vec{\Omega}\\ t\cap r}}\left[\vec{P}(\tilde{z}_r^+)\,\psi^+(t,\vec{\Omega}) - \vec{P}(\tilde{z}_r^-)\,\psi^-(t,\vec{\Omega})\right], \tag{5.23}$$

Remark that the presence of the angular numerical volume in the $\Delta\vec{J}_r$ term makes the zero order term of this formulation different from the standard SC approximation, where generally the analytic volume is used.

Once the polynomial moments of the angular flux ${}'\vec{\psi}_r(\vec{\Omega})$ are computed, we can obtain the expansion coefficients as:

$$\vec{\psi}_r(\vec{\Omega}\,) = \bar{\bar{\mathcal{P}}}_a^{-1}\ {}'\vec{\psi}_r(\vec{\Omega}\,). \tag{5.24}$$

Contrarily to what we had at first imagined, the use of the analytic inverse matrix has proven not to cause any problem in this last step. This approach has therefore been chosen, since it avoids the storage of the angular dependent inverse matrices.

The entering/exiting angular fluxes along the trajectories of Eq.(5.23) are computed via a polynomial version of the integral form of the neutron transport equation equivalent to Eq.(4.4), that will be described in section 5.4.

As a final remark, attention must be paid since the numerical angular volume can be equal to zero if no trajectory crosses the considered region for a given angle. In this particular case there is also no contribution to the $\Delta\vec{J}_r(\vec{\Omega})$ from the trajectories sweep and no contribution to the numeric version of the $\bar{\bar{\mathcal{P}}}(\vec{\Omega})$ matrix. Equation (5.22) then simplifies and the analytic $\bar{\bar{\mathcal{P}}}_a$ is used instead of the absent numerical counter-part.

5.3 Flux moment computation

The previous section described the procedure used to obtain the polynomial moments of the angular flux ${}'\vec{\psi}_r(\vec{\Omega})$, and the expansion coefficients $\vec{\psi}_r(\vec{\Omega})$. To close our system we need to compute the angular flux moments $\vec{\mathbf{\Phi}}_r$, as in Eq.(5.3). Using the angular balance of Eq.(5.22) obliges us to compute at first the angular expansion coefficients, and then to obtain the angular flux moments as:

$$\vec{\Phi}_r^n = \oint \frac{d\vec{\Omega}}{4\pi}\, A_n(\vec{\Omega})\, \vec{\psi}_r(\vec{\Omega}).$$

An alternative way to compute the angular flux moments has been implemented, which does not requires the explicit use of the angular balance equation. This *direct* computation is slightly less computational demanding and requires about the same storage as the angular balance equation (which requires to store the $\bar{\bar{\mathcal{P}}}(\vec{\Omega})$ angular matrices). This formulation has been developed mainly because it leads to a balance equation that it is also used for the synthetic acceleration, which is the object of chapter 6. The use of the balance equation described in this section allows therefore to avoid the storage of the angular $\bar{\bar{\mathcal{P}}}(\vec{\Omega})$ matrices, and to replace them with a similar size matrices that would be anyway necessary for the DP_N synthetic acceleration.

To do this we combine Eq.(5.24) with Eq.(5.3), obtaining:

$$\vec{\Phi}_r^n = \oint \frac{d\vec{\Omega}}{4\pi}\, A_n(\vec{\Omega})\, \bar{\bar{\mathcal{P}}}_a^{-1}\, {}'\vec{\psi}_r(\vec{\Omega}).$$

Since the value of the $\bar{\bar{\mathcal{P}}}_a^{-1}$ matrix in Eq.(5.21) is independent of the angle we can write:

$$\vec{\Phi}_r^n = \bar{\bar{\mathcal{P}}}_a^{-1}\, {}'\vec{\Phi}_r^n, \qquad\qquad {}'\vec{\Phi}_r^n = \oint \frac{d\vec{\Omega}}{4\pi}\, A_n(\vec{\Omega})\, {}'\vec{\psi}_r(\vec{\Omega}), \tag{5.25}$$

where ${}'\vec{\Phi}_r^n$ will be referred to with the redundant but necessary name of *polynomial and angular flux moments*. Now Eq.(5.22) can be used to express ${}'\vec{\psi}_r(\vec{\Omega})$, obtaining:

$$\begin{aligned}
\Sigma_r\, {}'\vec{\Phi}_r^n = & \oint \frac{d\vec{\Omega}}{4\pi}\, A_n(\vec{\Omega})\, \bar{\bar{\mathcal{P}}}_r(\vec{\Omega})\, \vec{q}_r(\vec{\Omega}) + && (A)\\
& - \oint \frac{d\vec{\Omega}}{4\pi}\, A_n(\vec{\Omega})\, \Delta\vec{J}_r(\vec{\Omega}) && (B)\\
& + \oint \frac{d\vec{\Omega}}{4\pi}\, A_n(\vec{\Omega})\, \mu\, \bar{\bar{\mathcal{C}}}_r\, {}'\vec{\psi}_r(\vec{\Omega}). && (C)
\end{aligned}$$

We analyse the three terms separately. For the first one we use Eq.(5.12) to express $\vec{q}_r(\vec{\Omega})$:

$$(A) \rightarrow \quad \oint \frac{d\vec{\Omega}}{4\pi}\, A_n(\vec{\Omega})\, \bar{\bar{\mathcal{P}}}_r(\vec{\Omega}) \sum_{n'}^{N_m} A_{n'}(\vec{\Omega})\, \vec{q}_r^{\,n'} = \mathbf{\mathcal{Z}}_r\, \vec{\boldsymbol{q}}_r,$$

where we have used the definition of $\vec{\boldsymbol{\mathcal{Z}}}$ given in Eq.(5.8) to define the following spatial-angular matrix:

$$\begin{aligned}
\mathbf{\mathcal{Z}}_r &= \oint \frac{d\vec{\Omega}}{4\pi}\, \left(\vec{A}(\vec{\Omega}) \otimes \vec{A}(\vec{\Omega})\right) \otimes \bar{\bar{\mathcal{P}}}_r(\vec{\Omega}) \\
&= \oint \frac{d\vec{\Omega}}{4\pi}\, \frac{1}{V_r(\vec{\Omega})} \int_r d\vec{r}\, \vec{\boldsymbol{\mathcal{Z}}}(\tilde{z}, \vec{\Omega}) \otimes \vec{\boldsymbol{\mathcal{Z}}}(\tilde{z}, \vec{\Omega}).
\end{aligned} \tag{5.26}$$

For the third term we use a relation between spherical harmonics that is presented in Eq.(B.2) of Appendix B. We use it to express the product $\vec{A}(\vec{\Omega})\,\mu = \bar{\bar{\alpha}} \otimes \bar{\bar{\mathfrak{C}}}_r$, obtaining:

$$(C) \rightarrow \quad \oint \frac{d\vec{\Omega}}{4\pi}\, \mu\, \vec{A}(\vec{\Omega}) \otimes \bar{\bar{\mathfrak{C}}}_r\, {}'\vec{\psi}_r(\vec{\Omega}) = \tag{5.27}$$
$$\bar{\bar{\alpha}} \otimes \bar{\bar{\mathfrak{C}}}_r \oint \frac{d\vec{\Omega}}{4\pi}\, \vec{A}(\vec{\Omega})\, {}'\vec{\psi}_r(\vec{\Omega}) = \mathfrak{D}\, {}'\vec{\mathbf{\Phi}}_r,$$

where $\mathfrak{D} = \bar{\bar{\alpha}} \otimes \bar{\bar{\mathfrak{C}}}_r$. Putting all back together we get:

$$\Sigma_r\, {}'\vec{\mathbf{\Phi}}_r = \mathfrak{Z}\, \vec{\boldsymbol{q}}_r - \oint \frac{d\vec{\Omega}}{4\pi}\, \vec{A}(\vec{\Omega}) \otimes \Delta\vec{J}_r(\vec{\Omega}) + \mathfrak{D}\, {}'\vec{\mathbf{\Phi}}_r. \tag{5.28}$$

The elements of matrix $\mathfrak{D}$ can be found in appendix B. We insist here in particular on the matrix profile, since the solution of this system strongly depends on it. If we write the previous equation only for the polynomial order p and we use the definition of the $\bar{\bar{\mathfrak{C}}}_r$ matrix given in Eq.(5.17), it is easy to see that the $\mathfrak{D}$ has a simple profile:

$$\begin{aligned}\Sigma_r\, {}'\vec{\Phi}_{r,p} &= (\mathfrak{Z}\, \vec{\boldsymbol{q}}_r)_p - \frac{1}{4\pi} \oint d\vec{\Omega}\, \vec{A}(\vec{\Omega}) \otimes \left(\Delta\vec{J}_r(\vec{\Omega})\right)_p + \left(\mathfrak{D}\, {}'\vec{\mathbf{\Phi}}_r\right)_p \\ &= (\mathfrak{Z}\, \vec{\boldsymbol{q}}_r)_p - \frac{1}{4\pi} \oint d\vec{\Omega}\, \vec{A}(\vec{\Omega}) \otimes \left(\Delta\vec{J}_r(\vec{\Omega})\right)_p + \frac{p}{\Delta z_r/2}\, \bar{\bar{\alpha}}_p\ {}'\vec{\Phi}_{r,p-1}.\end{aligned} \tag{5.29}$$

In the previous equation each polynomial moment p depends on the moments of lower order $p-1$ as a consequence of the last term. Therefore , $\mathfrak{D}$ matrix is lower-diagonal for the polynomial index dependency (see Table 5.1).

Moreover, as it will result from Eq.(B.2), $\mathfrak{D}$ couples each angular (k,l) moment with the $(k+1,l)$ and $(k-1,l)$ ones (for $k>0$). Luckily, these moments belong to a lower polynomial order $p-1$ as it is shown in Eq.(5.29). This makes possible a simple solution strategy based on the observation that to compute the set of angular moments of the last polynomial order, say N_p, in Eq.(5.29) one has to know the set of angular moments of the previous N_{p-1} polynomials but augmented by those relative to the $(l+1)$ angular index. This reasoning can be carried up until the first polynomial so as to conclude that the 0-th polynomial order moments have to be computed up to the $K_{scatt} + N_p$ harmonic index, where K_{scatt} is the scattering order, i.e. the order number of Legendre expansion in the collision operator. Once one has taken this precaution, the solution of Eq.(5.29) becomes a purely lower-diagonal inversion.

Taking into account the previous observation we can then affirm that each p-th diagonal block of the $\mathfrak{D}$ matrix represented in Table 5.1 contains a number of angular moment so

$\mathfrak{D} =$

	p=0	p=1	p=2	p=3
p=0	0	0	0	0
p=1	$\bar{\bar{D}}_{0,1}$	0	0	0
p=2	0	$\bar{\bar{D}}_{1,2}$	0	0
p=3	0	0	$\bar{\bar{D}}_{2,3}$	0

Table 5.1 – Profile of matrix $\mathfrak{D}$. Only the first lower diagonal per polynomial index sub-matrix are non zero. Also, remark that the size of each $\bar{\bar{D}}_{p-1,p}$ sub-matrix is different, even if represented as equal in the figure.

as to include all moments whose first harmonic index goes until $K_{scatt} + N_p - p$. Following this rule the maximum sub-matrix dimension will read:

$$N_D = (K_{scatt} + N_p + 1)^2. \tag{5.30}$$

In appendix B we also show with a simple example which is the exact size of each sub-matrix and the angular moments needed depending on the anisotropy and polynomial orders.

It is worth noting that the situation described above is not without consequences on the polynomial basis definition (Eq.(5.8)) since now an enlarged unknown vector has to be used to include the terms arising from the supplementary coupling terms in $\mathfrak{D}$. To stress the differences we will now call $\vec{\boldsymbol{\mathcal{Z}}}_D(\tilde{z}, \vec{\Omega})$ (where the subscript "D" stands for $\mathfrak{D}$ coherent quantities) the vector of Eq.(5.8), but with the angular terms needed by this balance formulation.

Having this in mind we can write:

$$\Sigma_r \, {}'\vec{\boldsymbol{\Phi}}_{r,D} = \boldsymbol{\mathcal{Z}}_{DV} \, \vec{\boldsymbol{q}}_r - \frac{1}{4\pi} \oint d\vec{\Omega} \; \vec{A}_D(\vec{\Omega}) \otimes \Delta \vec{J}_r(\vec{\Omega}) + \mathfrak{D} \, {}'\vec{\boldsymbol{\Phi}}_{r,D}, \tag{5.31}$$

where the $\boldsymbol{\mathcal{Z}}_{DV}$ is given by the analogue of Eq.(5.26) but with the tensorial product of $\vec{\boldsymbol{\mathcal{Z}}}_D \otimes \vec{\boldsymbol{\mathcal{Z}}}_V$ replaced inside and all vectors have an elongated dimension, except the source term, which is limited by the scattering and fission operators.

Once the system (5.31) is solved and all the necessary angular moments are obtained, only the moments belonging to the true scattering dimension ($\#N_m$) are kept (denoted by $\vec{\boldsymbol{\Phi}}_{r,V}$) and used to obtain the *polynomial coefficients of the angular flux moments* via Eq.(5.25):

$$\vec{\boldsymbol{\Phi}}_r = \boldsymbol{\mathcal{P}}_a^{-1} \, {}'\vec{\boldsymbol{\Phi}}_{r,V}, \tag{5.32}$$

where $\boldsymbol{\mathcal{P}}_a^{-1}$ is the equivalent of $\bar{\bar{\mathcal{P}}}_a^{-1}$, but the full angular-spatial dimension $(N_p + 1 \times N_m)^2$:

$$\boldsymbol{\mathcal{P}}_a^{-1} = \bar{\bar{\mathcal{P}}}_a^{-1} \otimes \underbrace{\bar{\bar{I}}_d}_{\#N_m^2}.$$

The balance formulation presented in Eq.(5.31), coupled with Eq.(5.32), will be adapted for the DP_N synthetic acceleration.

5.4 Transmission Equation

As anticipated in Section 5.2, we need to compute the entering/exiting $\psi^-(t, \vec{\Omega}), \psi^+(t, \vec{\Omega})$ angular fluxes along the trajectories, in order to cumulate the $\Delta \vec{J}_r(\vec{\Omega})$ term of Eq.(5.22). We explicit here the formulation that we have adopted for the transmission equation used to compute the angular flux along the trajectories. Substituting the source expansion reported in Eq.(5.12), into the generic transmission equation, Eq.(2.21), we obtain an equivalent of the generic high order transmission reported in Eq.(4.4):

$$\psi^+(\vec{\Omega}) = \psi^-(\vec{\Omega}) \, e^{-\Sigma_r \, l} + \int_0^l dt' \; \vec{P}\left[\tilde{z}_r(t')\right] \; \vec{q}_r(\vec{\Omega}) \; e^{-\Sigma_r (l - t')}. \tag{5.33}$$

To express the value of the polynomial basis along the trajectory we use the same expression of Eq.(5.19), and we obtain:

$$\psi^+(\vec{\Omega}) = \psi^-(\vec{\Omega}) \, e^{-\Sigma_r \, l} + \int_0^l dt' \; \sum_{p=0}^{N_p} \frac{(z_r^{in} - \bar{z}_r + \mu \; t')^p}{(\Delta z/2)^p} \; q_{r,p}(\vec{\Omega}) \; e^{-\Sigma_r (l - t')}.$$

Developing now the p-th power term we get:

$$\int_0^l dt' \sum_{p=0}^{N_p} \left[\sum_{k=0}^{p} \frac{p!}{k!(p-k)!} \left(\frac{z_r^{in} - \bar{z}_r}{\Delta z/2} \right)^k \mu^{p-k} \, t'^{p-k} \left(\frac{2}{\Delta z} \right)^{p-k} \right] q_{r,p}(\vec{\Omega}) \, e^{-\Sigma_r (l-t')}$$
$$= \sum_{p=0}^{N_p} \left[\sum_{k=0}^{p} c_{p,k} \, P_k(z_r^{in}) \, \mu^{p-k} \left(\frac{2}{\Delta z} \right)^{p-k} \frac{1}{\Sigma_r^{(p-k)}} \int_{\tau(0)}^{\tau(l)} d\tau' \, \tau'^{p-k} \, e^{(\tau' - \tau(l))} \right] \frac{q_{r,p}(\vec{\Omega})}{\Sigma_r},$$

where the *optical length* $\tau(t)$ is defined as:

$$\tau(t) = \Sigma_r \, t.$$

The transmission equation obtained reads:

$$\psi^+(\vec{\Omega}) = \psi^-(\vec{\Omega}) \, e^{-\Sigma_r l} + \sum_{p=0}^{N_p} \left[\sum_{k=0}^{p} c_{p,k} \, P_k(z_r^{in}) \, \mu^{p-k} \left(\frac{2}{\Delta z} \right)^{p-k} E_{p-k}(\tau) \right] \frac{q_{r,p}(\vec{\Omega})}{\Sigma_r}, \quad (5.34)$$

where the *escape coefficients* $E_{p-k}(\tau)$ are defined as:

$$E_{p-k}(\tau) = \frac{1}{\Sigma_r^{(p-k)}} \int_{\tau(0)}^{\tau(l)} d\tau' \, \tau'^{p-k} \, e^{(\tau' - \tau(l))}. \quad (5.35)$$

5.4.1 Computation of the escape coefficients

The *escape coefficients* needed to complete the transmission sweep could be computed directly by integrating Eq.(5.35). The standard approach used in TDT for *optical-length*-dependent coefficients is to use interpolation tables, since it has been proven to be fast and reliable also for vanishing chords or vacuum media. Moreover, as already presented in [23], integrating by part Eq.(5.35), we can obtain the useful recursive relation:

$$E_b(\tau) = l^b - \frac{b}{\Sigma_r} E_{b-1}(\tau) \qquad 1 \leq b \leq N_p. \quad (5.36)$$

One of the possible approaches is then to start from the 0-th order coefficient and to retrieve the others using a *forward approach*:

$$\begin{aligned} E_0(\tau) &= 1 - e^{-\tau}, \\ E_1(\tau) &= l - \frac{1}{\Sigma_r} + \frac{1}{\Sigma_r} e^{-\tau}, \\ &\vdots \\ E_b(\tau) &= l^b - \frac{b}{\Sigma_r} E_{b-1}(\tau) \qquad 1 \leq b \leq N_p. \end{aligned}$$

This would allow the use of only one interpolation table for the 0-th order value. Unfortunately the recursive relation Eq.(5.36) is ill-conditioned for small values of τ. Another possible approach that the author of [23] proposes, is to tabulate the following function:

$$E_b^T(\tau) = \frac{1}{\tau^b} \int_{\tau(0)}^{\tau(t)} d\tau' \; (\tau')^b \; e^{(\tau' - \tau(t))}, \quad (5.37)$$

and then retrieve the *escape coefficients* as:

$$E_b(\tau) = l^b \; E_b^T(\tau). \quad (5.38)$$

A visual representation of the function expressed by Eq.(5.37) is given in Fig.16. Moreover, after integrating by parts Eq.(5.37) it is possible to obtain also in this case recursive relations reading:

$$forward \rightarrow \quad E_b^T(\tau) = 1 - \frac{b}{\tau} E_{b-1}^T(\tau), \tag{5.39}$$

$$backward \rightarrow \quad E_b^T(\tau) = \frac{\tau}{b+1} \left[1 - E_{b+1}^T(\tau)\right]. \tag{5.40}$$

Using this approach would allow us to tabulate only the highest-order coefficient, and to retrieve the other ones by using the backward relation, which happens to be well-conditioned for small τ values. Even if this approach would solve the problem for small τ values, it suffers of numerical cancellation for large τ. The backward relation is in fact ill-conditioned in this case. Each *escape coefficient* tends asymptotically to 1, so using the backward relation would cause important numerical cancellation when performing the $1 - E_{b+1}^T(\tau)$ operation. The errors propagation associated to the forward and backward relations are shown in Fig.17 and Fig.18. In order to simulate the behaviour of the interpolation table, a small relative error ($O(10^{-9})$) is associated to the highest or to the lowest order coefficients, depending on the chosen strategy. The other coefficients are then computed via Eqs.(5.39) and (5.40) and the figures show the sensible τ intervals where the recurrent relations do not deliver accurate results.

Based on the analysis of these results, we have decided that constructing a table for each polynomial term was the most effective method. From the computational point of view, performing a linear interpolation for each polynomial degree to compute $E_b^T(\tau)$ and then applying Eq.(5.38) to obtain the true *escape coefficient* is not more computationally expensive than applying one of the two recurrent relations. As far as the memory is concerned, we do not believe that having several tables represents an important problem. The τ domain is in fact limited to some decades and Fig.16 assures us that increasing the polynomial degree will not result in needing a larger number of points, since the higher order coefficients are smoother and smoother. Moreover, using just one table requires a very high number of points, in order to avoid a small error to be propagated by one of the two recurrent relations. The single table approach was at first implemented, but eventually discarded.

To allow fast and simple memory access to the tabulated values, we used constant step tables. Moreover, given a tabulation point, all the polynomial terms corresponding to the different escape coefficients are allocated adjacently in memory, in order to minimize paging.

Tabulation of the escape coefficients

The procedure adopted to compute the tables depends on the value of the optical length, more precisely:

For large τ : The forward recursive relation is used for large values of the optical length to compute all the higher order coefficients starting from the 0-th order value.

$$E_0^T(\tau) = 1 - e^{-\tau},$$

$$\vdots$$

$$E_b^T(\tau) = 1 - \frac{b}{\tau} E_{b-1}^T(\tau),$$

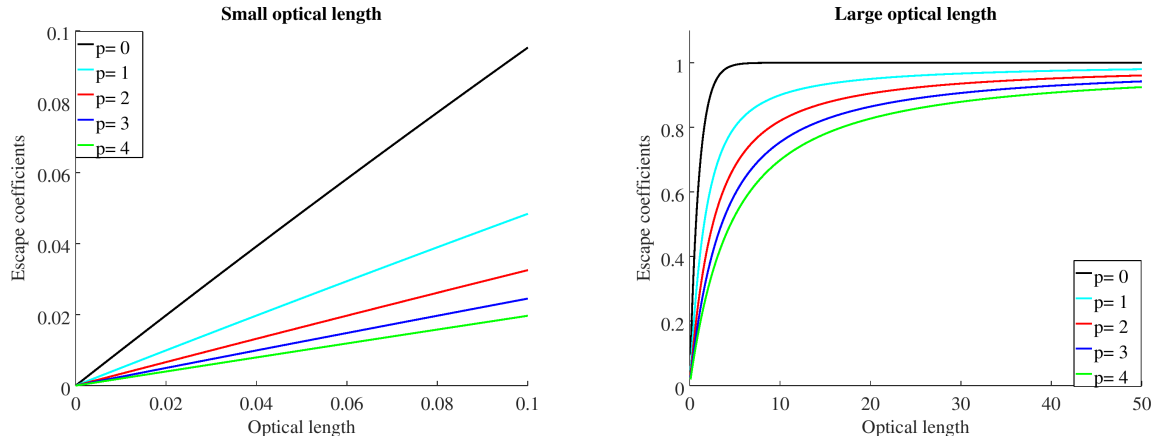


Figure 16 – Behaviour of the *tabulated escape coefficients* $E_b^T(\tau)$ as a function of the *optical length* for different values of the polynomial orders p.

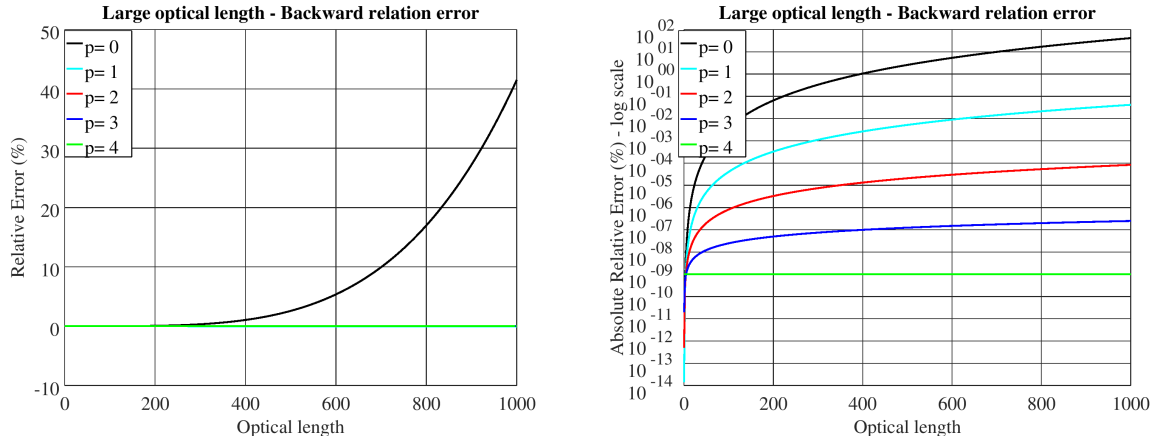


Figure 17 – Error associated to the use of the backward recurrent relation to compute the *tabulated escape coefficients* for large values of the *optical length*.

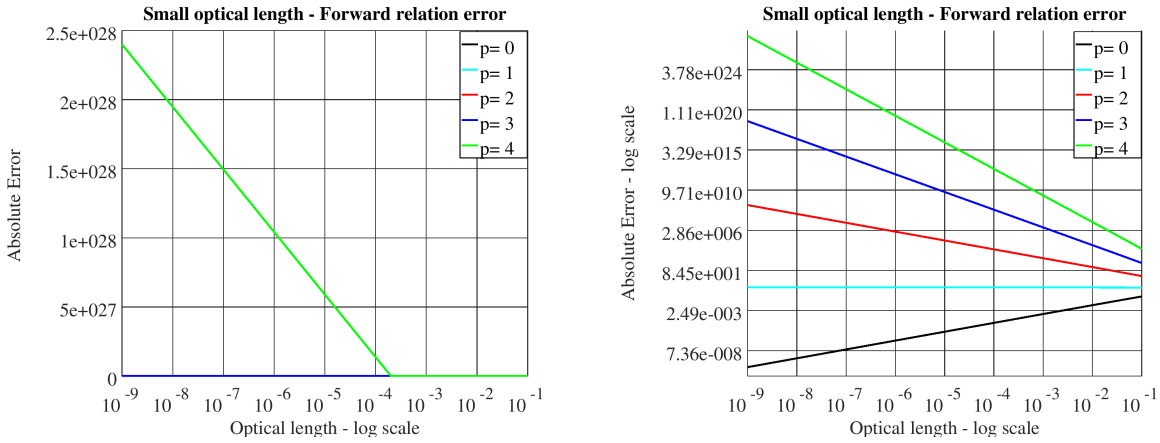


Figure 18 – Error associated to the use of the forward recurrent relation to compute the *tabulated escape coefficients* for small values of the *optical length*.

For small τ : For small τ values, a McLaurin series is used to expand the function $e^{-\tau}$. Every coefficient can in fact be expressed as a combination of exponential functions:

$$
\begin{aligned}
E_0^T(\tau) &= 1 - e^{-\tau} = \tau - \frac{\tau^2}{2} + \frac{\tau^3}{3!} - ..., \\
E_1^T(\tau) &= 1 - \frac{1}{\tau} E_0^T(\tau) = \frac{\tau}{2} - \frac{\tau^2}{3!} + \frac{\tau^3}{4!} - ..., \\
&\vdots \\
E_3^T(\tau) &= 1 - \frac{3}{\tau} E_2^T(\tau) = \frac{3!}{4!}\tau - \frac{3!}{5!}\tau^2 + \frac{3!}{6!}\tau^3 - ..., \\
E_b^T(\tau) &= \sum_{i=1}^{\infty} (-)^{i+1} \frac{b!}{(b+i)!} \tau^i, \\
&\simeq \frac{\tau}{b+1}\left[1 - \frac{\tau}{b+2}\left[1 - \frac{\tau}{b+3}\left[1 - \frac{\tau}{b+4}\right]\right]\right].
\end{aligned}
$$

5.4.2 Numerical-transmission equation

The computational cost associated to the polynomial transmission equation, Eq.(5.34), is very important, in particular when compared to the SC equivalent, Eq.(4.4), and considering that the transmission equation must be solved for each 3D chord.

The polynomial transmission equation is rearranged in a form that results much more numerically efficient, by inverting the sums order:

$$
\psi^+(\vec{\Omega}) = \psi^-(\vec{\Omega})\, e^{-\Sigma_r l} + \sum_{k=0}^{N_p} P_k(z_r^{in}) \sum_{p=k}^{N_p} c_{p,k}\, \mu^{p-k} \left(\frac{2}{\Delta z}\right)^{p-k} E_{p-k}(\tau)\, \frac{q_{r,p}(\vec{\Omega})}{\Sigma_r}.
$$

Thanks to this simple change we can write:

$$
\psi^+(\vec{\Omega}) = \psi^-(\vec{\Omega})\, e^{-\Sigma_r l} + \vec{P}(z_r^{in}) \cdot \vec{T}, \tag{5.41}
$$

where:

$$
\begin{aligned}
\vec{T} &= [T_k], \qquad T_k = \sum_{p=k}^{N_p} \bar{\bar{F}}_{p,k}\, \frac{q_{r,p}(\vec{\Omega})}{\Sigma_r}, \\
\bar{\bar{F}}_{p,k} &= c_{p,k}\, \mu^{p-k} \left(\frac{2}{\Delta z}\right)^{p-k} E_{p-k}(\tau).
\end{aligned} \tag{5.42}
$$

Besides being more aesthetically digestible, the new formulation of Eq.(5.41) allows to deeply exploit the *Chord Classification Method* (CCM) introduced in [23], and quickly described in subsection 3.1.2. This method, originally implemented to pre-compute the $\beta = 1 - e^{-\tau}$ coefficient for the SC method, is even more efficient when applied to the polynomial method.

Looking at Eq.(5.41) it is easy to understand that the product $\vec{P}(z_r^{in}) \cdot \vec{T}$ must be performed for each 3D chord during the transport sweep, since it depends on the trajectory entering point. The term $\vec{T}$, on the other hand, can be pre-computed. Thanks to the CCM, the important computational complexity that characterizes it can be drastically reduced. The method allows in fact to identify *classified* chords, whose length is the same, and to regroup them all in the same *class*. Depending of the *class* type, $\vec{T}$ can be expressed as:

$$
\begin{aligned}
&\text{For } \textit{V-chords} && T_k = T_k^v\,(i_{2D}, r_z, \theta), \\
&\text{For } \textit{H-chords} && T_k = T_k^h\,(r_{2D}, r_z, \theta),
\end{aligned}
$$

where $i_{2D}, r_{2D}, r_z, \theta$ represent a 2D chord, a 2D region, an axial plane and the polar coordinate index, respectively. These indexes make explicit the dependencies of the classified coefficients and also give an idea of the size of the information to be stored in order to apply the CCM method. Remark that the amount of storage needed is fixed for a given two-dimensional tracking, axial discretization and quadrature formula. The amount of 3D chords that the classes are able to represent, on the other hand, will also depend on the geometrical properties of the domain, such as the size of the radial and axial meshes.

Following this reasoning, $\vec{T}$ is pre-computed for all the V and H classes, and computed on-the-fly only for *M-chords*. The advantages of the CCM when applied to the polynomial algorithm, as compared to the SC method are two-fold: first, the computational cost of $\vec{T}$ is much higher in comparison to the β coefficients used in the SC method. The CCM method allows therefore to avoid a much higher computational cost of the polynomial method. Second, thanks to the high order flux representation, the polynomial method reduces the number of axial meshes that must be used for an equally precise solution, when compared to the SC method. This directly impacts the number of chords belonging to the same class, increasing it. This means that with a lower number of axial meshes, it is sufficient to compute the $\vec{T}$ terms for a lower number of classes, and this lower number of classes is able to represent a large percentage of the total number of 3D chords. In other words, the number of *classes*, for which $\vec{T}$ is pre-computed, and of *M-chords*, for which no classification is applied, is lower, when larger axial meshes are employed.

5.4.3 Transmission cost considerations

As described in the previous section, the CCM introduced in [23] greatly reduces the computational burden of the polynomial transmission algorithm that we have implemented. However, we can see that the SC transmission algorithm remains more computationally lighter, in comparison with its polynomial counterpart. Even if this sounds natural for a high order method, we would like to present in this section a cross comparison between the transmission algorithms of the two methods, in order to underline where the computational over-cost of the polynomial version is concentrated.

For this purpose we reformulate the ΔJ terms for the Step and Polynomial method of Eq.(3.2) and Eq.(5.23) as:

$$\Delta J_r(\vec{\Omega}) = \frac{\Delta_\perp(\vec{\Omega})}{V_r} \sum_{\substack{t \| \vec{\Omega} \\ t \cap r}} \delta_i(\vec{\Omega}) \qquad \delta_i(\vec{\Omega}) = \psi^+(t,\vec{\Omega}) - \psi^-(t,\vec{\Omega}),$$

$$\Delta \vec{J}_r(\vec{\Omega}) = \frac{\Delta_\perp(\vec{\Omega})}{V_r(\vec{\Omega})} \sum_{\substack{t \| \vec{\Omega} \\ t \cap r}} \vec{\delta}_i(\vec{\Omega}) \qquad \vec{\delta}_i(\vec{\Omega}) = \vec{P}(\tilde{z}_r^+)\, \psi^+(t,\vec{\Omega}) - \vec{P}(\tilde{z}_r^-)\, \psi^-(t,\vec{\Omega}).$$

The operations to be performed for each chord (i) during the transmission sweep, assuming that β (for the SC method) or $\vec{T}$ (for the polynomial method) are already available when

sweeping the trajectory, can be summarized as:

STEP	*POLYNOMIAL*
1) $\delta_i = \beta_i \left(\dfrac{q_r(\vec{\Omega})}{\Sigma_r} - \psi_r^-(t,\vec{\Omega}) \right)$	1) $\psi^+(t,\vec{\Omega}) = \psi^-(t,\vec{\Omega})\, e^{-\Sigma_r l} + \vec{P}(z_r^{in}) \cdot \vec{T}$
2) $\delta_{r(i)}(\vec{\Omega}) = \delta_{r(i)}(\vec{\Omega}) + \delta_i$	2) $\vec{\delta}_i = \vec{P}(\tilde{z}_r^+)\, \psi^+(t,\vec{\Omega}) - \vec{P}(\tilde{z}_r^-)\, \psi^-(t,\vec{\Omega})$
3) $\psi^+(t,\vec{\Omega}) = \psi^-(t,\vec{\Omega}) + \delta_i$	3) $\vec{\delta}_{r(i)}(\vec{\Omega}) = \vec{\delta}_{r(i)}(\vec{\Omega}) + \vec{\delta}_i$
$= 1$ flop	$= 1 + 2\,N_p$ flops

where the point 2) for the step and 3) for the polynomial methods coincide with the cumulation over $\sum_{\substack{t \parallel \vec{\Omega} \\ t \cap r}}$. Also, remark that, for simplicity, the computational cost of 2) for the polynomial method is assumed to be only equal to N_p floating-point operations, since we are assuming that $\vec{P}(\tilde{z}_r^-)\, \psi^-(t,\vec{\Omega})$ is already computed because it is equal to the $\vec{P}(\tilde{z}_r^+)\, \psi^+(t,\vec{\Omega})$ of the precedent chord (this assumption is not valid when crossing a horizontal surface, but in this case a simple sign change for the odd terms is enough to pass the polynomial flux product to the next region).

Even under some simplifying assumptions, the polynomial method still detains a much higher computational cost per chord. For example, for $N_p = 2$, which is the most common expansion we have used in our calculations, we have 5 floating-point operations against 1. From these considerations it results clear a non-negligible over cost of the polynomial method in comparison with the standard SC, at least for the transmission sweep phase just described and assuming a comparable number of 3D chords to be treated.

6. DP_N synthetic acceleration

It is well known that, to be able to perform realistic reactor physics transport calculations, an acceleration method to reduce the number of free iterations is mandatory [47]. In the framework of 3D MOC-based calculations the most popular approach is that based on the Coarse-Mesh Finite Difference (CMFD) technique.

In TDT, starting from its implementation in the APOLLO2 code, a synthetic approach to accelerate not only mono-group but also the external multi-group fission iterations was adopted. Even if this technique is more complicated to implement in comparison with CMFD, it has strong memory needs and it is difficult to solve, it has proved to be totally stable and very efficient. Moreover it can be applied to the acceleration of high order spatial and angular moments. Reference [45] describes the details regarding the implementation of the synthetic inner and outer acceleration, while the original idea for the outer treatment was introduced in [48]. The numerical details about the DP_N method for the 3D MOC Step method are presented in [26]. Note also that the polynomial formulation shares a lot with the SC acceleration method. In order to avoid excessive repetitions, in this chapter we mainly put emphasis on the aspects that are different in the Polynomial method, in comparison with the SC approximation described in [26].

We start this section with some basic principles of the acceleration strategy for inner mono-group, and outer multi-group iterations. Then, we re-derive our synthetic DP_N operator from first principles to be coherent with a 3D MOC Polynomial method.

6.1 Inner acceleration

In subsection 3.1.1 we briefly discussed the iterative strategy. Employing an approach similar to the one used for the transport solution, the acceleration strategy also is decomposed in several layers. The innermost level coincides with the *inner iterations*. As presented in algorithm 1, a mono-group flux solution is obtained and the self-scattering term is updated until convergence is reached. We can schematically write this procedure as:

$$\mathcal{L}\ \psi_{i+\frac{1}{2}} = \mathcal{H}^{g\to g}\ \psi_i + q^{ext}, \tag{6.1}$$

where i indicates the *inner iterations* index and the different terms are:

$$\begin{aligned} \mathcal{L} &= \vec{\Omega}\cdot\vec{\nabla} + \Sigma \\ \mathcal{H}^{g\to g} &= \text{self scattering operator} \\ q^{ext} &= \text{transfer from other groups+fission.} \end{aligned}$$

If we indicate the converged solution as ψ^{∞}, we can write:

$$\mathcal{L}\ \psi_{\infty} = \mathcal{H}^{g\to g}\ \psi_{\infty} + q^{ext}, \tag{6.2}$$

and we define $\Delta\psi$, as the correction that separates the flux at the current iteration from the converged solution:

$$\psi_\infty = \psi_{i+\frac{1}{2}} + \Delta\psi_{i+\frac{1}{2}}. \tag{6.3}$$

Subtracting Eq.(6.1) from Eq.(6.2) and using (6.3) we obtain:

$$\mathcal{L}\,\Delta\psi_{i+\frac{1}{2}} = \mathcal{H}^{g\to g}\Delta\psi_{i+\frac{1}{2}} + \mathcal{H}^{g\to g}\left(\psi_{i+\frac{1}{2}} - \psi_i\right). \tag{6.4}$$

Solving this problem is, of course, as difficult as solving the original one. Therefore the acceleration problem is generally solved with a lower order approximation which must be computationally cheaper in comparison with the transport solution, even if not as much precise. In our case the DP_N synthetic acceleration is used to search the solution of Eq.(6.4), that can be written as:

$$\underbrace{\left(\mathcal{L} - \mathcal{H}^{g\to g}\right)}_{DP_N}\Delta\psi_{i+\frac{1}{2}} = \mathcal{H}^{g\to g}\left(\psi_{i+\frac{1}{2}} - \psi^i\right).$$

The acceleration operator is used in this case for the solution of this mono-group equation, where the source term is replaced with a source depending on the difference between two successive inner iterations fluxes. The solution of this problem, $\Delta\psi_{i+\frac{1}{2}}$, is added to the previous transport solution, in order to obtain a better estimation of the solution, which should be closer to the converged value:

$$\psi_{i+1} = \psi_{i+\frac{1}{2}} + \Delta\psi_{i+\frac{1}{2}}.$$

6.2 Outer acceleration

In order to describe the outer acceleration procedure, we need first to introduce the notation used to define the transport outer iterations. Using again i as *inner iterations* index, and o for the *outer iteration* index, we can write:

$$\left(\mathcal{L} - \mathcal{H}^d\right)\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}} = \mathcal{H}^u\ \psi_i^{o+\frac{1}{2}} + \frac{1}{\lambda^o}\mathcal{F}\ \psi_{i_{tot}}^o, \tag{6.5}$$

where:

$$\begin{aligned}
\mathcal{L} &= \vec{\Omega}\cdot\vec{\nabla} + \Sigma \\
\mathcal{H}^d &= \text{self + down scattering operator} \\
\mathcal{H}^u &= \text{up scattering operator} \\
\mathcal{F} &= \text{fission operator} \\
i_{tot} &= \text{the number of inner iterations to attain convergence.}
\end{aligned}$$

Following a similar reasoning as in the previous section, we can write the converged equation as:

$$\left(\mathcal{L} - \mathcal{H}^d\right)\psi_{i_{tot}}^\infty = \mathcal{H}^u\ \psi_{i_{tot}}^\infty + \frac{1}{\lambda^\infty}\mathcal{F}\ \psi_{i_{tot}}^\infty, \tag{6.6}$$

and a $\Delta\psi$, which separates the solution at the current iteration, from the converged one, as:

$$\psi_{i_{tot}}^\infty = \psi_{i+\frac{1}{2}}^{o+\frac{1}{2}} + \Delta\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}. \tag{6.7}$$

Replacing Eq.(6.7) into Eq.(6.6), we get:

$$\left(\mathcal{L}-\mathcal{H}^d\right)\Delta\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}=\mathcal{H}^u\Delta\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}+\frac{1}{\lambda^\infty}\mathcal{F}\Delta\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}-\left(\mathcal{L}-\mathcal{H}^d\right)\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}+\mathcal{H}^u\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}+\frac{1}{\lambda^\infty}\mathcal{F}\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}.$$

We replace at this point the $\left(\mathcal{L}-\mathcal{H}^d\right)\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}$ term using Eq.(6.5), and we obtain:

$$\left(\mathcal{L}-\mathcal{H}^d\right)\Delta\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}=\mathcal{H}^u\Delta\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}+\frac{1}{\lambda^\infty}\mathcal{F}\Delta\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}+\mathcal{H}^u\left(\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}-\psi_i^{o+\frac{1}{2}}\right)+\mathcal{F}\left(\frac{\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}}{\lambda^\infty}-\frac{\psi_{i_{tot}}^{o}}{\lambda^o}\right).$$

We recast this as:

$$\left(\mathcal{L}-\mathcal{H}^d\right)\Delta\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}=\mathcal{H}^u\Delta\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}+\frac{1}{\lambda^\infty}\mathcal{F}\,\Delta\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}+q_{DP_N}^{ext}(\lambda^\infty),\tag{6.8}$$

where:

$$q_{DP_N}^{ext}(\lambda^\infty)=\mathcal{H}^u\left(\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}-\psi_i^{o+\frac{1}{2}}\right)+\mathcal{F}\left(\frac{\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}}{\lambda^\infty}-\frac{\psi_{i_{tot}}^{o}}{\lambda^o}\right).$$

The solution of Eq.(6.8) also requires an iteration strategy. To present this iterative procedure, we drop all the iteration indexes that are not required to compute $\Delta\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}$ and λ^∞. Both inner and outer iterations are necessary at this point, but we write only the synthetic outer iterations index b, assuming that the inner strategy is the same as for the original problem. Equation (6.8) can be re-written as:

$$\left(\mathcal{L}-\mathcal{H}^d\right)\Delta\psi^{b+1}=\mathcal{H}^u\Delta\psi^{b+1}+\frac{1}{\lambda^b}\mathcal{F}\,\Delta\psi^{b}+q_{DP_N}^{ext}(\lambda^b).$$

The outer iterations are also quite similar to the original problem (except for the presence of the $q_{DP_N}^{ext}$ term). The most peculiar aspect of the Suslov algorithm presented in [48] is the power iteration used for the eigenvalue updating, which reads:

$$\lambda^{b+1}=\lambda^b\frac{\left|\left|\mathcal{F}\left[\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}+\Delta\psi^{b+1}\right]\right|\right|}{\left|\left|\mathcal{F}\left[\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}+\Delta\psi^{b}\right]\right|\right|}=\lambda^b\frac{\left|\left|\mathcal{F}\left[\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}+\Delta\psi_{i+\frac{1}{2}}^{o+\frac{1}{2},b+1}\right]\right|\right|}{\left|\left|\mathcal{F}\left[\psi_{i+\frac{1}{2}}^{o+\frac{1}{2}}+\Delta\psi_{i+\frac{1}{2}}^{o+\frac{1}{2},b}\right]\right|\right|}.$$

where we used at first the simplified iteration notation, and then the complete one. $\mathcal{F}\psi$ is the fission integral defined in Eq.(2.13).

6.3 Polynomial DP_N formulation

To obtain the synthetic equations used to solve the transport operator $\mathcal{L}-\mathcal{H}^{g\to g}$, two hypotheses are used. First, the source term is expanded as in Eq.(5.10), but with a lower order "V" approximation:

$$q(\vec{r},\vec{\Omega})=\vec{\boldsymbol{\mathcal{Z}}}_V(\tilde{z},\vec{\Omega})\cdot\vec{\boldsymbol{q}}_r,$$

where $\vec{\boldsymbol{\mathcal{Z}}}_V(\tilde{z},\vec{\Omega})$ is supposed to have a lower number of angular modes than those used in the transport calculation but the same spatial polynomial basis.

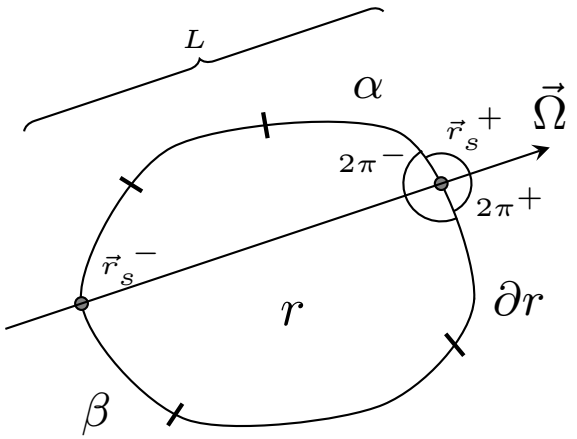


Figure 19 – Given a computational region r and a trajectory of direction $\vec{\Omega}$ that draws a chord of length L on r, two boundary surfaces are identified over the boundary ∂r which we denote by α and β. The entering and exiting impact points are $\vec{r}_s^{\,-}$ and $\vec{r}_s^{\,+}$, respectively. At each impact point a suitable set of entering/exiting directions $(2\pi)^{\pm}$ w.r.t. the region r are defined.

Second, the boundary of each region is decomposed into surfaces, labeled with the index α. The boundary angular flux is also expanded with an approximation similar to Eq.(5.10):

$$\Psi(\vec{r}_s,\vec{\Omega}) = \vec{\boldsymbol{\mathcal{Z}}}_S^{\pm}(\tilde{z}_s,\vec{\Omega})\cdot\vec{\boldsymbol{\Phi}}_r^{\alpha^{\pm}} \qquad \vec{r}_s\in\alpha,\ \vec{\Omega}\in S_{2\pi}^{\pm}. \tag{6.9}$$

Here $\vec{\boldsymbol{\mathcal{Z}}}_V$ and $\vec{\boldsymbol{\mathcal{Z}}}_S^{\pm}$ represent the same set of functions but with two different orders, representing a *Volume* and a *Surface* approximation. Remember also the existence of $\vec{\boldsymbol{\mathcal{Z}}}_D$ related to Eq.(5.31). Here and in the following "V","S" and "D" will be used to subscript quantities related to these moment definitions. The dimensions of these quantities are due to the scattering order associated to them and to the chosen degree for the polynomial expansion. They read:

$$\begin{aligned} N_{m,S} &= (K_S+1)^2,\\ N_{m,V} &= (K_V+1)^2,\\ N_{m,D} &= (K_V+N_p+1)^2. \end{aligned}$$

For $N_{m,D}$ the same rule as in Eq.(5.30) applies here. In Eq.(6.9) the symbol $\pm$ refers to the exiting/entering values through a surface, while $\vec{r}_s$ and $\tilde{z}_s$ are the position vector and the axial coordinate on the region boundary. Moreover in Eq.(6.9) the symbols $\pm$ are also used to distinguish which polynomial base is to be used, depending on the surface side (see Fig.19 for a sketch of the situation).

Another important remark has to be addressed regarding the type of surfaces. From now on we will refer to vertical and horizontal surfaces with α_v and α_h respectively, while the common α subscript will be used if no distinction between the two cases is necessary. For horizontal surfaces the vectors $\vec{\boldsymbol{\mathcal{Z}}}_S^{\pm}(\tilde{z}_s,\vec{\Omega})$ of Eq.(6.9) is constituted by repeated sets of identical spatial functions. For this it is sufficient to remark that $P_p(+1)=1$ and $P_p(-1)=(-1)^p$. It should be therefore unuseful and harmful to expand the surface flux over such not linearly-independent set.

Therefore, for the horizontal surfaces, we limit the boundary angular flux expansion to the spatially constant term and the expansion of Eq.(6.9) is now considered divided into the

following two sub-cases:

$$\begin{aligned}\Psi(\vec{r}_s,\vec{\Omega}) =& \vec{\boldsymbol{\mathcal{Z}}}_S^{\pm}(\tilde{z}_s,\vec{\Omega})\cdot\vec{\boldsymbol{\Phi}}_r^{\alpha_v^{\pm}} \qquad\qquad \vec{r}_s\in\alpha_v,\ \vec{\Omega}\in S_{2\pi}^{\pm},\\ \Psi(\vec{r}_s,\vec{\Omega}) =& \vec{A}_S^{\pm}(\vec{\Omega})\cdot\vec{\Phi}_r^{\alpha_h^{\pm}} \qquad\qquad \vec{r}_s\in\alpha_h,\ \vec{\Omega}\in S_{2\pi}^{\pm},\end{aligned} \tag{6.10}$$

where $\vec{A}_S^{\pm}(\vec{\Omega})$ is the subset of $\vec{\boldsymbol{\mathcal{Z}}}_S^{\pm}(\tilde{z}_s,\vec{\Omega})$ corresponding to the zero order polynomial value.

DP_N Balance Equation

A balance equation for the synthetic method can be obtained applying the projection $\oint \frac{d\vec{\Omega}}{4\pi}\int_r d\vec{r}\ \vec{\boldsymbol{\mathcal{Z}}}_D(\vec{r},\vec{\Omega})$ to Eq.(2.11), which results in an expression similar to Eq.(5.31):

$$(\Sigma_r-\boldsymbol{\mathcal{D}})'\vec{\boldsymbol{\Phi}}_{r,D}=\boldsymbol{\mathcal{Z}}_{DV}\ \vec{\boldsymbol{q}}_r-\oint\frac{d\vec{\Omega}}{4\pi}\ \vec{A}_D(\vec{\Omega})\otimes\Delta\vec{J}_r(\vec{\Omega}). \tag{6.11}$$

The current term is then developed, decomposing the surface integral over ∂r as the contribution of the different surfaces α for the given region:

$$\begin{aligned}\oint\frac{d\vec{\Omega}}{4\pi}\ \vec{A}_D(\vec{\Omega})\otimes\Delta\vec{J}_r(\vec{\Omega}) =& \oint\frac{d\vec{\Omega}}{4\pi}\ \vec{A}_D(\vec{\Omega})\otimes\frac{1}{V_r}\int_{\partial r}d\vec{r}_s\ \vec{\Omega}\cdot\hat{n}\ \vec{P}(\tilde{z}_s)\Psi(\vec{r}_s,\vec{\Omega})\\ =&\frac{1}{V_r}\sum_{\alpha\in r}\int_\alpha d\vec{r}_s\left[\int_{2\pi^+}\frac{d\vec{\Omega}}{4\pi}\ |\vec{\Omega}\cdot\hat{n}|\ \vec{A}_D(\vec{\Omega})\otimes\vec{P}(\tilde{z}_s)\,\Psi(\vec{r}_s,\vec{\Omega})\right.\\ &\left.-\int_{2\pi^-}\frac{d\vec{\Omega}}{4\pi}\ |\vec{\Omega}\cdot\hat{n}|\ \vec{A}_D(\vec{\Omega})\otimes\vec{P}(\tilde{z}_s)\,\Psi(\vec{r}_s,\vec{\Omega})\right].\end{aligned}$$

Eq.(6.11) becomes:

$$(\Sigma_r-\boldsymbol{\mathcal{D}})'\vec{\boldsymbol{\Phi}}_{r,D}=\boldsymbol{\mathcal{Z}}_{DV}\ \vec{\boldsymbol{q}}_r-\frac{1}{V_r}\sum_{\alpha\in r}\left(\vec{\boldsymbol{J}}_\alpha^{+}-\vec{\boldsymbol{J}}_\alpha^{-}\right), \tag{6.12}$$

where the partial currents $\vec{\boldsymbol{J}}_\alpha^{\pm}$ are defined as:

$$\vec{\boldsymbol{J}}_\alpha^{\pm}=\int_\alpha d\vec{r}_s\int_{2\pi^\pm}\frac{d\vec{\Omega}}{4\pi}\ |\vec{\Omega}\cdot\hat{n}|\ \vec{\boldsymbol{Z}}_D^{\pm}(\tilde{z}_s,\vec{\Omega})\,\Psi(\vec{r}_s,\vec{\Omega}), \tag{6.13}$$

with $\vec{\boldsymbol{Z}}_D^{\pm}(\tilde{z}_s,\vec{\Omega})=\vec{A}_D(\vec{\Omega})\otimes\vec{P}^{\pm}(\tilde{z}_s)$, which can be written in a more explicit form for horizontal surfaces:

$$\vec{\boldsymbol{J}}_{\alpha_h}^{\pm}=\vec{P}_b(\alpha_h^{\pm})\otimes\vec{J}_{\alpha_h}^{\pm} \qquad and \qquad \vec{J}_{\alpha_h}^{\pm}=\int_\alpha d\vec{r}_s\int_{2\pi^\pm}\frac{d\vec{\Omega}}{4\pi}\ |\vec{\Omega}\cdot\hat{n}|\ \vec{A}_D(\vec{\Omega})\,\Psi(\vec{r}_s,\vec{\Omega}), \tag{6.14}$$

where $\vec{J}_{\alpha_h}^{\pm}$ is the constant current term and $\vec{P}_b(\alpha_h)$ is the parity vector whose components are equal to 1 for upper horizontal side and to $(-1)^p$ for the lower side.

The partial currents Eqs.(6.13) and (6.14) can also be written in an alternative form using Eqs.(6.10) for $\Psi(\vec{r}_s,\vec{\Omega})$:

$$\vec{\boldsymbol{J}}_{\alpha_v}^{\pm}=\boldsymbol{\mathcal{Z}}_{\alpha_v^{\pm}}\ \vec{\boldsymbol{\Phi}}_r^{\alpha_v^{\pm}} \qquad and \qquad \vec{J}_{\alpha_h}^{\pm}=\bar{\bar{\mathcal{A}}}_{\alpha_h^{\pm}}\ \vec{\Phi}_r^{\alpha_h^{\pm}}, \tag{6.15}$$

where:

$$\mathfrak{Z}_{\alpha_v^\pm} = \int_\alpha d\vec{r}_s \int_{2\pi^\pm} \frac{d\vec{\Omega}}{4\pi} \;\; |\vec{\Omega}\cdot\hat{n}| \;\; \vec{\boldsymbol{\mathcal{Z}}}_D(\tilde{z}_s,\vec{\Omega}) \otimes \vec{\boldsymbol{\mathcal{Z}}}_S(\tilde{z}_s,\vec{\Omega}), \tag{6.16}$$

$$\bar{\bar{\mathcal{A}}}_{\alpha_h^\pm} = \int_\alpha d\vec{r}_s \int_{2\pi^\pm} \frac{d\vec{\Omega}}{4\pi} \;\; |\vec{\Omega}\cdot\hat{n}| \;\; \vec{A}_D(\vec{\Omega}) \otimes \vec{A}_S(\vec{\Omega}).$$

The matrices $\mathfrak{Z}_{\alpha_v^\pm}$ and $\bar{\bar{\mathcal{A}}}_{\alpha_h^\pm}$ have dimensions $(N_{m,D}\times N_p)\times(N_{m,S}\times N_p)$ and $N_{m,D}\times N_{m,S}$, respectively. Equation (6.14) for $\vec{\boldsymbol{J}}^\pm_{\alpha_h}$ shows that for horizontal surfaces the polynomial components needed to close the DP_N problem (as shown in Eq.(6.11)) can be obtained applying the possible sign change to the constant component $\vec{J}^\pm_{\alpha_h}$. Having this in mind, we write (6.11) explicitly as:

$$(\Sigma_r - \boldsymbol{\mathcal{D}})\,'\vec{\boldsymbol{\Phi}}_{r,D} = \mathfrak{Z}_{DV}\;\vec{\boldsymbol{q}}_r - \frac{1}{V_r}\left[\sum_{\alpha_v\in r}\left(\vec{\boldsymbol{J}}^+_\alpha - \vec{\boldsymbol{J}}^-_\alpha\right) + \sum_{\alpha_h\in r}\left(\vec{P}_b(\alpha_h^+)\otimes\vec{J}^+_{\alpha_h} - \vec{P}_b(\alpha_h^-)\otimes\vec{J}^-_{\alpha_h}\right)\right]. \tag{6.17}$$

Current Transmission

The transmission equation (5.34) is used to obtain a relation for the exiting current from the surface α as a function of the entering surface fluxes and of the source coefficients. First, we express the entering angular flux $\psi^-(\vec{\Omega})$ using Eqs.(6.10). Then, we use Eq.(5.12) for $q_{r,p}(\vec{\Omega})$ and derive separate expressions for horizontal and vertical surfaces:

$$\begin{aligned} for\ \tilde{z}_s^-\in\beta_v \qquad \Psi(\vec{r}_s{}^+,\vec{\Omega}) =& \vec{\boldsymbol{\mathcal{Z}}}_S(\tilde{z}_s^-,\vec{\Omega})\;\vec{\boldsymbol{\Phi}}_r^{\beta_v^-}\;e^{-\tau} \\ &+\sum_n^{N_m} A_n(\vec{\Omega})\sum_{p=0}^{N_p}\left[\sum_{k=0}^{p} c_{p,k}\;P_k(\tilde{z}_s^-)\;\mu^{p-k}\left(\frac{2}{\Delta z}\right)^{p-k} E_{p-k}(\tau)\right]\frac{q^n_{r,p}}{\Sigma_r}, \end{aligned} \tag{6.18}$$

$$\begin{aligned} for\ \tilde{z}_s^-\in\beta_h \qquad \Psi(\vec{r}_s{}^+,\vec{\Omega}) =& \vec{A}_S(\vec{\Omega})\cdot\vec{\Phi}_r^{\beta_h^-}\;e^{-\tau} \\ &+\sum_n^{N_m} A_n(\vec{\Omega})\sum_{p=0}^{N_p}\left[\sum_{k=0}^{p} c_{p,k}\;P_k(\tilde{z}_s^-)\;\mu^{p-k}\left(\frac{2}{\Delta z}\right)^{p-k} E_{p-k}(\tau)\right]\frac{q^n_{r,p}}{\Sigma_r}, \end{aligned} \tag{6.19}$$

where $\tau=\Sigma_r\,l$ and $l=|\vec{r}_s^{\,+}-\vec{r}_s^{\,-}|$ and we assume that the exiting point $\vec{r}_s^{\,+}$ belongs to a surface α^+, while the trajectory entering point $\vec{r}_s^{\,-}$ belongs to a surface β^-, as depicted in Fig.19.

Next, we convert the integral over the positive directions through the surface α^+ using a discrete sum over the entering surfaces β^-:

$$\int_{\alpha^+_{v/h}} d\vec{r}_s^{\,+}\int_{2\pi^+} d\vec{\Omega} = \int_{\alpha^+_{v/h}} d\vec{r}_s^{\,+}\sum_{\beta_v\in r}\int_{(\beta_v^-\to\alpha^+_{v/h})} d\vec{\Omega} + \int_{\alpha^+_{v/h}} d\vec{r}_s^{\,+}\sum_{\beta_h\in r}\int_{(\beta_h^-\to\alpha^+_{v/h})} d\vec{\Omega},$$

$$\int_{\alpha^+} d\vec{r}_s^{\,+}\int_{2\pi^+} d\vec{\Omega} = \int_{\alpha^+} d\vec{r}_s^{\,+}\sum_{\beta\in r}\int_{(\beta^-\to\alpha^+)} d\vec{\Omega}$$

where v/h means that the previous relation is valid for both horizontal and vertical surfaces, and $\beta_v^-\to\alpha^+_{v/h}$ and $\beta_h^-\to\alpha^+_{v/h}$ indicate that the angular integral is performed over the set of directions that connects the vertical and horizontal surfaces β^- to α^+, respectively.

Keeping this in mind and substituting Eqs.(6.18) and (6.19) into Eqs.(6.13) and (6.14) we obtain two relations, depending on the surface type. We write this using a compact notation:

$$\begin{bmatrix} \vec{\boldsymbol{J}}^{+}_{\alpha_v} \\ \vec{J}^{+}_{\alpha_h} \end{bmatrix} = \sum_{\beta \in r} \begin{bmatrix} \boldsymbol{\mathfrak{T}}_{\alpha_v^+ \beta_v^-} & \boldsymbol{\mathfrak{T}}_{\alpha_v^+ \beta_h^-} \\ \boldsymbol{\mathfrak{T}}_{\alpha_h^+ \beta_v^-} & \boldsymbol{\mathfrak{T}}_{\alpha_h^+ \beta_h^-} \end{bmatrix} \begin{bmatrix} \vec{\boldsymbol{\Phi}}^{-}_{\beta_v} \\ \vec{\Phi}^{-}_{\beta_h} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\mathfrak{E}}_{\alpha_v^+} \\ \boldsymbol{\mathfrak{E}}_{\alpha_h^+} \end{bmatrix} \vec{\boldsymbol{q}}_r, \tag{6.20}$$

where the *transmission coefficients* $\boldsymbol{\mathfrak{T}}_{\alpha\beta}$ are defined as:

$$\begin{aligned}
\boldsymbol{\mathfrak{T}}_{\alpha_v^+ \beta_v^-} &= \int_{\alpha_v^+} d\vec{r}_s^{\,+} \int_{(\beta_v^- \to \alpha_v^+)} \frac{d\vec{\Omega}}{4\pi} \; |\vec{\Omega} \cdot \hat{n}| \; \vec{\boldsymbol{\mathcal{Z}}}_S(\tilde{z}_s^{\,+}, \vec{\Omega}) \otimes \vec{\boldsymbol{\mathcal{Z}}}_S(\tilde{z}_s^{\,-}, \vec{\Omega}) \; e^{-\tau}, \\
\boldsymbol{\mathfrak{T}}_{\alpha_v^+ \beta_h^-} &= \int_{\alpha_v^+} d\vec{r}_s^{\,+} \int_{(\beta_h^- \to \alpha_v^+)} \frac{d\vec{\Omega}}{4\pi} \; |\vec{\Omega} \cdot \hat{n}| \; \vec{\boldsymbol{\mathcal{Z}}}_S(\tilde{z}_s^{\,+}, \vec{\Omega}) \otimes \vec{A}_S(\vec{\Omega}) \; e^{-\tau}, \\
\boldsymbol{\mathfrak{T}}_{\alpha_h^+ \beta_v^-} &= \int_{\alpha_h^+} d\vec{r}_s^{\,+} \int_{(\beta_v^- \to \alpha_h^+)} \frac{d\vec{\Omega}}{4\pi} \; |\vec{\Omega} \cdot \hat{n}| \; \vec{A}_S(\vec{\Omega}) \otimes \vec{\boldsymbol{\mathcal{Z}}}_S(\tilde{z}_s^{\,-}, \vec{\Omega}) \; e^{-\tau}, \\
\boldsymbol{\mathfrak{T}}_{\alpha_h^+ \beta_h^-} &= \int_{\alpha_h^+} d\vec{r}_s^{\,+} \int_{(\beta_h^- \to \alpha_h^+)} \frac{d\vec{\Omega}}{4\pi} \; |\vec{\Omega} \cdot \hat{n}| \; \vec{A}_S(\vec{\Omega}) \otimes \vec{A}_S(\vec{\Omega}) \; e^{-\tau},
\end{aligned} \tag{6.21}$$

and the *escape coefficients* $\boldsymbol{\mathfrak{E}}_{\alpha^+}$ are:

$$\begin{aligned}
\boldsymbol{\mathfrak{E}}_{\alpha_v^+} &= \int_{\alpha_v^+} d\vec{r}_s^{\,+} \int_{2\pi^+} \frac{d\vec{\Omega}}{4\pi} \; |\vec{\Omega} \cdot \hat{n}| \; \vec{\boldsymbol{\mathcal{Z}}}_S(\tilde{z}_s^{\,+}, \vec{\Omega}) \otimes \vec{\boldsymbol{W}}_V(\tilde{z}_s^{\,-}, \vec{\Omega}), \\
\boldsymbol{\mathfrak{E}}_{\alpha_h^+} &= \int_{\alpha_h^+} d\vec{r}_s^{\,+} \int_{2\pi^+} \frac{d\vec{\Omega}}{4\pi} \; |\vec{\Omega} \cdot \hat{n}| \; \vec{A}_S(\vec{\Omega}) \otimes \vec{\boldsymbol{W}}_V(\tilde{z}_s^{\,-}, \vec{\Omega}),
\end{aligned} \tag{6.22}$$

and $\vec{\boldsymbol{W}}_V(\tilde{z}_s^{\,-}, \vec{\Omega})$ reads:

$$\left(\vec{\boldsymbol{W}}_V(\tilde{z}_s^{\,-}, \vec{\Omega}) \right)_{n,p} = \frac{A_n(\vec{\Omega})}{\Sigma_r} \sum_{k=0}^{p} c_{p,k} \; P_k(\tilde{z}_s^{-}) \; \mu^{p-k} \left(\frac{2}{\Delta z} \right)^{p-k} E_{p-k}(\tau). \tag{6.23}$$

It is important to note that we chose to consider different the current vector dimension used in transmission with respect to that appearing in the balance Eq.(6.11). This is to reduce the cost of the transmission problem, which is the most expensive part of the DP_N method.

We use at this point Eqs.(6.15) to replace the flux with the currents terms is Eq.(6.20), obtaining:

$$\begin{bmatrix} \vec{\boldsymbol{J}}^{+}_{\alpha_v} \\ \vec{J}^{+}_{\alpha_h} \end{bmatrix} = \sum_{\beta \in r} \begin{bmatrix} \tilde{\boldsymbol{\mathfrak{T}}}_{\alpha_v^+ \beta_v^-} & \tilde{\boldsymbol{\mathfrak{T}}}_{\alpha_v^+ \beta_h^-} \\ \tilde{\boldsymbol{\mathfrak{T}}}_{\alpha_h^+ \beta_v^-} & \tilde{\boldsymbol{\mathfrak{T}}}_{\alpha_h^+ \beta_h^-} \end{bmatrix} \begin{bmatrix} \vec{\boldsymbol{J}}^{-}_{\beta_v} \\ \vec{J}^{-}_{\beta_h} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\mathfrak{E}}_{\alpha_v^+} \\ \boldsymbol{\mathfrak{E}}_{\alpha_h^+} \end{bmatrix} \vec{\boldsymbol{q}}_r, \tag{6.24}$$

where:

$$\begin{aligned}
\tilde{\boldsymbol{\mathfrak{T}}}_{\alpha_v^+ \beta_v^-} &= \boldsymbol{\mathfrak{T}}_{\alpha_v^+ \beta_v^-} \; \boldsymbol{\mathcal{Z}}^{-1}_{\beta_v^-}, & \tilde{\boldsymbol{\mathfrak{T}}}_{\alpha_v^+ \beta_h^-} &= \boldsymbol{\mathfrak{T}}_{\alpha_v^+ \beta_h^-} \; \bar{\bar{\mathcal{A}}}^{-1}_{\beta_h^-}, \\
\tilde{\boldsymbol{\mathfrak{T}}}_{\alpha_h^+ \beta_v^-} &= \boldsymbol{\mathfrak{T}}_{\alpha_h^+ \beta_v^-} \; \boldsymbol{\mathcal{Z}}^{-1}_{\beta_v^-}, & \tilde{\boldsymbol{\mathfrak{T}}}_{\alpha_h^+ \beta_h^-} &= \boldsymbol{\mathfrak{T}}_{\alpha_h^+ \beta_h^-} \; \bar{\bar{\mathcal{A}}}^{-1}_{\beta_h^-},
\end{aligned} \tag{6.25}$$

$$\vec{\boldsymbol{\Phi}}_r^{\alpha_v^\pm} = \boldsymbol{\mathcal{Z}}^{-1}_{\alpha_v^\pm} \; \vec{\boldsymbol{J}}^{\pm}_{\alpha_v} \qquad \& \qquad \vec{\Phi}_r^{\alpha_h^\pm} = \bar{\bar{\mathcal{A}}}^{-1}_{\alpha_h^\pm} \; \vec{J}^{\pm}_{\alpha_h}. \tag{6.26}$$

Much care has to be paid in the *tildation* process described above. The final form of Eq.(6.24) is only possible if $\mathfrak{Z}_\beta$ and $\bar{\bar{\mathcal{A}}}_\beta$ are invertible. The practice shows also that many difficulties arise even when, although not singular, the matrix is ill-conditioned. A small surface that is not crossed by a sufficiently large number of trajectories is the typical case for which the inversion could result ill-posed. In this case, the approach followed consists in reducing the matrix rank until a decent numerical inversion is obtained. The remaining elements are set to zero. The chosen criterion that classifies the inversion process quality is the value of the principal pivot of the Gauss direct inversion algorithm, as compared to the first diagonal element of the matrix. Even if this approach is rather heuristic, it is necessary in order to obtain a stable algorithm. The alternative would be to use a tracking integration step so fine to have a good representation of every surface integrals, but this would of course result in a too expensive calculation. The same procedure used for the SC method is applied also for the polynomial counterpart. A more detailed explanation of this process can be found in [49].

Final DP_N Balance

The final balance equations for the DP_N problem are obtained expressing the flux vector only as a function of the entering currents. A number of subtleties are to be used in this phase, since the number of angular moments in the balance equation (6.17) is larger than the one used in Eq.(6.20). Hence, we have to distinguish two possibilities.

The first sub-case treats those moments $V \subseteq D$, that we will indicate as ${}'\vec{\mathbf{\Phi}}_{r,V}$, viz those that are comprised in the range of angular moments of Eqs.(6.20). Using Eqs.(6.20) to express the exiting currents of Eq.(6.17), we can write:

$$
\begin{aligned}
(\Sigma_r - \mathfrak{D})\ {}'\vec{\mathbf{\Phi}}_{r,V} = \mathfrak{Z}_{VV}\ \vec{\boldsymbol{q}}_r - \frac{1}{V_r}\sum_{\alpha_v\in r}\left[\sum_{\beta_v\in r}\tilde{\mathfrak{T}}_{\alpha_v^+\beta_v^-}\ \vec{\boldsymbol{J}}^-_{\beta_v} + \sum_{\beta_h\in r}\tilde{\mathfrak{T}}_{\alpha_v^+\beta_h^-}\ \vec{J}^-_{\beta_h} + \boldsymbol{\mathcal{E}}_{\alpha_v^+}\ \vec{\boldsymbol{q}}_r - \vec{\boldsymbol{J}}^-_{\alpha_v}\right] \\
-\frac{1}{V_r}\sum_{\alpha_h\in r}\left[\vec{P}_b(\alpha_h^+)\otimes\left(\sum_{\beta_v\in r}\tilde{\mathfrak{T}}_{\alpha_h^+\beta_v^-}\ \vec{\boldsymbol{J}}^-_{\beta_v} + \sum_{\beta_h\in r}\tilde{\mathfrak{T}}_{\alpha_h^+\beta_h^-}\ \vec{J}^-_{\beta_h} + \boldsymbol{\mathcal{E}}_{\alpha_h^+}\ \vec{\boldsymbol{q}}_r\right) - \vec{P}_b(\alpha_h^-)\otimes\vec{J}^-_{\alpha_h}\right].
\end{aligned}
$$

Now inverting the sum order we obtain:

$$
\begin{aligned}
\sum_{\alpha_v\in r}\sum_{\beta_v\in r}\tilde{\mathfrak{T}}_{\alpha_v^+\beta_v^-}\ \vec{\boldsymbol{J}}^-_{\beta_v} + \sum_{\alpha_v\in r}\sum_{\beta_h\in r}\tilde{\mathfrak{T}}_{\alpha_v^+\beta_h^-}\ \vec{J}^-_{\beta_h} + \sum_{\alpha_h\in r}\vec{P}_b(\alpha_h^+)\otimes\sum_{\beta_v\in r}\tilde{\mathfrak{T}}_{\alpha_h^+\beta_v^-}\ \vec{\boldsymbol{J}}^-_{\beta_v} \\
+\sum_{\alpha_h\in r}\vec{P}_b(\alpha_h^+)\otimes\sum_{\beta_h\in r}\tilde{\mathfrak{T}}_{\alpha_h^+\beta_h^-}\ \vec{J}^-_{\beta_h} = \sum_{\beta_v\in r}\tilde{\mathfrak{T}}_{\beta_v^-}\ \vec{\boldsymbol{J}}^-_{\beta_v} + \sum_{\beta_h\in r}\tilde{\mathfrak{T}}_{\beta_h^-}\ \vec{J}^-_{\beta_h}.
\end{aligned}
$$

where:

$$
\tilde{\mathfrak{T}}_{\beta_v^-} = \sum_{\alpha_v\in r}\tilde{\mathfrak{T}}_{\alpha_v^+\beta_v^-} + \sum_{\alpha_h\in r}\vec{P}_b(\alpha_h^+)\otimes\tilde{\mathfrak{T}}_{\alpha_h^+\beta_v^-} \qquad \textit{and} \qquad \tilde{\mathfrak{T}}_{\beta_h^-} = \sum_{\alpha_v\in r}\tilde{\mathfrak{T}}_{\alpha_v^+\beta_h^-} + \sum_{\alpha_h\in r}\vec{P}_b(\alpha_h^+)\otimes\tilde{\mathfrak{T}}_{\alpha_h^+\beta_h^-}. \tag{6.27}
$$

A similar procedure applies to the *escape terms*:

$$
\boldsymbol{\mathcal{E}}_r = \frac{1}{V_r}\left[\sum_{\alpha_v\in r}\boldsymbol{\mathcal{E}}_{\alpha_v^+} + \sum_{\alpha_h\in r}\vec{P}_b(\alpha_h^-)\otimes\boldsymbol{\mathcal{E}}_{\alpha_h^+}\right]. \tag{6.28}
$$

With these notations we can write:

$$(\Sigma_r - \mathfrak{D})\ {}'\vec{\mathbf{\Phi}}_{r,V} = (\mathfrak{Z} - \mathfrak{E}_r)_{VV}\ \vec{q}_r + \frac{1}{V_r}\left[\sum_{\beta_v \in r}\left(\mathcal{I}_d - \tilde{\mathfrak{T}}_{\beta_v^-}\right)_{VS}\ \vec{J}^-_{\beta_v} + \sum_{\beta_h \in r}\left(\vec{P}_b(\beta_h^-)\otimes \bar{\bar{I}}_d - \tilde{\mathfrak{T}}_{\beta_h^-}\right)_{VS}\ \vec{J}^-_{\beta_h}\right]. \tag{6.29}$$

In the previous formulas we have distinguished the horizontal and vertical contributions. It is worthy to note that the horizontal part is an operator of dimension $V \times N_{m,S}$. Moreover, the output-input dimension for matrices $\tilde{\mathfrak{T}}_{\alpha^-}$ and $\mathfrak{E}$ are "VS" and "VV" respectively, and are not directly indicated in previous formulas to enlighten the notation.

The second sub-case considers the moments referred to as ${}'\vec{\mathbf{\Phi}}_{r,D/V}$, which designates the complementary part of "D", not considered by the "V". For the part of the "D" output currents that does not belong to the "V" set, Eq.(6.29) remains valid but this time $\tilde{\mathfrak{T}}_{\alpha^-}$ and $\mathfrak{E}_r$ cannot be computed from transmission matrices terms, and must be independently accumulated in tracking sweep. In other words the previous balance equation (6.29) was written under the hypotheses that all currents present in it could be expressed through (6.20). But to limit the cost of our DP_N operator the size of this last problem is maintained under a limited dimension which we have previously defined as "S". For moments of higher order than "V" the balance equation is written using a low-order closure as:

$$(\Sigma_r - \mathfrak{D})\ {}'\vec{\mathbf{\Phi}}_{r,D/V} = (\mathfrak{Z} - \mathfrak{E}_r)_{D/V,V}\ \vec{q}_r + \frac{1}{V_r}\left[\sum_{\beta_v \in r}\left(\mathcal{H}^{D/V}_{\beta_v^-} - \tilde{\mathfrak{T}}_{\beta_v^-}\right)\ \vec{J}^-_{\beta_v} + \sum_{\beta_h \in r}\left(\vec{P}_b(\beta_h^-)\otimes \bar{\bar{H}}^{D/V}_{\beta_h^-} - \tilde{\mathfrak{T}}_{\beta_h^-}\right)\ \vec{J}^-_{\beta_h}\right]. \tag{6.30}$$

where:

$$\mathcal{H}^{D/V}_{\beta_v^-} = \mathfrak{Z}_{(D/V,S),\beta_v^-}\ \mathfrak{Z}^{-1}_{(S,S),\beta_v^-} \qquad \textit{and} \qquad \bar{\bar{H}}^{D/V}_{\beta_h^-} = \bar{\bar{\mathcal{A}}}_{(D/V,S),\beta_h^-}\ \bar{\bar{\mathcal{A}}}^{-1}_{(S,S),\beta_h^-}.$$

The vectors $\vec{J}^{\pm}_{\alpha}$ are strictly dimensioned to the "S" transmission problem (6.20). In the previous expression $\mathfrak{Z}_{(S,S),\alpha^\pm}$ is a square matrix of dimension "S" while $\mathfrak{Z}_{(D/V,S),\alpha^\pm}$ a rectangular matrix of dimension "$D/V, S$". The two sets of Eqs. (6.29) and (6.30) constitute the full balance system where the used closure relation can be constructed by saying that higher (than "S") order currents are computed by supposing the "S" DP_N expansion (6.9) for the surface angular flux.

Recalling now Eq.(5.11) for the self-scattering term, we get:

$$\begin{aligned}(\Sigma_r - \mathfrak{D})\ {}'\vec{\mathbf{\Phi}}_{r,V} &= (\mathfrak{Z} - \mathfrak{E}_r)_{VV}\ (\mathbf{\Sigma}^g_{r,s}\ \vec{\mathbf{\Phi}}_{r,V} + \vec{q}_r^{\,ext}) + \\ &+ \frac{1}{V_r}\left[\sum_{\beta_v \in r}\left(\mathcal{I}_d - \tilde{\mathfrak{T}}_{\beta_v^-}\right)\ \vec{J}^-_{\beta_v} + \sum_{\beta_h \in r}\left(\vec{P}_b(\beta_h^-)\otimes \bar{\bar{I}}_d - \tilde{\mathfrak{T}}_{\beta_h^-}\right)\ \vec{J}^-_{\beta_h}\right], \\ (\Sigma_r - \mathfrak{D})\ {}'\vec{\mathbf{\Phi}}_{r,D/V} &= (\mathfrak{Z} - \mathfrak{E}_r)_{D/V,V}\ (\mathbf{\Sigma}^g_{r,s}\ \vec{\mathbf{\Phi}}_{r,V} + \vec{q}_r^{\,ext}) + \\ &+ \frac{1}{V_r}\left[\sum_{\beta_v \in r}\left(\mathcal{H}^{D/V}_{\beta_v^-} - \tilde{\mathfrak{T}}_{\beta_v^-}\right)\ \vec{J}^-_{\beta_v} + \sum_{\beta_h \in r}\left(\vec{P}_b(\beta_h^-)\otimes \bar{\bar{H}}^{D/V}_{\beta_h^-} - \tilde{\mathfrak{T}}_{\beta_h^-}\right)\ \vec{J}^-_{\beta_h}\right].\end{aligned}$$

Next we define:

$$\mathfrak{C}_{DV} = \mathfrak{Z} - \mathfrak{E}_r \qquad \mathfrak{X} = \Sigma_r \mathcal{I}_d - \mathfrak{D} - \mathfrak{C}_{DV}\ \mathbf{\Sigma}^g_{r,s}\ \mathfrak{P}^{-1}_{VV}, \tag{6.31}$$

where we have used Eq.(5.32) to substitute:

$$\vec{\Phi}_{r,V} = \mathfrak{P}_{VV}^{-1}\ {}'\vec{\Phi}_{r,V}.$$

With these definitions we get:

$$\begin{aligned}\mathfrak{X}\ {}'\vec{\Phi}_{r,D} =& \mathfrak{C}_{DV}\ \vec{q}_r^{\,ext} + \\ & \frac{1}{V_r}\left\{\sum_{\beta_v\in r}\begin{bmatrix}\mathcal{I}_d - \tilde{\mathfrak{T}}_{\beta_v^-} \\ \mathcal{H}^{D/V}_{\beta_v^-} - \tilde{\mathfrak{T}}_{D/V,\beta_v^-}\end{bmatrix}\ \vec{J}^-_{\beta_v} + \sum_{\beta_h\in r}\begin{bmatrix}\vec{P}_b(\beta_h^-)\otimes\bar{\bar{I}}_d - \tilde{\mathfrak{T}}_{\beta_h^-} \\ \vec{P}_b(\beta_h^-)\otimes\bar{\bar{H}}^{D/V}_{\beta_h^-} - \tilde{\mathfrak{T}}_{D/V,\beta_h^-}\end{bmatrix}\ \vec{J}^-_{\beta_h}\right\}.\end{aligned} \tag{6.32}$$

Inverting the matrix on the left hand side we can obtain an expression for the polynomial and angular flux moments, as a function of external sources and currents. The result of this procedure would be more general than is necessary since only "scattering" moments are needed. Considering then the projection over the only part of flux moments that is used in the rest of the DP_N operator, one has:

$${}'\vec{\Phi}_{r,V} = \tilde{\mathfrak{C}}\ \vec{q}_r^{\,ext} + \sum_{\beta\in r}\begin{bmatrix}\tilde{\mathfrak{J}}_{\beta_v} & \tilde{\mathfrak{J}}_{\beta_h}\end{bmatrix}\ \begin{bmatrix}\vec{J}^-_{\beta_v} \\ \vec{J}^-_{\beta_h}\end{bmatrix}, \tag{6.33}$$

$${}'\vec{\Phi}_{r,V} = \tilde{\mathfrak{C}}\ \vec{q}_r^{\,ext} + \sum_{\beta\in r}\tilde{\mathfrak{J}}_{\beta}\ \vec{J}^-_{\beta}$$

where:

$$\begin{aligned}\tilde{\mathfrak{C}} &= \mathfrak{X}_{VD}^{-1}\ \mathfrak{C}_{DV}, && \text{"}\textit{Collision term}\text{"},\\ \tilde{\mathfrak{J}}_{\beta_v} &= \frac{1}{V_r}\ \mathfrak{X}_{VD}^{-1}\ \begin{bmatrix}\mathcal{I}_d - \tilde{\mathfrak{T}}_{\beta_v^-} \\ \mathcal{H}^{D/V}_{\beta_v^-} - \tilde{\mathfrak{T}}_{D/V,\beta_v^-}\end{bmatrix} && \text{"}\textit{Incoming vertical}\text{"},\\ \tilde{\mathfrak{J}}_{\beta_h} &= \frac{1}{V_r}\ \mathfrak{X}_{VD}^{-1}\ \begin{bmatrix}\vec{P}_b(\beta_h^-)\otimes\bar{\bar{I}}_d - \tilde{\mathfrak{T}}_{\beta_h^-} \\ \vec{P}_b(\beta_h^-)\otimes\bar{\bar{H}}^{D/V}_{\beta_h^-} - \tilde{\mathfrak{T}}_{D/V,\beta_h^-}\end{bmatrix} && \text{"}\textit{Incoming horizontal}\text{"}.\end{aligned} \tag{6.34}$$

Equation (6.33) can be expressed by saying that the flux in a computational region is given by the collision of emission densities plus the incoming contribution of entering currents.

The final step consists in switching from the polynomial moments, to the expansion coefficients. Using Eq.(5.32) we write:

$$\vec{\Phi}_{r,V} = {}^{\dagger}\mathfrak{C}\ \vec{q}_r^{\,ext} + \sum_{\beta\in r}\begin{bmatrix}{}^{\dagger}\mathfrak{J}_{\beta_v} & {}^{\dagger}\mathfrak{J}_{\beta_h}\end{bmatrix}\ \begin{bmatrix}\vec{J}^-_{\beta_v} \\ \vec{J}^-_{\beta_h}\end{bmatrix}, \tag{6.35}$$

where:

$${}^{\dagger}\mathfrak{C} = \mathfrak{P}_{VV}^{-1}\ \tilde{\mathfrak{C}}, \qquad {}^{\dagger}\mathfrak{J}_{\beta_v} = \mathfrak{P}_{VV}^{-1}\ \tilde{\mathfrak{J}}_{\beta_v}, \qquad {}^{\dagger}\mathfrak{J}_{\beta_h} = \mathfrak{P}_{VV}^{-1}\ \tilde{\mathfrak{J}}_{\beta_h}.$$

Final Currents System

Starting again from Eqs.(6.24) and using Eq.(5.11) for the source term:

$$\begin{bmatrix}\vec{J}^+_{\alpha_v} \\ \vec{J}^+_{\alpha_h}\end{bmatrix} = \sum_{\beta\in r}\begin{bmatrix}\tilde{\mathfrak{T}}_{\alpha_v^+\beta_v^-} & \tilde{\mathfrak{T}}_{\alpha_v^+\beta_h^-} \\ \tilde{\mathfrak{T}}_{\alpha_h^+\beta_v^-} & \tilde{\mathfrak{T}}_{\alpha_h^+\beta_h^-}\end{bmatrix}\ \begin{bmatrix}\vec{J}^-_{\beta_v} \\ \vec{J}^-_{\beta_h}\end{bmatrix} + \begin{bmatrix}\mathfrak{E}_{\alpha_v^+} \\ \mathfrak{E}_{\alpha_h^+}\end{bmatrix}\ \left(\Sigma^g_{s,r}\ \vec{\Phi}_{r,V} + \vec{q}_r^{\,ext}\right).$$

Now, the flux term is expressed through Eq.(6.35), getting:

$$\begin{bmatrix}\vec{J}^{+}_{\alpha_v}\\ \vec{J}^{+}_{\alpha_h}\end{bmatrix} = \sum_{\beta\in r}\begin{bmatrix}\tilde{\mathfrak{T}}_{\alpha_v^+\beta_v^-} & \tilde{\mathfrak{T}}_{\alpha_v^+\beta_h^-}\\ \tilde{\mathfrak{T}}_{\alpha_h^+\beta_v^-} & \tilde{\mathfrak{T}}_{\alpha_h^+\beta_h^-}\end{bmatrix}\begin{bmatrix}\vec{J}^{-}_{\beta_v}\\ \vec{J}^{-}_{\beta_h}\end{bmatrix} + \\ + \begin{bmatrix}\mathcal{E}_{\alpha_v^+}\\ \mathcal{E}_{\alpha_h^+}\end{bmatrix}\left\{\Sigma^g_{s,r}\begin{bmatrix}{}^{\dagger}\mathcal{C}\ \vec{q}^{\,ext}_r + \sum_{\beta\in r}\left({}^{\dagger}\mathfrak{I}_{\beta_v} \quad {}^{\dagger}\mathfrak{I}_{\beta_h}\right)\begin{pmatrix}\vec{J}^{-}_{\beta_v}\\ \vec{J}^{-}_{\beta_h}\end{pmatrix}\end{bmatrix} + \vec{q}^{\,ext}_r\right\}.$$

Rearranging everything:

$$\begin{bmatrix}\vec{J}^{+}_{\alpha_v}\\ \vec{J}^{+}_{\alpha_h}\end{bmatrix} = \sum_{\beta\in r}\begin{bmatrix}\tilde{\mathfrak{T}}_{\alpha_v^+\beta_v^-}\tilde{\mathfrak{T}}_{\alpha_v^+\beta_h^-}\\ \tilde{\mathfrak{T}}_{\alpha_h^+\beta_v^-}\tilde{\mathfrak{T}}_{\alpha_h^+\beta_h^-}\end{bmatrix}\begin{bmatrix}\vec{J}^{-}_{\beta_v}\\ \vec{J}^{-}_{\beta_h}\end{bmatrix} + \\ + \begin{bmatrix}\mathcal{E}_{\alpha_v^+}\\ \mathcal{E}_{\alpha_h^+}\end{bmatrix}\Sigma^g_{s,r}\sum_{\beta\in r}\begin{bmatrix}{}^{\dagger}\mathfrak{I}_{\beta_v} & {}^{\dagger}\mathfrak{I}_{\beta_h}\end{bmatrix}\begin{bmatrix}\vec{J}^{-}_{\beta_v}\\ \vec{J}^{-}_{\beta_h}\end{bmatrix} + \begin{bmatrix}\mathcal{E}_{\alpha_v^+}\\ \mathcal{E}_{\alpha_h^+}\end{bmatrix}\left(\Sigma^g_{s,r}\ {}^{\dagger}\mathcal{C} + \mathcal{I}_d\right)\ \vec{q}^{\,ext}_r.$$

We define at this point:

$$\vec{J}^{+} = \left\{\vec{J}^{+}_{\alpha},\ \alpha = 1, N_{curr}\right\},$$

where $\vec{J}_{\alpha}$ regroups both vertical and horizontal contributions and N_{curr} is the number of surfaces in which the region is decomposed. The same definition applies to incoming currents and to all matrices. Using this definition we can write:

$$\vec{J}^{+} = \left(\tilde{\mathfrak{T}} + \vec{\mathcal{E}}^{+}\Sigma^g_{s,r}\ {}^{\dagger}\mathfrak{I}\right)\ \vec{J}^{-} + \vec{\mathcal{E}}^{+}\left(\mathcal{I}_d + \Sigma^g_{s,r}\ {}^{\dagger}\mathcal{C}\right)\ \vec{q}^{\,ext}_r,$$

and we obtain a final system of equation for the currents:

$$\vec{J}^{+} = {}^{\dagger}\mathfrak{T}\ \vec{J}^{-} + {}^{\dagger}\mathcal{E}\ \vec{q}^{\,ext}_r, \tag{6.36}$$

where:

$${}^{\dagger}\mathfrak{T} = \left(\tilde{\mathfrak{T}} + \vec{\mathcal{E}}^{+}\Sigma^g_{s,r}\ {}^{\dagger}\mathfrak{I}\right) \qquad and \qquad {}^{\dagger}\mathcal{E} = \vec{\mathcal{E}}^{+}\left(\mathcal{I}_d + \Sigma^g_{r,s}\ {}^{\dagger}\mathcal{C}\right). \tag{6.37}$$

Equation (6.36) can be resumed by saying that the outgoing currents are given by the multi-collisional contribution of entering currents and escape of external sources.

Linear system solution

The current vector is the unknown of the linear system of equations defined by Eq.(6.36), that we can briefly write as:

$$\mathcal{A}\ \vec{J} = \vec{b}$$

$$\mathcal{A} = \mathcal{I}_d - {}^{\dagger}\mathfrak{T} \qquad\qquad \vec{b} = {}^{\dagger}\mathcal{E}\ \vec{q}^{\,ext}_r$$

The solution of this system is obtained with an iterative method, particularly suited for large sparse systems. The solution of this problem for the polynomial method has no real difference from the SC one, asides from the matrix size. The already implemented solution algorithm, based on Krylov subspace iterative method, and developed for the solution of SC 3D problems, has been used and no further developments have been realized during this work in this direction. Reference [26] gives a detailed description of how the preconditioning technique implemented for the two-dimensional solver has been adapted to the 3D version. The parallel algorithm implemented to obtain the linear system solution is also described in this paper.

7. Practical aspects concerning the DP_N synthetic acceleration

This chapter deals with three aspects of the practical implementation of the DP_N coefficients calculation: the parallel strategy adopted during the computation phase, the procedure used to exploit the CCM to strongly reduce the computational cost associated to the coefficients calculation and the non-linear least squares fitting method that we have implemented to reduce the coefficients memory size.

7.1 About the DP_N coefficients calculation

Coherence between the acceleration and transport solution requires the acceleration matrices to be computed with the same trajectory-based spatial discretization used for the MOC sweep. Since the acceleration is not meant to necessarily deliver a precise solution, this consistency can be relaxed, in order to decrease the acceleration computational burden. However, if the two approximations differ too much, instabilities issues may appear. In practical terms, this means that each term of Eqs.(6.21) and Eqs.(6.22) have to be numerically computed with each 3D chord of the system. Unfortunately, the computational cost of this operation is larger for the Polynomial method, in comparison with the SC one. The polynomial method issue is not constituted by the fact that the matrices are larger, since this increased size is largely counter-balanced by the decrease of the matrices number due to the axial meshes number reduction. The computational over cost of the polynomial method, in comparison with the SC equivalent, is mainly due to the operations required to compute the $\boldsymbol{\mathcal{E}}_\alpha$ terms of Eqs.(6.22). For the Step method, in fact, there is no real need to directly compute these *escape* terms. A so-called *complementary* relation between *transmission* and *escape* terms allows in fact to compute the latter as a function of the former. For the polynomial method, a relation of this kind could be obtained, but with a greater complexity in comparison with the Step method. Consequently, we have preferred to use a straightforward approach, and to directly compute both the *escape*, and the *transmission* terms.

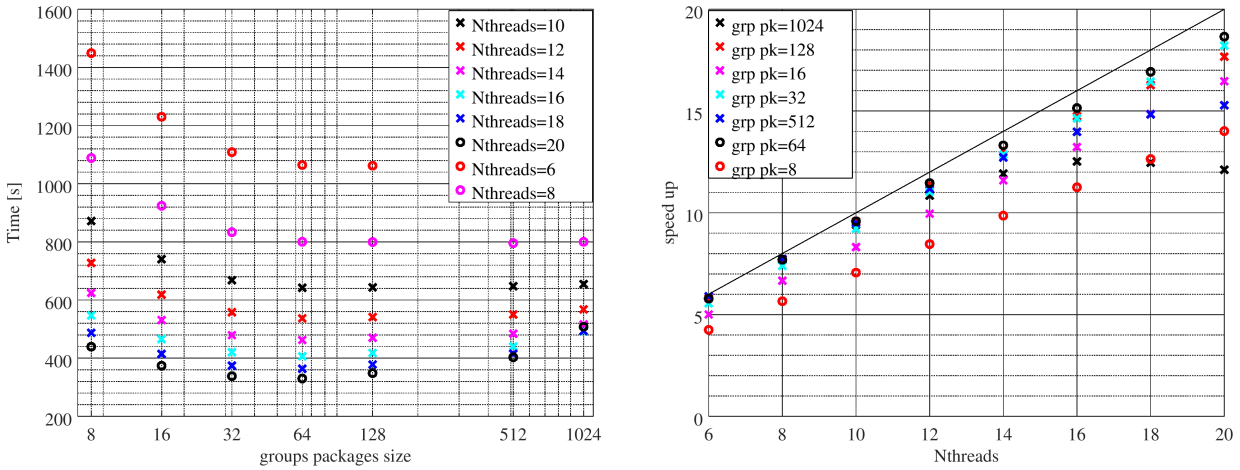
The parallel strategy that was implemented for the coefficients construction of the SC method had to be slightly changed in order to make it more efficient. Otherwise, the coefficients construction would have represented a very large part of the total computational time, when using the polynomial method. The previous parallel strategy consisted in computing the DP_N coefficients performing a serial trajectories sweep and cumulating for each chord the corresponding coefficient (Eqs.(6.21) and (6.22)). The parallel version of this algorithm simply divides the group numbers in *packages* among the different threads. While performing the trajectory sweep, each thread cumulates only the values in the group package into the final storage structures, shared among all the threads. Although of simple implementation, this method had some drawbacks. First, the load balancing is not very

efficient, especially if the number of groups is low. Secondly, some operations proper to the trajectory sweep are repeated by multiple threads. Thirdly, all threads work on large matrices, shared among all processors.

Noticing a speed-up factor stagnation when increasing the threads number, we decided to adopt a different strategy: instead of applying the parallel algorithm to the group number, we apply it to the 2D regions. The idea is that whereas the total number of groups can be low, the number of 2D computational regions is always large. This allows a better workload balancing among threads. Moreover, we used private variables for the matrices construction. Reducing the matrices computational sizes per-region values, permits in fact to duplicate this information for all the threads, without a noticeable increase in memory footprint. The private copies allowed a faster information access to each thread.

In a first phase, each thread had to repeat the whole trajectory sweep, in order to cumulate the chord related values only if the considered chord belonged to the considered 2D region. We abandoned this strategy since for large cases the cost related to the repeated trajectories reconstruction and sweep was important. The final strategy that we have adopted consists in a preliminary phase in which all the necessary informations are gathered through a trajectory sweep, followed by the actual computation. During the preliminary phase, the number of classified and unclassified chords for each 2D region is computed and stored. For each class or unclassified chord all the information needed for the coefficients computation are stored (length, exiting/entering surfaces, etc.). Once these structures are created, the parallel construction can begin. For each two-dimensional region, each thread can access the necessary information to construct the DP_N coefficients without any need to reconstruct the trajectory information. Aside from the first data gathering phase, which is performed using multiple threads but needs to use some mutual exclusion mechanism, the rest of the algorithm can be, at least in theory, performed in a perfect parallel manner, without any operation repetition for different threads or race conditions.

The final issue that we have encountered is related to the private variable sizes, which directly affected the coefficients computation times, more than what initially expected. In order to maximize the vectorization possibilities and to avoid additional complexity, we asked to each thread to compute the coefficients for all the energy groups used in the calculation. We believed that, since the amount of private data that each thread had to work with was small, when compared to the total final storage size, no memory size effect would have affected our calculation performances. Since each thread had to work only on one two-dimensional region, the private variables size corresponds to the total storage size divided by a factor at least between 10^2 and 10^3. We have tested this strategy with a maximum thread number of 20, and no appreciable effect on the total engaged memory was noticed. On the other hand, we have noticed that the total computational time suffers from the variable sizes, especially when increasing the threads number. In order to circumvent this problem we decided to assign to a thread both a 2D region and a user defined groups interval for the calculation. We run several tests for different *group package* sizes and for different number of threads, in order to verify the strategy. As Figures 20a and 20b show, a small *group package* size leads to poor performances since the vectorization degree is small. On the other hand, large *group package* size is the best choice if the number of threads is low, but it strongly affects the parallel efficiency if the thread number increases, as the speed-up factor clearly shows. For the case that we have tested and with a number of threads comprised between 1 and 20, by choosing a proper *group package* size it is possible to obtain an almost linear speed-up.



(a) (b)

Figure 20 – Computational time of the DP_N coefficients computation as a function of the threads numbers and of the group package (a) and speed-up factors (b). The case used for this test is the *half-column* sub-assembly of the ASTRID reactor. This case features 115 two-dimensional computational regions and 1968 energy groups.

DP_N coefficients and Classification

As before, the CCM introduced in [23] has an important application in the polynomial method. The CCM allows to reduce enormously the computational cost of the polynomial DP_N coefficients calculation. As briefly discussed in subsection 3.1.2, the CCM method applies to *V-chords* or *H-chords*. Since the use of the polynomial method translates into computational meshes with a very elongated aspect ratio, the *V-chords* are the most common chords type. We limit therefore the application of the chords classification method to the DP_N coefficients only for this chords type. We believe this choice not to be penalizing, since, if the geometry imposes very small axial meshes, for which the presence of *H-chords* is not a rare event, then it is also probable that the polynomial method will not offer the better performances, in comparison with the SC method.

Under these assumptions, we apply the CCM to the computation of a subset of the coefficients in Eqs.(6.21) and (6.22). In particular, we analyse:

$$\boldsymbol{\mathfrak{T}}_{\alpha_v^+\beta_v^-} = \int_{\alpha_v^+} d\vec{r}_s^{\,+} \int_{(\beta_v^- \to \alpha_v^+)} \frac{d\vec{\Omega}}{4\pi} \, |\vec{\Omega}\cdot\hat{n}| \; \vec{\boldsymbol{Z}}_S(\tilde{z}_s^{\,+},\vec{\Omega}) \otimes \vec{\boldsymbol{Z}}_S(\tilde{z}_s^{\,-},\vec{\Omega}) \; e^{-\tau},$$

$$\boldsymbol{\mathcal{E}}_{\alpha_v^+} = \int_{\alpha_v^+} d\vec{r}_s^{\,+} \int_{2\pi^+} \frac{d\vec{\Omega}}{4\pi} \; |\vec{\Omega}\cdot\hat{n}| \; \vec{\boldsymbol{Z}}_S(\tilde{z}_s^{\,+},\vec{\Omega}) \otimes \vec{\boldsymbol{W}}_V(\tilde{z}_s^{\,-},\vec{\Omega}).$$

Using definitions (5.8) and (6.23), we can write more explicitly:

$$\left(\boldsymbol{\mathfrak{T}}_{\alpha_v^+\beta_v^-}\right)_{n,n',p,p'} = \int_{\alpha_v^+} d\vec{r}_s^{\,+} \int_{(\beta_v^-\to\alpha_v^+)} \frac{d\vec{\Omega}}{4\pi}\ |\vec{\Omega}\cdot\hat{n}|\ A_n(\vec{\Omega})\ A_{n'}(\vec{\Omega})\ P_p(\tilde{z}_s^+)\ P_{p'}(\tilde{z}_s^-)\, e^{-\tau}, \tag{7.1}$$

$$\left(\boldsymbol{\mathcal{E}}_{\alpha_v^+}\right)_{n,n',p,p'} = \int_{\alpha_v^+} d\vec{r}_s^{\,+} \int_{2\pi^+} \frac{d\vec{\Omega}}{4\pi}\ |\vec{\Omega}\cdot\hat{n}|\ \frac{1}{\Sigma_r} A_n(\vec{\Omega})\ A_{n'}(\vec{\Omega})\ P_p(\tilde{z}_s^+) \sum_{k=0}^{p'} c_{p',k}\ P_k(\tilde{z}_s^-)\ \mu^{p'-k} \left(\frac{2}{\Delta z}\right)^{p'-k} E_{p'-k}(\tau). \tag{7.2}$$

We analyse the *escape* coefficients, since this method will be extended easily also to the *transmission* terms. We recall that $\boldsymbol{\mathcal{E}}_{\alpha_v^+}$ indicates the vertical surface α exiting from a given region. Two chords types will contribute to this integrals: the ones starting from the vertical surfaces β_v^-, and the ones starting from the horizontal surfaces β_h^-. We can write these contributions as:

$$\boldsymbol{\mathcal{E}}_{\alpha_v^+} = \sum_{\beta_v} \boldsymbol{\mathcal{E}}_{\beta_v^-\to\alpha_v^+} + \sum_{\beta_h} \boldsymbol{\mathcal{E}}_{\beta_h^-\to\alpha_v^+}.$$

As anticipated we neglect the horizontal contribution in the application of the CCM method. Thus, we focus on the first term, and we write it as:

$$\left(\boldsymbol{\mathcal{E}}_{\alpha_v^+}\right)_{n,n',p,p'} = \int_{\alpha_v^+} d\vec{r}_s^{\,+} \int_{\beta_v^-\to\alpha_v^+} \frac{d\vec{\Omega}}{4\pi}\ |\vec{\Omega}\cdot\hat{n}|\ \frac{1}{\Sigma_r} A_n(\vec{\Omega})\ A_{n'}(\vec{\Omega})\ P_p(\tilde{z}_s^+) \sum_{k=0}^{p'} c_{p',k}\ P_k(\tilde{z}_s^-)\ \mu^{p'-k} \left(\frac{2}{\Delta z}\right)^{p'-k} E_{p'-k}(\tau). \tag{7.3}$$

At this point we consider the integral over the surface α_v^+ decomposed as:

$$\int_{\alpha_v^+} d\vec{r}_s^{\,+} = \int_{\alpha_{v,2D}^+} d_2 r_\perp \int_{\alpha_{v,z}^+} dz,$$

where we have separated the radial from the axial surface integral contributions. A further reformulation of the radial integral coherent with the trajectories-based discretization leads to:

$$\int_{\alpha_v^+} d\vec{r}_s^{\,+} = \sum_{\substack{i_{2D}\in\alpha_v^+ \\ \beta_v^-\to\alpha_v^+}} \int_{\alpha_{v,\tilde{z}}^+} d\tilde{z},$$

where i_{2D} designates a two-dimensional chord, and the sum gathers the 2D chords that are crossing the two considered surfaces. Since all the 3D chords that share the same 2D footprint and cross two vertical surfaces have the same length, they also share the same values for $E_{p'-k}(\tau)$. Taking advantage of this and using the previous formulation for the 2D integral we can write Eq.(7.3) as:

$$\left(\boldsymbol{\mathcal{E}}_{\alpha_v^+}\right)_{n,n',p,p'} = \sum_{\substack{i_{2D}\in\alpha_v^+ \\ \beta_v^-\to\alpha_v^+}} \int_{\beta_v^-\to\alpha_v^+} \frac{d\vec{\Omega}}{4\pi}\ |\vec{\Omega}\cdot\hat{n}| \frac{A_n(\vec{\Omega})\ A_{n'}(\vec{\Omega})}{\Sigma_r} \sum_{k=0}^{p'} c_{p',k}\ \bar{\bar{B}}_{i_{2D}}^{p,k}\ \mu^{p'-k} \left(\frac{2}{\Delta z}\right)^{p'-k} E_{p'-k}(\tau),$$

where:

$$\bar{\bar{B}}^{p,k}_{i_{2D}} = \int_{\alpha^+_{v,\tilde{z}}} d\tilde{z}\, P_p(\tilde{z}^+_s)\; P_k(\tilde{z}^-_s).$$

Since the matrices $\bar{\bar{B}}_{i_{2D}}$ are group independent, they are computed and stored during the tracking phase. Paying the reasonable price of storing these matrices, the computational cost of the *escape* coefficients for the classified *V-chords* is greatly reduced. Moreover, a similar formulation can be applied also to the vertical *transmission* term of Eq.(7.1), that becomes:

$$\left(\mathfrak{T}_{\alpha^+_v \beta^-_v}\right)_{n,n',p,p'} = \sum_{\substack{i_{2D}\in\alpha^+_v \\ \beta^-_v \to \alpha^+_v}} \int_{\beta^-_v \to \alpha^+_v} \frac{d\vec{\Omega}}{4\pi}\, |\vec{\Omega}\cdot\hat{n}| \frac{A_n(\vec{\Omega})\; A_{n'}(\vec{\Omega})}{\Sigma_r}\; \bar{\bar{B}}^{p,p'}_{i_{2D}}\; e^{-\tau}$$

We end this section underlying that if no *classification* were applied, the computational cost of the DP_N coefficients for the polynomial method would be probably high enough to make the polynomial approach not advantageous when compared to the Step method. In most of the cases we have tested, in fact, more than 90% of the chords are of the *V-type* and can be classified. Thanks to this evidence, the procedure just described allows a great reduction of the computational cost associated to the polynomial DP_N coefficients calculation, roughly corresponding to the number of 3D chords belonging to the VV classes, divided by the number of classes.

7.2 DP_N coefficients non-linear least squares fitting

The second major issue related to the acceleration strategy we have chosen, after the computationally expensive coefficients construction, is represented by the memory requirements related to the storage of the acceleration matrices. This problem is not intrinsically related to the polynomial method developed in this work. When comparing the polynomial method with the SC equivalent, we observed a significant reduction of the total memory requirement for the DP_N synthetic acceleration matrices for cases with similar precision. However, the matrices size grows with the chosen polynomial degree. For the largest matrices type, which are $\mathfrak{T}_{\alpha^+_v \beta^-_v}$ and $\mathfrak{E}_{\alpha^+_v}$ (Eqs.(6.21) and (6.22)), this growth is $\propto N_p^2$. It is easy to see that the problem related to the important memory requirements is far from being solved using the polynomial method, even if it is largely reduced in comparison with the SC, because of the supposedly smaller number of computational regions.

The problem is somehow similar to what happens for the short characteristics solvers. In this case, large group-dependent matrices of coefficients are used to compute the transport solution, and storage for a large number of groups can result prohibitive. To overcome this problem, the authors of [19], for example, adopt a bi-dimensional tabulation strategy to interpolate on-the-fly the coefficients set in the GENESIS code. The sometimes prohibitive memory requirements of the DP_N synthetic acceleration matrices for TDT can be addressed in a similar way. It has though the advantage that if the precision of such matrices is slightly degraded, the final transport result is not affected. The acceleration quality can suffer from poor matrices representation and if the approximations are too aggressive, this may also result in instabilities causing the calculation to diverge. This eventuality has of course to be avoided, but knowing that introducing a small error in the acceleration matrices does not affect the solution accuracy offers an interesting degree of freedom which is useful to achieve substantial gains in terms of memory requirements.

The large dimension of the acceleration matrices is due to both a spatial and an energy dependence of the acceleration coefficients. To compress the information related to the energy dependence of the synthetic matrices we adopted a strategy based on a non-linear data fitting, solving a least squares problem. The implemented method allows to avoid the storage of the DP_N matrices for each energy group. This information is replaced with a set of coefficients matrices used to reconstruct on-the-fly the mono-group set of coefficients. In terms of size, each matrix of coefficients coincides with the size used to store the set of DP_N matrices for one energy group. As a consequence, the method reduces the memory demand only if the number of coefficients used for the function fitting is lower than the number of energy groups used for the calculation.

The choice of the function type and the number of coefficients was the consequence of a series of considerations and practical tests. An alternative and more rigorous approach would have been to use interpolation tables. We considered this option at the beginning of this work phase, but we eventually choose a fitting approach for several reasons, which are summarized in the following.

Problem description and possible solutions

To apply the DP_N synthetic acceleration method described in Section 6, we need to solve Eqs.(6.36) and (6.35) for each energy group. This requires to have in memory, or to construct on-the-fly, the following matrices: $^{\dagger}\mathfrak{T}$, $^{\dagger}\mathfrak{E}$, $^{\dagger}\mathfrak{J}$ and $^{\dagger}\mathfrak{C}$. Note that the formulation development described in Sec.6, as well as the actual implementation, requires to treat differently vertical and horizontal surfaces. On the other hand, in the present discussion we will neglect this difference, since it does not affect the description.

The most straightforward approach consists in computing and storing these matrices in a preliminary phase, before starting the iterating procedure to solve the problem. Algorithm 2 gives a simplified representation of this approach, which is the default option.

One of the possible ways to avoid the storage of these matrices is of course to recompute them on the fly for each energy group. This possibility has not been directly investigated since the computational cost associated to these matrices is very important, as discussed in Section 7.1. The heaviest computation associated to these matrices calculation is represented by the trajectory-coherent evaluation of Eqs.(6.21) and (6.22). Following this part, a series of operations has to be done with these matrices, to arrive at the final formulation reported in Eqs.(6.36) and (6.35). Even if less computationally expensive than the first phase, repeating this second part at the beginning of each energy group would require an important number of operations to be performed. We stress the different computational aspects of these two stages, since we considered two possible ways to reduce the memory footprint.

The first possibility is to fit or tabulate the coefficients expressed by Eqs.(6.21) and (6.22) as a function of Σ (remember that $\tau = \Sigma\, l$) after the coefficients construction. When the solution of the synthetic problem is required for a given energy group, these coefficients have to be re-constructed, and then the suite of operations to arrive at the final form needed by Eqs.(6.36) and (6.35) has to be performed. This method has the obvious disadvantage that it requires a series of operation to be repeated at the beginning of each mono-group iteration. Please note that not only the operations described by Eqs. from (6.21) to (6.36) have to be done, but also there is a hidden cost associated to the preconditioning technique described in [26], which has to be added. The great advantage of this approach is that the expression of the functions that we want to approach is well known, and the only variable depending on the energy is the total cross section. As a consequence, a simple interpolation

Coefficients construction:

!$OMP PARALLEL
for $r_{2d} = 1, nb_{reg}$ **do**
  1) Compute $\mathfrak{T}$ and $\mathfrak{E}$ trajectories-coherently // Eqs.(6.21), (6.22)
  2) Compute $^{\dagger}\mathfrak{T}$,$^{\dagger}\mathfrak{E}$,$^{\dagger}\mathfrak{I}$,$^{\dagger}\mathfrak{C}$ // Eqs.(6.25), (6.27),(6.28),(6.31),(6.34), (6.37)
  3) ILU0 Preconditioning
  4) Store the final version of $^{\dagger}\mathfrak{T}$,$^{\dagger}\mathfrak{E}$,$^{\dagger}\mathfrak{I}$,$^{\dagger}\mathfrak{C}$
end
!$OMP END PARALLEL

Solution:

Start outer iterations loop:
while *the fission integral is not converged* **do**
  Iterates on groups, starting from the highest energy
  for $g = 1, N_g$ **do**
    1) Retrieve $^{\dagger}\mathfrak{T}$,$^{\dagger}\mathfrak{E}$,$^{\dagger}\mathfrak{I}$,$^{\dagger}\mathfrak{C}$ for the given energy group
    2) System solution // Eqs. (6.36),(6.35)
    3) etc...
  end
end

Algorithm 2: DP_N coefficients default option: direct storage.

table would give good results with a relatively low number of points. Another possible way to approach this problem, even more efficient than the use of an interpolation table, is, in our opinion, the use of a fitting function, whose coefficients are obtained solving a least squares problem. Consider, for example, the $\mathfrak{T}_{\alpha_v^+\beta_v^-}$ matrix, whose expression is:

$$\mathfrak{T}_{\alpha_v^+\beta_v^-} = \int_{\alpha_v^+} d\vec{r}_s^{\,+} \int_{(\beta_v^- \to \alpha_v^+)} \frac{d\vec{\Omega}}{4\pi} \; |\vec{\Omega}\cdot\hat{n}| \; \vec{\boldsymbol{\mathcal{Z}}}_S(\tilde{z}_s^{\,+}, \vec{\Omega}) \otimes \vec{\boldsymbol{\mathcal{Z}}}_S(\tilde{z}_s^{\,-}, \vec{\Omega})\, e^{-\tau},$$

and denoting by $\mathfrak{T}_i$ one element of the matrix, we can try to approximate the energetic dependence of this set of values (one for each energy group), as, for example:

$$f(\vec{\alpha}, \Sigma) = \alpha_1 \; e^{\alpha_2 \Sigma}.$$

The choice of the parametric function is of course quite heuristic, but is based on the fact that both *transmission* and *escape* coefficients described by Eqs.(6.21) and (6.22) are computed as a sum of elements whose dependence in energy is always in the form $e^{-\tau}$. Even if the *escape* coefficients do not show explicitly this dependence, since they are computed with the $E_b(\tau)$ terms expressed by Eq.(5.35) and using the recursive relation of Eq.(5.36), it is interesting to see that also these terms can be written as a sum of exponentials:

$$\begin{aligned}
E_0(\tau) &= 1 - e^{-\tau}, \\
E_1(\tau) &= 1 - \frac{1}{\Sigma} + \frac{1}{\Sigma} e^{-\tau}, \\
&\vdots \\
E_4(\tau) &= l^4 - \frac{4\cdot l^3}{\Sigma} + \frac{4\cdot 3\cdot l^2}{\Sigma^2} - \frac{4!\cdot l}{\Sigma^3} + \frac{4!}{\Sigma^4} - \frac{4!}{\Sigma^4} e^{-\tau}, \\
E_b^T(\tau) &= \sum_{i=0}^{b} (-1)^i \frac{b!}{(b-i)!} \frac{l^i}{\Sigma^i} + (-1)^{b+1} \frac{b!}{\Sigma^b} e^{-\tau}.
\end{aligned}$$

As a result from these considerations, we believe that, if the chosen approach was the one just described, this could be considered maybe a little bit exotic but justifiable. Since we can compute the expected coefficient $\mathfrak{T}_i$ values, we can obtain the fitting coefficients $\vec{\alpha}$ minimizing the residual defined by:

$$res(\vec{\alpha}) = \frac{1}{2}\sqrt{\sum_{g=1}^{N_g}\left(\mathfrak{T}_i^g - f(\vec{\alpha}, \Sigma^g)\right)^2},$$

as in a classical least squares problem, where the number of points used to obtain the fitting coincides with the number of energy groups N_g. Once the set of coefficients $\vec{\alpha}$ are computed for all the element of a matrix for a 2D region, the multi-group reference values $\mathfrak{T}_i$ can be discarded. Since this procedure can be (almost) done for each 2D region independently, the memory required to store the multi-group values only for the 2D regions computed simultaneously is not judged to be important.

The second possibility that we have explored is to represent not the first matrices expressed by Eqs.(6.21) and (6.22) but the final preconditioned versions used in Eqs.(6.36) and (6.35). This second approach has the advantage that the number of floating-point operations to be performed at the beginning of each group iteration to reconstruct the coefficients would be significantly reduced, if a proper way to obtain the final values can be found. The main disadvantage of this method is that the functions that we are trying to approximate are not any more dependent on a single variable. As we can see in Eqs.(6.31) and (6.37), a dependence on the scattering cross section appears. The implemented version of the DP_N synthetic acceleration employs at most a linear anisotropy approximation (DP_1). As a consequence, the functions that must be approximated following this approach depend on two or three variables (depending on the anisotropy order of the considered media) and they do not have any more a clear formulation, since they result from quite complicated matrix by matrix and matrix by vector products, plus some numerical inversion and finally the preconditioner-related operations. Given the three-variable dependence, the interpolation table approach would be, in our opinion, too complicated to implement in order to allow an effective memory compression. For this reason only the fitting method has been tested on the final matrices.

Even if more exotic and less justifiable, this second possibility has been the one we focused our efforts on, since we aimed to avoid that the floating point operations be repeated during iterations to obtain the final coefficients used in Eq.(6.36) starting from (6.27). To implement this method, we need a parametric function able to represent our coefficients set depending on two or three variables. After several attempts, we decided to settle down with the following formula:

$$f(\vec{\alpha}, \Sigma, \Sigma_{s,0}, \Sigma_{s,1}) = \alpha_1\, e^{\alpha_2\Sigma_r}\ (\Sigma)^{\alpha_3} + \alpha_4\, e^{\alpha_5\Sigma_r}\ (\Sigma_{s,0})^{\alpha_6} + \alpha_7\, e^{\alpha_8\Sigma_r}\ (\Sigma_{s,1})^{\alpha_9}\,.$$

Using this parametric function, a non-linear regression is performed on each element of the DP_N matrices. The number of elements per each matrices position, constituted by the number of energy groups, is replaced with the set of nine coefficients. In order to assure that the method delivers a chosen precision, the elements whose error exceeds this tolerance value are stored. In order to know the element values and position, the set of matrices has been thought as a unique row vector. In this way it is possible to store per each *non-fitted* element a real number containing the element value, and an integer number containing the element position. In this way two vectors per each energy group are sufficient to store all the elements that are not correctly represented by the regression model. Notice that each element not correctly fitted occupies twice the memory size in comparison to the same

element using the direct storage option, since we need to store also the element position. As a consequence, it is very important to keep the number of *non-fitted* elements as low as possible.

Some elements of the mathematical approach

The implemented non-linear least squares solver algorithm has been extracted from the work presented in [50]. We recall here some elements of the problem and of the possible solutions presented in the cited manuscript and using similar notations. For an exhaustive explanation we refer directly to [50].

Given a set of values (t_i, y_i) and a function $g(\vec{x}, t_i)$, we want to compute the set of parameter $\vec{x}$, in order to minimize the residual norm defined as:

$$F(\vec{x}) = \frac{1}{2} \sum_{i=1}^{N} \left(f_i(\vec{x})\right)^2 = \frac{1}{2}\, \vec{f}(\vec{x}) \cdot \vec{f}(\vec{x}), \tag{7.4}$$

where:

$$f_i(\vec{x}) = y_i - g(\vec{x}, t_i).$$

This has been done using a variant of the method presented in [50] as the Gauss-Newton method. The iterative procedure starts by defining the initial values of the parameters set $\vec{x}_0$. Then, the idea is to approximate $\vec{f}(\vec{x})$ with a Taylor expansion for a small neighbourhood of $\vec{x}$, $\vec{h}$:

$$\vec{f}(\vec{x} + \vec{h}) \simeq \vec{l}(\vec{h}) = \vec{f}(\vec{x}) + \bar{\bar{J}}(\vec{x})\ \vec{h}, \tag{7.5}$$

where $\bar{\bar{J}}$ is the *Jacobian* matrix:

$$\left(\bar{\bar{J}}(\vec{x})\right)_{i,j} = \frac{\partial f_i}{\partial x_j}(\vec{x}).$$

Replacing (7.5) in (7.4) we can write:

$$F(\vec{x} + \vec{h}) \simeq L(\vec{h}) = \frac{1}{2}\ \vec{l}(\vec{h}) \cdot \vec{l}(\vec{h}) = F(\vec{x}) + \vec{h}^T \bar{\bar{J}}^T \vec{f} + \frac{1}{2}\ \vec{h}^T \bar{\bar{J}}^T \bar{\bar{J}} \vec{h},$$

where $\bar{\bar{J}} = \bar{\bar{J}}(\vec{x})$ and $\vec{f} = \vec{f}(\vec{x})$. With the *Gauss-Newton step*, we can compute $\vec{h}$ in order to minimize $L(\vec{h})$, by imposing:

$$L'(\vec{h}) = \bar{\bar{J}}^T \vec{f} + \bar{\bar{J}}^T \bar{\bar{J}} \vec{h} = 0. \tag{7.6}$$

This can be solved for $\vec{h}$ and a new estimation of $\vec{x}$ can be obtained as:

$$\vec{x}_1 = \vec{x}_0 + \vec{h}.$$

The method converges, provided that (7.5) is a good approximation. Otherwise it may happen that

$$L(\vec{h}) < L(0) \quad but \quad F(\vec{x} + \vec{h}) > F(\vec{x}).$$

To stabilize the convergence of the solution, the *Levenberg-Marquardt* variant, again taken from [50], has been implemented. This method, originally proposed by [51], uses a *dumping*

parameter μ to stabilize the iterative procedure. Instead of solving (7.6) at each iteration, it uses:

$$\left(\bar{\bar{J}}^T \bar{\bar{J}} + \mu \bar{\bar{I}}_d\right) \vec{h} = -\bar{\bar{J}}^T \vec{f} \qquad with \quad \mu > 0,$$

where $\bar{\bar{I}}_d$ is the identity matrix. The dumping parameter is updated during the iterations, and get smaller and smaller while $F(\vec{x})$ decreases. For very small values of μ, the method becomes the same as the classical *Gauss-Newton* version. After each new evaluation of $\vec{h}$ the *gain ratio*:

$$\sigma = \frac{F(\vec{x}) - F(\vec{x} + \vec{h})}{L(0) - L(\vec{h})},$$

is used to establish if the new solution is better than the previous. For $\sigma < 0$ the new residual is larger than the previous, so the current iteration solution is discarded, the μ parameter is increased and a new estimation of $\vec{h}$ is obtained. For $\sigma > 0$ the new solution is better than the previous one and it is kept. In this case μ is decreased before proceeding to the next iteration.

Part III

Results & Conclusions

8. Results

This chapter is devoted to an exhaustive presentation of the results obtained using the polynomial method. Two kinds of comparisons are carried out in the following: a first comparison between step and polynomial methods and then a comparison between the Polynomial method and several reference Monte Carlo solutions. The comparison between the two deterministic methods is meant to show the differences in performances (memory usage and computational time) for a similar level of accuracy. On the other hand, the comparison with the Monte Carlo calculations is meant to validate the results obtained using the polynomial method. Unless differently specified, all the calculations have been run with the OpenMp parallel option activated and using 15 threads on a Xeon E5-2680@2.8 GHz, which is composed of 2 CPUs sharing their memory and each CPU has 10 cores.

8.1 ASTRID reactor

The first set of results shown here is dedicated to an assembly of the ASTRID reactor. ASTRID is a French design of Gen-IV bsodium cooled fast breeder reactor. A detailed description of the reactor design can be found in [52]. Aside from the technicalities, the ASTRID reactor most remarkable peculiarity directly related to our work is the pancaked core design. In few words, in order to reduce the positive sodium void coefficient, an axially heterogeneous core design is adopted. A sequence of fertile and fissile layers constitutes the reactor core. In case of loss of coolant, this should increase the neutron leakage towards a neutron absorber and decrease the reactivity. The reactor should feature a compact core of hexagonal fuel assemblies.

An internal assembly of this reactor has been chosen to validate the accuracy and performances of the SC, as anticipated in section 3.1.5. For a very heterogeneous design in the axial direction, the classic two steps calculations are not appropriate. For this reason, this study case was chosen to validate the accuracy of the Step method developed in [23]. For the same reason we have used this case for our first results set. Using the same assembly type allows us to present a method-to-method comparison, where we can optimize the axial meshes for both methods in order to obtain similar accuracies, and then compare the methods performances.

Three results are presented in this part, corresponding to three different axial heights and materials compositions, but with the same two dimensional layout. Figure 21 presents the reactor axial design and gives a simplified idea of the three computational domains that will be presented in the next pages. From the smallest to the largest, the three cases will be referred to as *small cyclic assembly*, *half column assembly* and *full column assembly*. The first two cases are used for the SC to polynomial comparisons, while the third is presented for the Monte Carlo reference comparison, since this case is the most physically challenging. Figure 22 shows eleven radial cuts of the assembly geometry as well as an axial view obtained with the Tripoli4® geometry visualizer. The radial cuts correspond to the

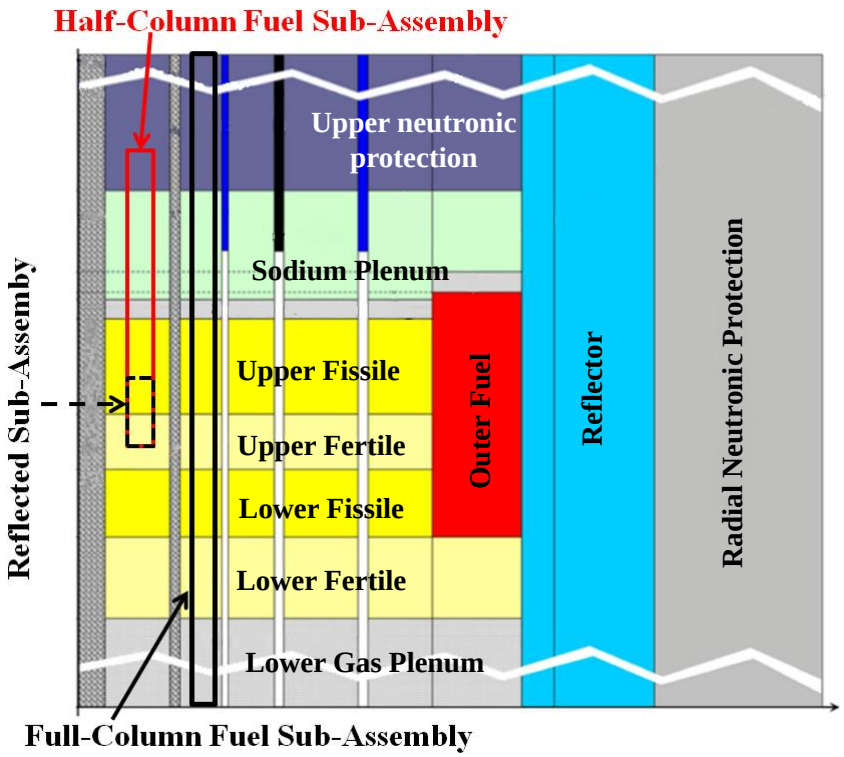


Figure 21 – ASTRID axial layout and simplified view of the three computational cases.

eleven different material planes composing the *full column assembly* case. The other two cases can be considered as sub-cases with fictive reflective boundary conditions on one or both axial sides.

8.2 Polynomial vs step comparison

The polynomial to step methods comparison is carried out for the two smaller computational cases shown in Fig.21, and for each of them the following parameters are compared: number of axial meshes, k-effective, computational time and memory usage. The quadrature formula, the radial and axial distances between trajectories and the convergence criteria are the same for the two methods. Unless differently specified, the set of parameters used in the ASTRID section is reported in Table 8.1.

8.2.1 Small cyclic assembly

The *small cyclic assembly*, or *reflected assemby* (as indicated in Fig.21), has been used in [23] as a first test to validate the results obtained using the Step method. In this case a reflective boundary condition is applied on every side and a cyclic tracking is used. Physically, this case of about 30 cm height is only representative of a small interface between a fertile and a fissile layer. However, since the two interface materials are very different, the axial flux gradients are quite important. As a consequence, a large number of axial meshes is needed using the Step method in order to obtain an accurate solution. Figure 23 shows the axial fluxes obtained using the Step method with 30 axial meshes and the reference Monte Carlo solution with the relative standard deviation. Using the Polynomial method, the same level of accuracy was obtained using only two axial meshes. Figure 24 gives a visual interpretation of the different meshes used for the Step and Polynomial

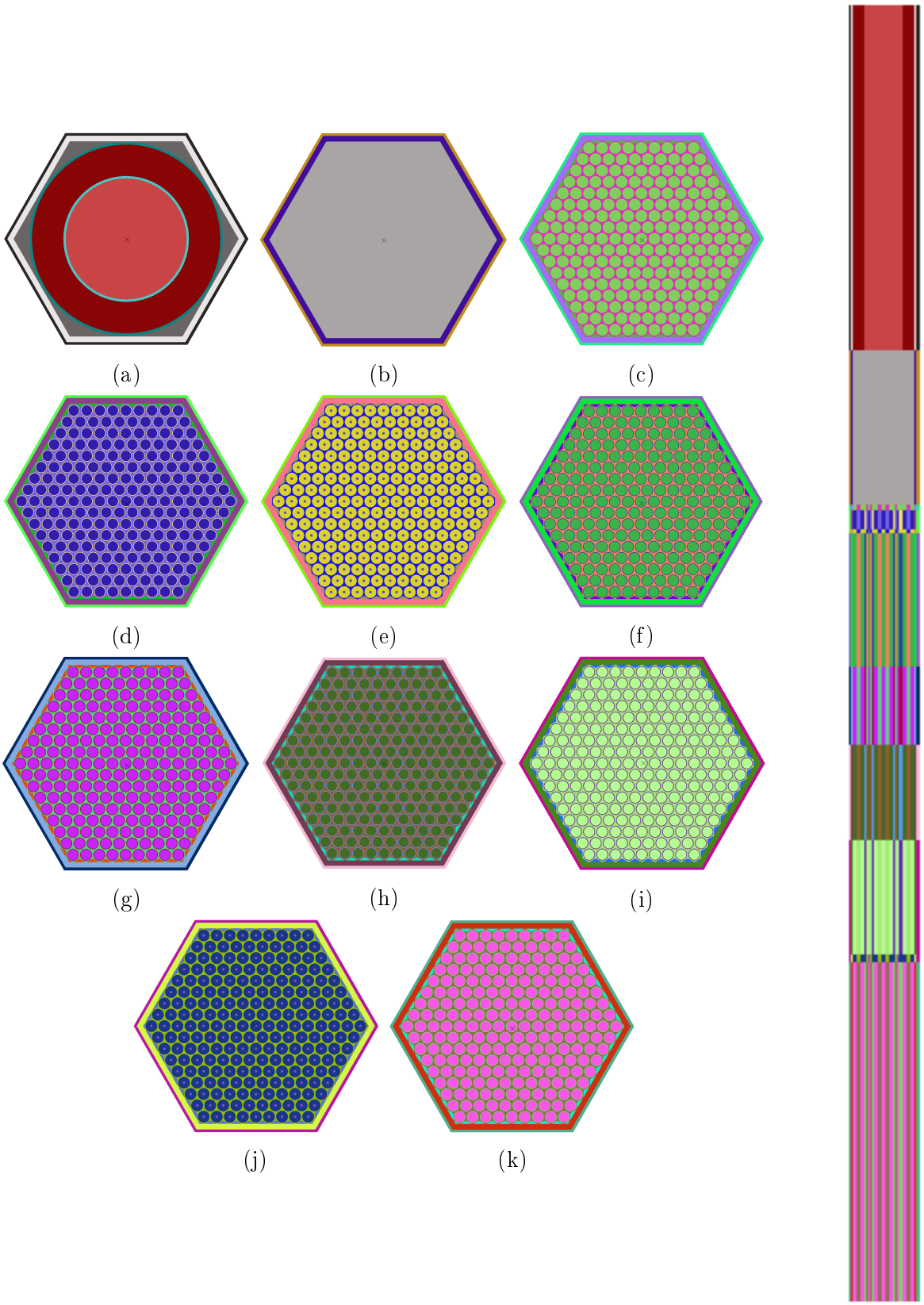


Figure 22 – Axial and radial view of the ASTRID *full column assembly* geometry visualized using the TRIPOLI4 graphic tool. The eleven radial cuts corresponding to different material layers are represented with different colors. a) Neutronic protection. b) Sodium plenum. c,d,e) Structure materials, springs and gas plenum. f) Fissile layer. g) Fertile layer. h) Fissile layer. i) Fertile layer. j) Structure materials. k) Gas plenum.

Computational parameters	
Quadrature formula	Gauss-Legendre
# Azimuthal angles $(0, \pi)$	24
# Polar angles $[0, \frac{\pi}{2})$	4
Δr (cm)	0.05
Δs (cm)	0.4
Anisotropy order	5
Energy groups	1200
Eigenvalue precision	$1e^{-5}$
Fission source precision	$1e^{-4}$

Table 8.1 – Standard parameters set used for all the ASTRID assembly calculations. Δr and Δs are integration parameters and Fig.9 gives a graphical representation of their meaning.

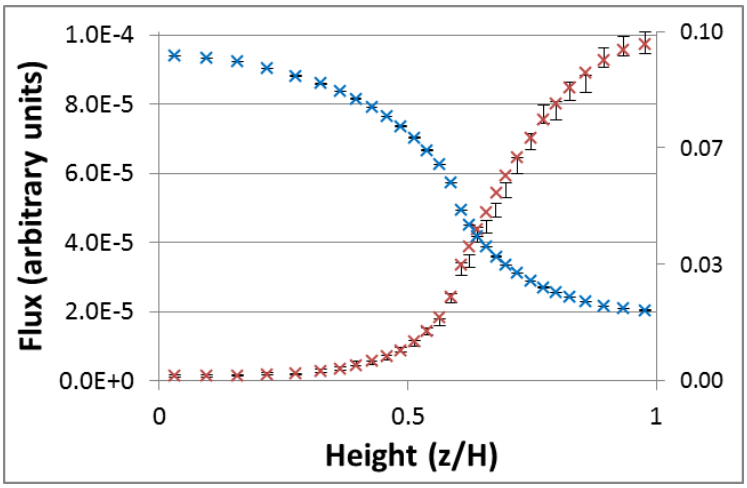


Figure 23 – Axial fluxes for a fuel pin for the *small cyclic assembly* computed using the Step method (crosses) and comparison with the reference calculation (black). The red and blue colors corresponds respectively to a thermal and a fast energy group and they are normalized using different factors. Image taken from [23].

methods, while Table 8.2 compares the performances of the Polynomial method against the SC solution. For both methods the computational time and memory usage are reported both with and without the use of the acceleration method. As we can see, if no acceleration is used the Polynomial method performances are slightly better, when compared with the Step method, but the larger improvement appears when activating the acceleration option for both methods.

8.2.2 Half column assembly

The *half column assembly* features five different material axial layers: a fertile zone, a fissile one, a gas plenum, a sodium plenum and a neutronic protection containing boron carbide, for a total height of about 1 meters. Once again, a simplified view of the assembly inside the reactor core is given in Fig.21. This second case is still not representative of the actual reactor axial gradients, since it uses a reduced neutron protection height and a fictive reflective boundary condition at the bottom. Even if not fully physically representative, it is still very interesting from the computational point of view. Since the materials are more

SMALL CYCLIC ASSEMBLY				
T4 k_{eff}	1.16103 ± 3 pcm			
	Step		**Polynomial ($\mathbf{N_p = 2}$)**	
k_{eff}	1.16052		1.16055	
δk_{eff} (pcm)	-44		-41	
# Axial meshes	30		2	
# Chords (M)	12.45		10.39	
	Free	**Acc.**	**Free**	**Acc.**
# Outer	10	5	10	5
# Inner	260 063	6 000	220 480	6 000
# Outer DP_N	-	14	-	12
Memory (Gb)	2.17	17.88	1.67	5.33
Time (s)	18 891	1 719	15 647	788

Table 8.2 – Comparison between SC and Polynomial method for the *small cyclic assembly*. M stands for million. The relative error in k_{eff} is computed against the reference Monte Carlo simulation obtained with Tripoli4.

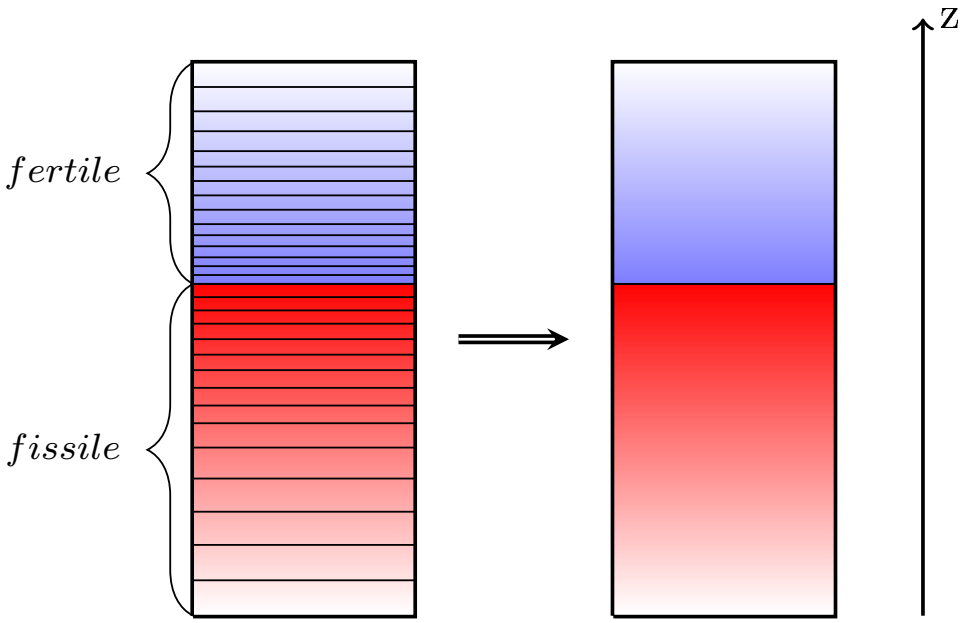


Figure 24 – Visual representation of the number of axial meshes used for the *small cyclic assembly* case for the Step (left) and the Polynomial method (right).

HALF COLUMN ASSEMBLY				
T4 k_{eff}	1.16318 ± 5 pcm			
	Step		**Polynomial ($\mathbf{N_p = 2}$)**	
k_{eff}	1.16262		1.16331	
δk_{eff} (pcm)	-48		+11	
# Axial meshes	110		6	
# Chords (M)	63.78		54.71	
	Free	**Acc.**	**Free**	**Acc.**
# Outer	12	6	11	5
# Inner	211 575	7 200	191 577	6 000
# Outer DP_N	-	15	-	14
Memory (Gb)	6	47	2.6	14
Time (h)	17.66	1.98	12.85	0.70

Table 8.3 – Comparison between SC and Polynomial method for the *half column assembly*. The relative error in k_{eff} is computed against the reference Monte Carlo simulation obtained with Tripoli4. M stands for million.

heterogeneous in comparison with the first case just presented, the axial flux gradients are much more severe. Figure 13b shows the axial fluxes for a fuel pin computed using the Step method. The different performances obtained using the Step and the Polynomial method for about the same level of accuracy in this case, are compared in Table 8.3. As the table shows, in order to reach a similar accuracy, the Step and Polynomial methods need 110 and 6 axial meshes, respectively. For the Polynomial method the sixth axial mesh has been added in the upper neutronic protections, which features very strong flux gradients. The important difference in the number of axial meshes necessary to represent the axial flux gradient translates again in an important difference both in computational time and in memory footprint. Table 8.4 is meant to show the convergence of the k_{eff} as a function of the number of axial meshes of both the Step and the Polynomial method. Figure 25 shows the axial fluxes for a fuel pin for the *half column assembly* using polynomials of order 1 and 2 and several axial discretizations. As we can see, increasing the polynomial order from linear to parabolic allows a considerable reduction in the number of axial meshes necessaries to obtain a proper representation of the axial flux gradients. This set of figures is also meant to give a graphical understanding of some of the results reported in table 8.4.

8.3 Physical Comparison with Tripoli4

The physical comparison of reaction rates has been carried out using the *full column assembly*. This case features all the eleven materials shown in Figure 22. The total axial height is of about 3.3 meters. However, the geometry used for the APOLLO3 calculation does not exactly coincide with the one used for the TRIPOLI4 simulation. This is mainly due to the fact that our solver treats only extruded geometries. As a consequence, the basic two dimensional geometry has to be obtained superposing all the two dimensional geometries of the different axial planes. The drawback of this approach is that superposing very different

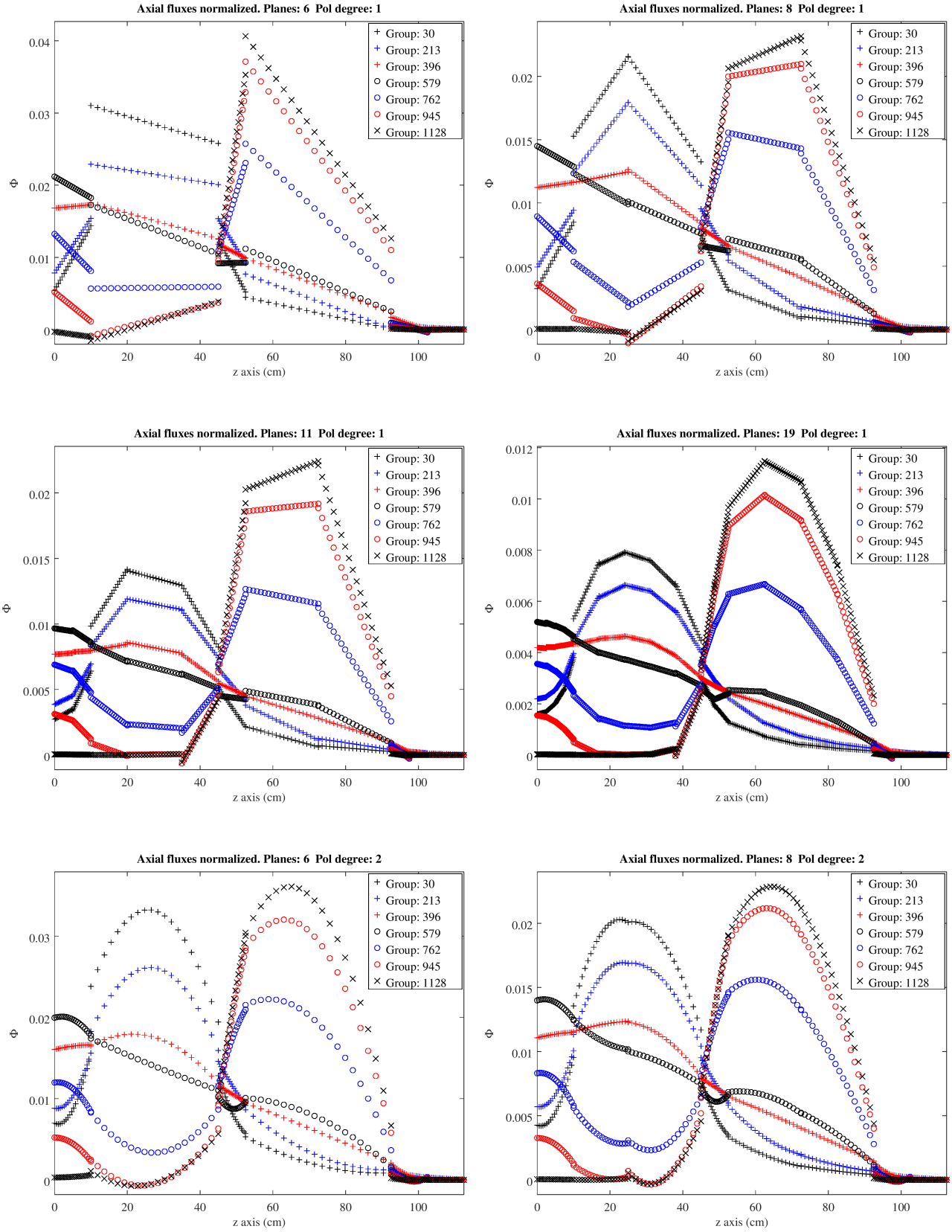


Figure 25 – Axial fluxes for a fuel pin in the *half column assembly* for different energy groups and for several axial discretizations and polynomial orders. Remark that using a linear flux approximation requires an important meshes refinement in order to properly approximate the axial flux gradients, while a polynomial of order 2 delivers good results already with 6 axial meshes.

HALF COLUMN ASSEMBLY AXIAL MESHES CONVERGENCE

Method	Step				Polynomial ($\mathbf{N_p = 1}$)					
# Axial meshes	57	110	180	257	5	6	7	8	11	19
δk_{eff} (pcm)	-164	-48	-19	+2	-1666	-1596	-192	-188	-61	+6

Method	Polynomial ($\mathbf{N_p = 2}$)					Polynomial ($\mathbf{N_p = 3}$)				
# Axial meshes	5	6	7	8	11	5	6	7	8	11
δk_{eff} (pcm)	-5	+11	+10	+13	+10	+28	+33	+11	+11	+10

Table 8.4 – Axial meshes convergence analysis for the SC and Polynomial method for the *half column assembly*. The relative error in k_{eff} is computed against the reference Monte Carlo simulation obtained with Tripoli4. M stands for million.

geometries will results in a heavily discretized basic two-dimensional geometry, which has to be imposed to every axial plane. As a consequence, a relatively homogeneous plane must be anyway discretized, resulting in a useless computational over-cost. A possible workaround to this problem can be obtained replacing homogeneous equivalent materials, if this is believed to have a low impact in the quality of the result.

For this computation, the detail level used for the basic two dimensional geometry has been chosen equivalent to the fuel layers discretization, and imposed to all the other layers. This entails that the sodium plenum (Fig.22b) is over-discretized, while the two small slices housing the set of springs holding the fuel pins (Figs.22e and 22j), were replaced with a homogeneous material with the size of the fuel pin, representative of the spring and the helium inside it. Finally, the neutronic protection (Fig.22a) was also replaced with a homogeneous material. Since it is a very strong absorber far away from the fuel, it is considered well represented by a homogeneous material, instead of paying the price of imposing an additional computational cost to all the other layers. Even if not completely physically representative of the problem, the same geometry has been used in the Monte Carlo simulation in order to have a fair code-to-code comparison. This kind of approach has been inspired by previous works [53].

Tables 8.5 and 8.6 show the information relative to several calculations performed on the *full column* assembly with different polynomial degrees and number of axial meshes. In these tables are displayed the reactivity error associated to each computation, as well as the number of inner and outer iterations, the computational time and the memory engaged. As we can see, using 11 axial meshes (corresponding to the 11 material layers) a polynomial of order 3 shows more accurate results, in comparison with the polynomial of order 2. However, adding some meshes both methods converge to similar values and the use of a polynomial of order 2 is less expensive both in terms of memory and in computational time.

Figures from 26 to 29 display the macroscopic fission reaction rates obtained with the Polynomial method and associated relative and absolute errors in comparison with the Monte Carlo reference calculations, for different axial planes, for the *full column* assembly in nominal condition, computed with a polynomial of order 2. The results are obtained integrating the reaction rates over the whole assembly radial cross section, in each different axial zone and collapsing the energy dependence over 33 energy groups. From these results we can see that large values of the relative errors on the fission rates correspond to very low reaction-rate absolute values, while when the fission reaction rate is important, the

FULL COLUMN ASSEMBLY NOMINAL						
T4 k_{eff}	**Nominal**: 1.12587 ± 3 pcm					
	Polynomial ($N_p = 2$)			**Polynomial ($N_p = 3$)**		
# 2D regions	115			115		
# Axial meshes	11	15	17	11	15	17
# Chords (M)	202.02	202.46	202.68	202.02	202.46	202.68
# Classes (M)	4.07	5.59	6.35	4.07	5.59	6.35
Classes/chords (%)	2.01	2.76	3.13	2.01	2.76	3.13
Classification rate (%)	98.84	98.41	98.19	98.84	98.41	98.19
δk_{eff} (pcm)	-88.9	+57.0	+56.7	-17.4	+58.4	+55.7
# Outer	6	6	5	9	6	6
# Inner	7 200	7 200	6 000	10 800	7 200	7 200
# Outer DP_N	30	35	30	46	39	36
Memory (Gb)	31	39	43	45	58	64
Time (h)	2.02	2.39	2.21	3.75	3.45	3.72

Table 8.5 – Results obtained with the Polynomial method for the *full column* assembly in nominal condition. M stands for million.

maximum relative errors are around ±1%. Figure 30 shows the neutronic capture in the upper protection. Here the maximum relative errors for non-negligible reaction rate values are around ±10%. As Figure 13b shows, in this particular zone the neutron flux decreases by several orders of magnitude and the axial flux behaviour cannot be represented correctly with a polynomial approximation.

Figures 31 through 34 report several axial profiles of the macroscopic fission reaction rate integrated over the whole energy domain and radial cross section, and associated relative error for different number of axial meshes and polynomial degrees, both for the nominal and the voided configurations. These figures show that the polynomial approximation is indeed well suited to approach the axial behaviour of the neutron flux, at least in the reactor core for the presented problem. Moreover, we can see that the relative errors corresponding to non-negligible reaction-rate values are mostly comprised in a ±2% error band. We can also see that the polynomial of order 3 is more successful in representing the particular fission rate behaviour of Figure 32.

FULL COLUMN ASSEMBLY VOIDED						
T4 k_{eff}	**Voided**: 1.09643 ± 4.5 pcm					
	Polynomial ($N_p = 2$)			**Polynomial ($N_p = 3$)**		
# 2D regions	115			115		
# Axial meshes	11	15	17	11	15	17
# Chords (M)	202.02	202.46	202.68	202.02	202.46	202.68
# Classes (M)	4.07	5.59	6.35	4.07	5.59	6.35
Classes/chords (%)	2.01	2.76	3.13	2.01	2.76	3.13
Classification rate (%)	98.84	98.41	98.19	98.84	98.41	98.19
δk_{eff} (pcm)	-323.14	+112.4	+109.4	-99.2	+109.1	+110.5
# Outer	6	6	6	7	6	6
# Inner	7 200	7 200	7 200	8 400	7 200	7 200
# Outer DP_N	29	28	31	43	33	33
Memory (Gb)	31	39	43	45	58	64
Time (h)	2.37	2.88	3.13	3.90	4.13	4.66

Table 8.6 – Results obtained with the Polynomial method for the *full column* assembly in voided condition. M stands for million.

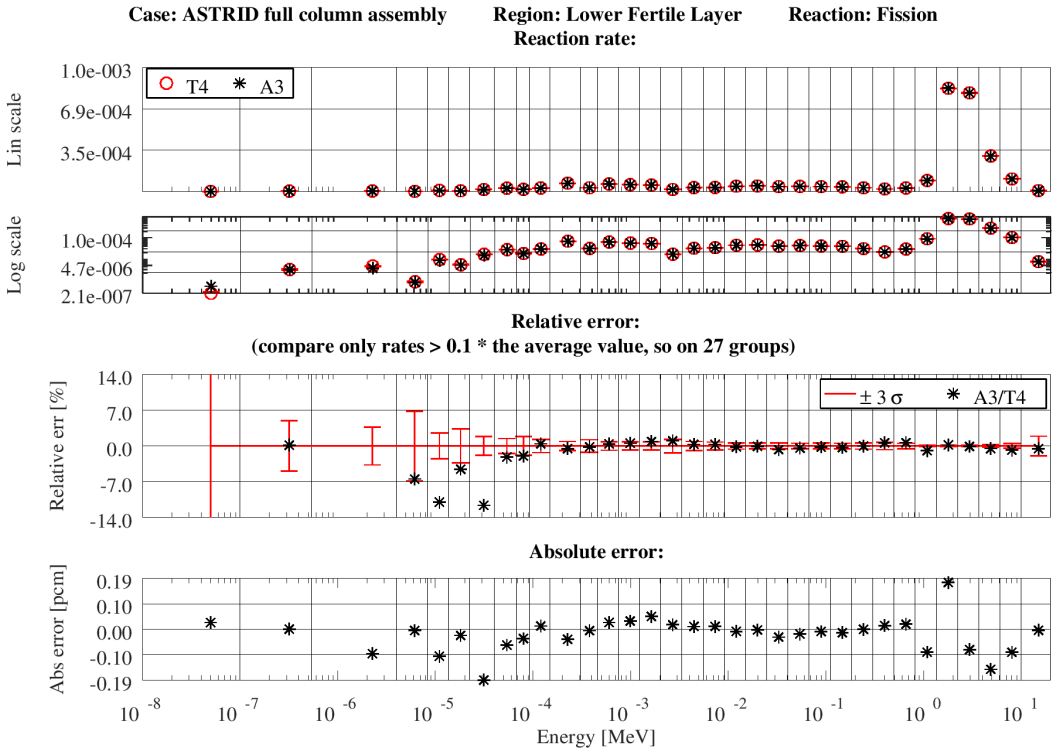


Figure 26 – Comparison between the Polynomial method and the reference Monte Carlo calculation for the *full column* assembly in nominal condition on the macroscopic fission reaction rate for the lower fertile layer.

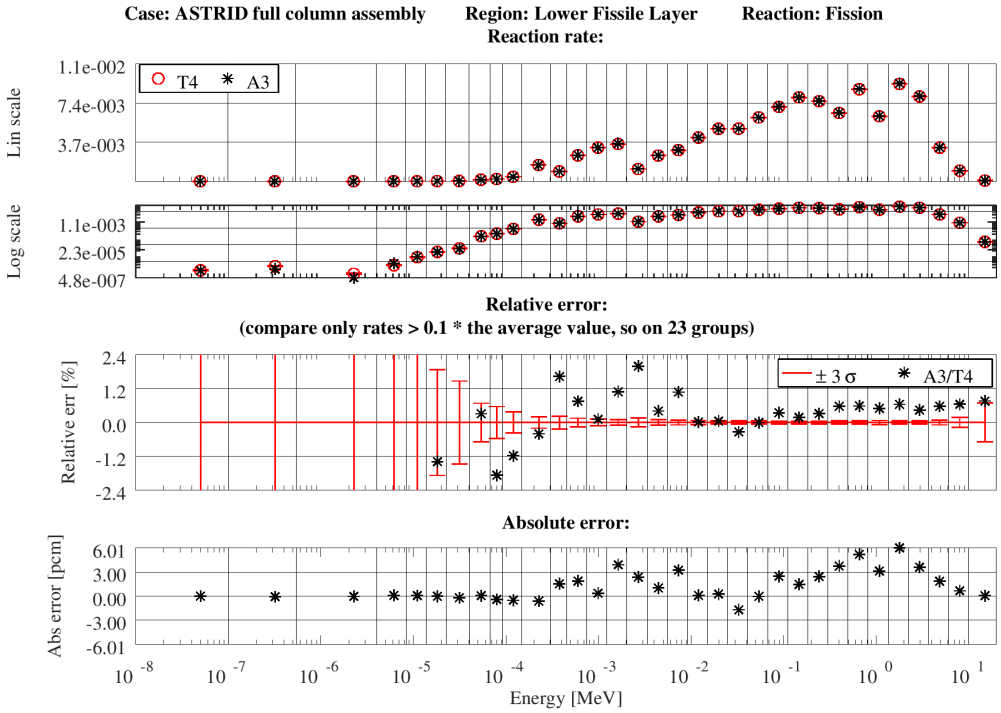


Figure 27 – Comparison between the Polynomial method and the reference Monte Carlo calculation for the *full column* assembly in nominal condition on the macroscopic fission reaction rate for the lower fissile layer.

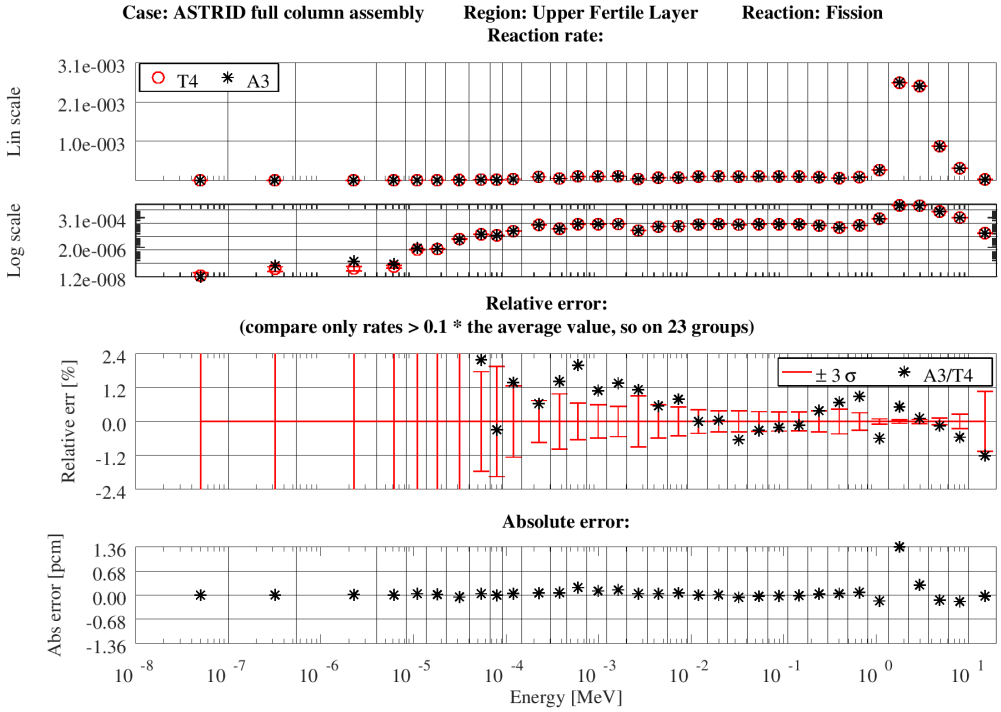


Figure 28 – Comparison between the Polynomial method and the reference Monte Carlo calculation for the *full column* assembly in nominal condition on the macroscopic fission reaction rate for the upper fertile layer.

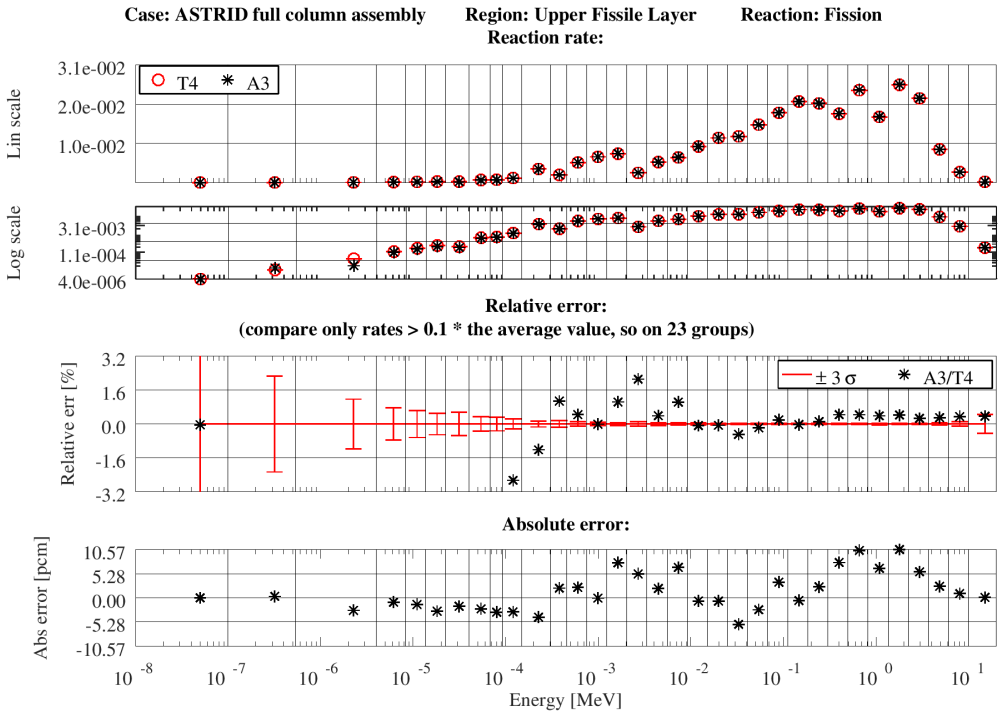


Figure 29 – Comparison between the Polynomial method and the reference Monte Carlo calculation for the *full column* assembly in nominal condition on the macroscopic fission reaction rate for the upper fissile layer.

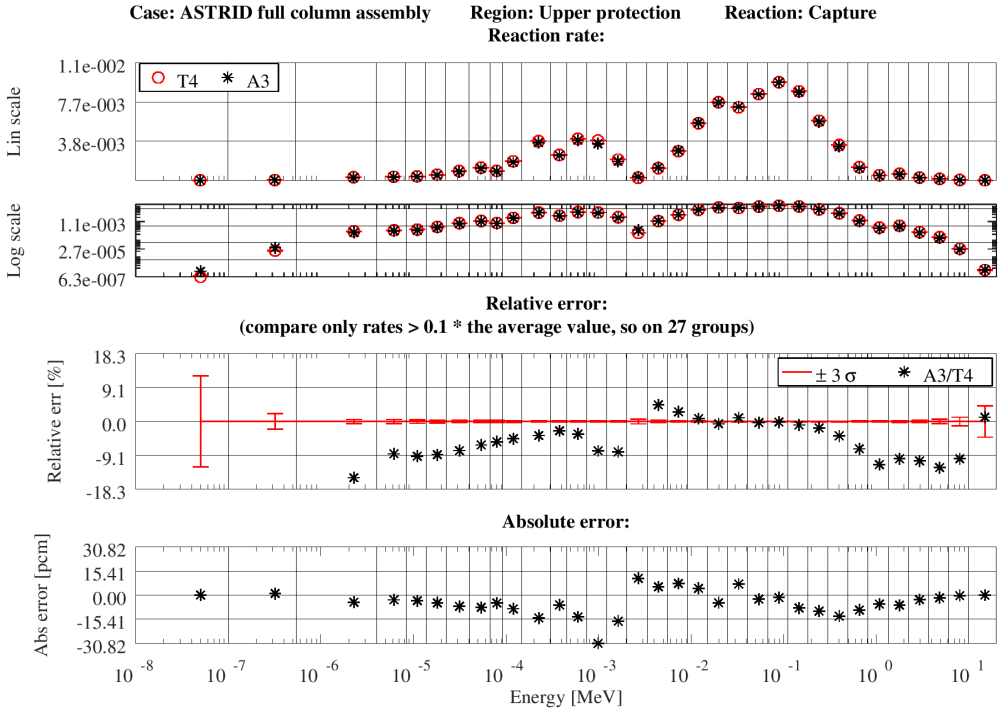


Figure 30 – Comparison between the Polynomial method and the reference Monte Carlo calculation for the *full column* assembly in nominal condition on the macroscopic capture reaction rate for the upper neutronic protection.

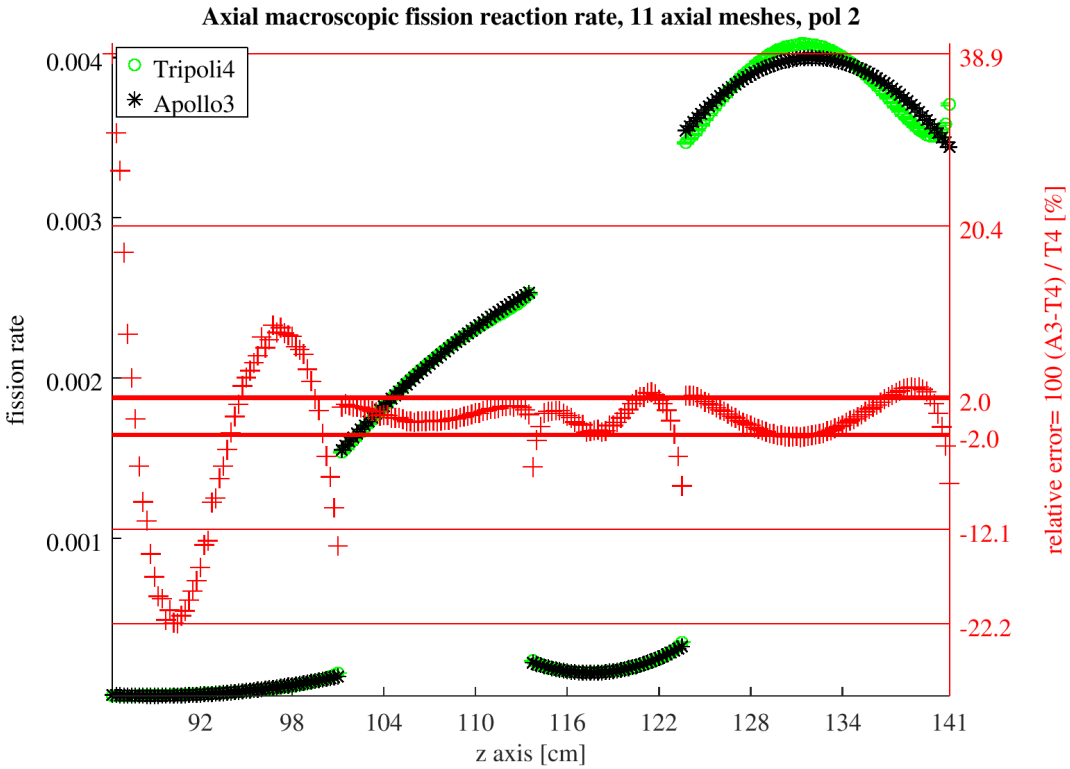


Figure 31 – Axial profile of the macroscopic fission rate and associated relative error for the *full column* ASTRID assembly in nominal conditions. The reference Monte Carlo calculation has been obtained scoring the fission rate on 220 axial meshes with constant height.

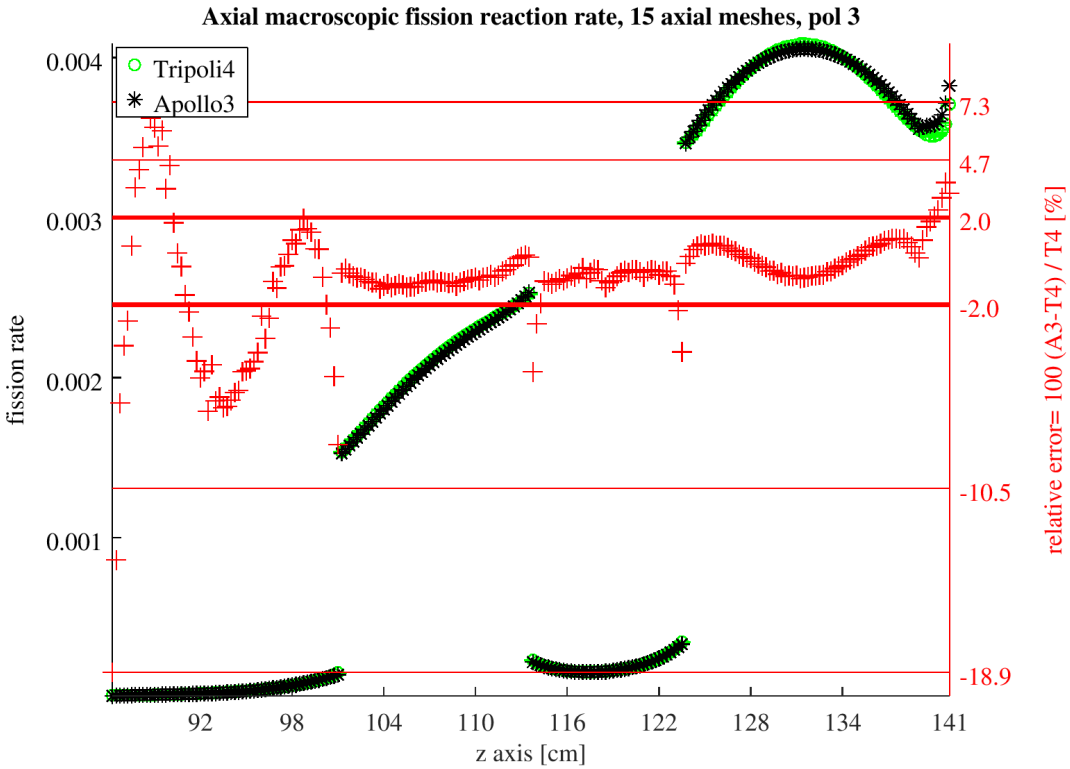


Figure 32 – Axial profile of the macroscopic fission rate and associated relative error for the *full column* ASTRID assembly in nominal conditions. The reference Monte Carlo calculation has been obtained scoring the fission rate on 220 axial meshes with constant heigl

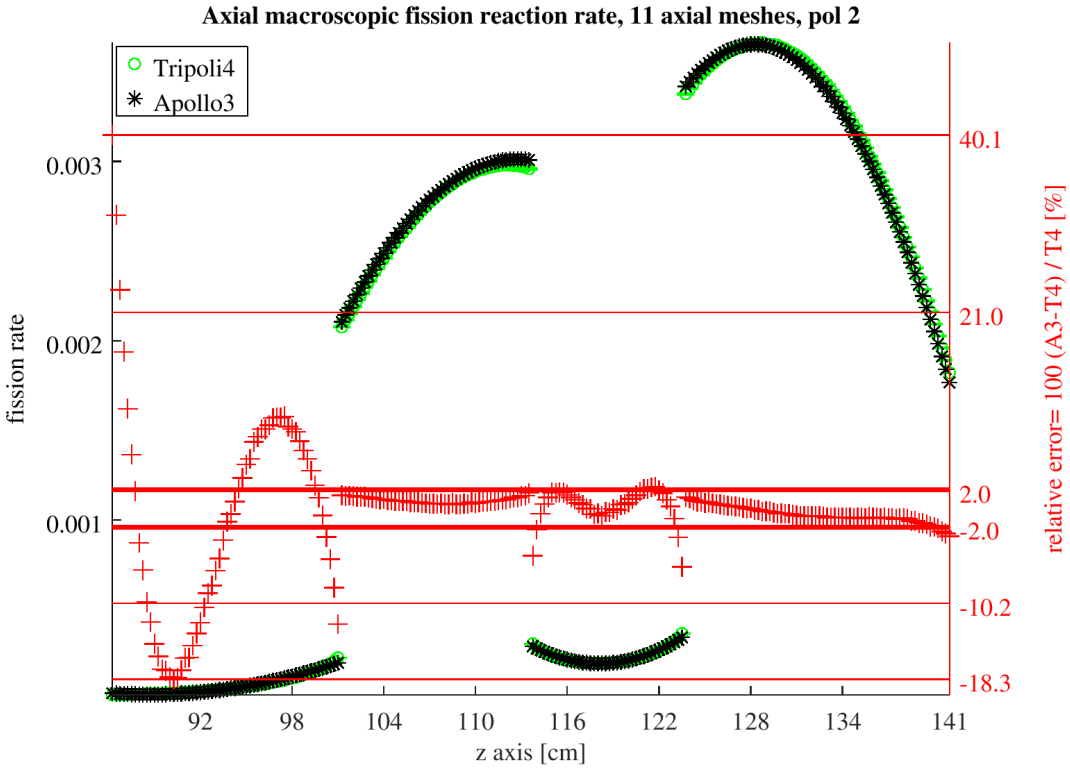


Figure 33 – Axial profile of the macroscopic fission rate and associated relative error for the *full column* ASTRID assembly in voided conditions. The reference Monte Carlo calculation has been obtained scoring the fission rate on 220 axial meshes with constant height.

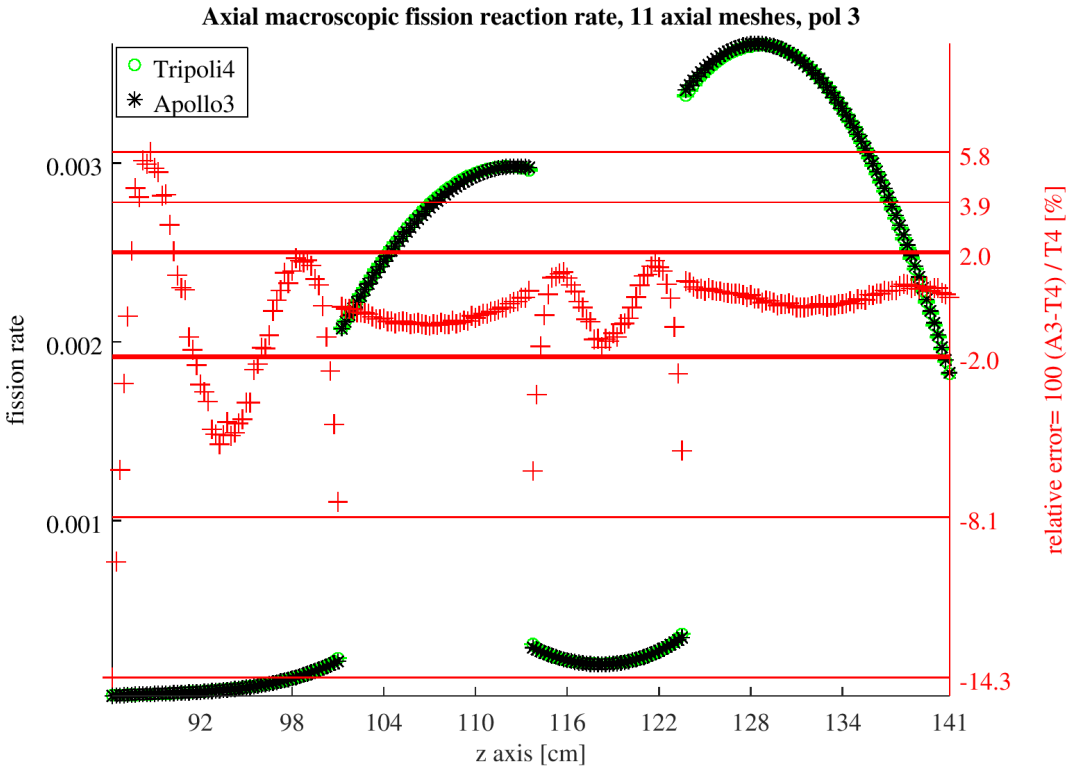


Figure 34 – Axial profile of the macroscopic fission rate and associated relative error for the *full column* ASTRID assembly in voided conditions. The reference Monte Carlo calculation has been obtained scoring the fission rate on 220 axial meshes with constant height.

8.4 Results obtained with the non-linear fitting method

This section is meant to show the performances of the fitting method that we have introduced in order to reduce the total memory footprint of the DP_N synthetic acceleration matrices used in the polynomial method. We compare the computational times and memory requirements with and without the non-linear fitting method on two cases that we have presented previously in table 8.5: the *full column* assembly in nominal condition with 15 axial meshes using a polynomial of order 2 and 3.

Table 8.7 shows the memory required by the different acceleration coefficients for the considered case, both for a polynomial of order 2 and 3. It also displays the percentage of memory occupied by the acceleration matrices, over the total memory needed for the calculation. As we can see, the acceleration coefficients constitute the largest portion of the total memory footprint. The size of the acceleration matrices depends linearly on the number of energy groups, so this proportion can suffer important changes depending on the energy discretization used. These calculations are run with 1200 groups, so a very large value if compared to standard thermal reactor calculations.

Table 8.8 shows the accuracy of the fitting method for these cases. For each level of accuracy the percentage indicates the amount of coefficients that are correctly represented by the model. Moreover, we have highlighted the area corresponding to the precision we considered sufficient. In practice, we observed that it is not possible to increase the accepted relative error above the level of $\sim 1 \times 10^{-3}$, because the errors introduced in the DP_N matrices cause the acceleration method to be unstable. This table also shows that the method is more successful in representing the *transmission* and *escape* matrices, in comparison with the *collision* and *incoming* matrices. The fitting function that we have chosen, is probably less adapted to approximate the $^{\dagger}\mathcal{C}$ and $^{\dagger}\mathcal{J}$ matrices, for which the choice of a combination

Coefficients memory [Gb]		
	Polynomial 2	**Polynomial 3**
$^{t}\mathcal{T}$	19.03	31.42
$^{t}\mathcal{E}$	4.18	7.07
$^{t}\mathcal{C}$	1.19	2.12
$^{t}\mathcal{J}$	4.18	7.07
TOT acc.	28.58	47.68
TOT acc./TOT [%]	73.3	82.2

Table 8.7 – DP_N matrices memory size for the *full column* ASTRID assembly in nominal conditions with 15 axial planes.

ϵ	**Fitted within ϵ [%]**			
	$^{t}\mathcal{T}$, $^{t}\mathcal{E}$		$^{t}\mathcal{T}$, $^{t}\mathcal{E}$, $^{t}\mathcal{C}$, $^{t}\mathcal{J}$	
	Polynomial 2	**Polynomial 3**	**Polynomial 2**	**Polynomial 3**
$< 1 \times 10^{-1}$	99.9998	99.9998	99.9680	99.9960
$< 1 \times 10^{-2}$	99.9712	99.9785	99.8087	99.9170
$< 1 \times 10^{-3}$	99.1392	99.3183	98.1376	98.4265
$< 5 \times 10^{-4}$	98.0883	98.4605	96.4749	96.9259
$< 1 \times 10^{-4}$	93.0796	93.2755	88.5373	89.9042

Table 8.8 – Fitting precision for the *full column* ASTRID assembly in nominal conditions with 15 axial planes. Results obtained when the fitting method is applied only to the $^{t}\mathcal{T}$ and $^{t}\mathcal{E}$ matrices (left) or to all the acceleration matrices (right). The highlighted area displays the precision we have chosen and the corresponding fitted percentage.

of exponential and rational power functions is in fact less justifiable in comparison to the $^{t}\mathcal{T}$ and $^{t}\mathcal{E}$ coefficients.

The results regarding the impact of this method on the total memory used during the calculation and the computational time are reported in Table 8.9. As we can see, the regression adds computational time for both the coefficients construction, and the total iteration time, since at the beginning of each group the coefficients must be re-evaluated. Moreover, since the regression formula we have chosen contains exponential and rational power functions, the associated computational cost is more important in comparison with regular floating-point operations.

	Polynomial 2		Polynomial 3	
	Standard	**Least squares**	**Standard**	**Least squares**
Accepted relative error ϵ	-	1×10^{-3}	-	1×10^{-3}
DP_N coeff. mem. (GB)	28.58	-	47.68	-
Fit coeff. mem. (GB)	-	0.26	-	0.45
Unfitted DP_N coeff. mem. (GB)	-	1.06	-	1.50
Build time (s)	752	1 882	2 509	4 182
Re-construct time (s)	-	563	-	881
Total memory (GB)	**39**	**18**	**58**	**20**
Total time overhead [%]	-	$\sim$ **+12**	-	$\sim$ **+13**
Total memory reduction [%]	-	$\sim$ **−54**	-	$\sim$ **−65**

Table 8.9 – Memory footprint and computational times of DP_N operator and relative impact on the whole $MOC + DP_N$ scheme when applying the fitting method to all the acceleration matrices.

9. Conclusions and Perspectives

The objective of the present work is the introduction of a polynomial approximation to represent the axial spatial dependence of the neutron angular flux in the TDT code. The method of long characteristics is used in TDT to solve the multi-group neutron transport equation for two-dimensional unstructured and three-dimensional extruded geometries. The method extension from two-dimensional to three-dimensional geometries has been realized during a previous PhD work [23]. In this context the classic SC approximation has been used to approximate the spatial dependence of the neutron angular flux. This approximation is acceptable only if the size of the computational meshes is small, when compared to the macroscopic flux gradients. Small axial meshes generally imply a high number of unknowns, which also means a high memory usage and computational time. The purpose of a polynomial approximation is to represent the same solution, but with a lower number of axial meshes.

This work is focused on the development of a polynomial approximation in the axial direction, while the step approximation has been conserved on the radial plane. A polynomial basis has been chosen to represent the axial variation of the angular fluxes. A modified version of the integral transport equation has been obtained using this polynomial representation and the solution of such equation has been implemented. Successively, the acceleration issue has been addressed. The DP_N synthetic acceleration, which was already implemented for the two-dimensional and three-dimensional methods, has been modified to be able to accelerate all the polynomial moments.

The polynomial method we have implemented is able to represent the solutions on a set of test cases with a similar level of accuracy, when compared with the SC method, but using a lower number of axial meshes. The considerable reduction in number of axial meshes resulted in a substantial decrease in computational time and memory requirements.

Even if the polynomial method proved to be efficient, it revealed also to be less robust than the Step method. We encountered several difficulties that had to be circumvented in order to be able to obtain a stable method. A first important issue regarding particle conservation has been addressed, after a careful reading of [44], by numerically computing a set of matrices in order to obtain coherency between the balance and the transmission equation. Issues related to poor angular and/or spatial numerical integration arose all along during this work, both for the transport and the acceleration part. A series of precautions had to be adopted in order to prevent from numerical ill-conditioning to cause the polynomial method to be unstable. Far from being perfect, the method proved however to be useful and be better performing than the Step equivalent.

It is clear that a direct full reactor three-dimensional solution of the neutron transport equation with the method of characteristics remains extremely expensive to consider this work an important improvement. However, for particular applications or reference calculations, the polynomial MOC could reveal to be interesting, since it is able to deliver quite accurate results.

In order to be useful, the polynomial MOC must be able to treat depletion calculations. At the present time, the method assumes to have constant cross sections in each axial layer. Even if not penalizing at zero burn-up, this approximation prevents the method to be efficient for depletion calculation. If the neutron flux is correctly represented by a polynomial function, also the macroscopic cross sections will inherit such behaviour as a consequence of isotope transmutation. At the present time, the only way to run this calculation is to increase the number of axial meshes when the error introduced by assuming a constant cross section in a given axial layer becomes too large. This approach is of course not optimal since it does not correctly represent the cross section spatial depletion. Moreover, this would increase the number of axial meshes used. Such computational overcost is not considered acceptable, especially for a method that is already expensive.

The elegant solution to this problem, which constitutes also one of the main perspective of the present work, is to represent the cross sections with the same polynomial approximation used to represent the angular fluxes. A polynomial representation would allow a fine representation of the spatial behaviour of the macroscopic cross sections during the isotopes evolution. Using such approximation, this method would be able to deliver very interesting reference calculations for three-dimensional assemblies or clusters. Moreover, in order to be able to apply this method to larger cases, a distributed memory parallel scheme could be implemented.

Part IV

Annexes

A. Sanchez balance formula for particles conservation

The angular balance equation obtained in Section 5.2 is conservative. In a first phase of our work Eq.(5.16) has been used as a balance equation, but found to be unstable under certain space integration circumstances. The transition from Eq.(5.16) to (5.22) is justified by the following demonstration, which is a more explicit formulation of what was synthetically proven in [44].

We need to compute the polynomial moments of the angular flux, given by Eq.(5.13):

$$'\vec{\psi}_r(\vec{\Omega}) = \frac{1}{V_r}\int_r d\vec{r}\ \vec{P}(\tilde{z}_r)\,\psi(\vec{r},\vec{\Omega}),$$

which can be written coherently with the trajectory-based discretization, giving:

$$'\vec{\psi}_r(\vec{\Omega}) = \frac{1}{V_r}\int_{\partial r} d_2 r_\perp \int_0^l dt\ \vec{P}\left[\tilde{z}_r(t)\right]\ \psi\left[\vec{r}(t),\vec{\Omega}\right].$$

Focusing in a first time only on the integral along the line and dropping the angular dependency for simplicity, we can express the spatial dependency of $P(\tilde{z}_r)$ and $\psi(\vec{r},\vec{\Omega})$ only as a function of t, the local coordinate along the trajectory. Thanks to Eq.(5.19) we get:

$$\tilde{z}_r = \frac{z_r^{in} - \bar{z}_r + \mu\ t}{\Delta z_r/2} \qquad\qquad \mu = cos(\theta) \qquad t \in [0,l]$$

and the transmission equation (5.33) allows us to express the angular flux as a function of t:

$$\psi(t) = \psi(0)\,e^{-\Sigma_r t} + \int_0^t dt'\ \vec{P}(t')\cdot\vec{q}_r\ e^{-\Sigma_r(t-t')}$$

where we have replaced $\vec{P}\left[\tilde{z}_r(t')\right]$ with $\vec{P}(t')$. We can obtain the polynomial moments of the angular flux along the line by computing:

$$\int_0^l dt\ \vec{P}(t)\,\psi(t) = \int_0^l dt\ \vec{P}(t)\,\psi(0)\ e^{-\Sigma_r t} + \int_0^l dt\ \vec{P}(t)\int_0^t dt'\ \vec{P}(t')\cdot\vec{q}_r\ e^{-\Sigma_r(t-t')}$$

Going through the two terms separately, we integrate by part the first one:

$$\int_0^l dt\ \vec{P}(t)\,\psi(0)\,e^{-\Sigma_r t} = \frac{1}{\Sigma}\left[\psi(0)\,\vec{P}(0) - \psi(0)\,e^{-\Sigma_r l}\,\vec{P}(l) + \psi(0)\int_0^l dt\ \frac{d\vec{P}(t)}{dt}e^{-\Sigma_r t}\right]$$

Then we invert the two integrals of the second term and we integrate by parts this term too:

$$\int_0^l dt\ \vec{P}(t) \int_0^t dt'\ \vec{P}(t') \cdot \vec{q}_r\ e^{-\Sigma_r (t-t')} =$$

$$\vec{q}_r \cdot \int_0^l dt'\ \vec{P}(t')\, e^{\Sigma_r\, t'} \int_{t'}^l dt\ \vec{P}(t)\, e^{-\Sigma_r\, t} =$$

$$\vec{q}_r \cdot \int_0^l dt'\ \vec{P}(t')\, e^{\Sigma_r\, t'} \left[\frac{1}{\Sigma_r} \left(\vec{P}(t')\, e^{-\Sigma_r\, t'} - \vec{P}(l)\, e^{-\Sigma_r\, l} + \int_{t'}^l dt\ \frac{d\vec{P}(t)}{dt}\, e^{-\Sigma_r\, t} \right) \right] =$$

$$\frac{1}{\Sigma_r} \left[\int_0^l dt'\ \vec{P}(t') \left(\vec{P}(t') \cdot \vec{q}_r \right) - \vec{P}(l) \int_0^l dt'\, \vec{P}(t')\, e^{-\Sigma (l-t')} \cdot \vec{q}_r + \right.$$

$$\left. + \vec{q}_r \cdot \int_0^l dt'\ \vec{P}(t') \int_{t'}^l dt\ \frac{d\vec{P}(t)}{dt}\, e^{-\Sigma (t-t')} \right].$$

Putting all back together, switching again the order of the two integrals and regrouping some terms, we get:

$$\Sigma_r \int_0^l dt\, \vec{P}(t)\, \psi(t) =$$

$$\int_0^l dt'\ \vec{P}(t') \left(\vec{P}(t') \cdot \vec{q}_r \right) + \vec{P}(0)\, \psi(0) - \vec{P}(l) \left(\psi(0)\, e^{-\Sigma_r\, l} + \int_0^l dt'\ \vec{P}(t')\, e^{-\Sigma_r (l-t')} \cdot \vec{q}_r \right)$$

$$+ \int_0^l dt\ \frac{d\vec{P}(t)}{dt} \left(\psi(0)\, e^{-\Sigma_r\, t} + \int_0^t dt'\, \vec{P}(t')\, e^{-\Sigma_r (t-t')} \cdot \vec{q}_r \right).$$

Recognizing the nature of the two terms between the parentheses we eventually get:

$$\Sigma_r \int_0^l dt\, \vec{P}(t)\, \psi(t) = \int_0^l dt'\ \vec{P}(t') \left(\vec{P}(t') \cdot \vec{q}_r \right) + \vec{P}(0)\, \psi(0) - \vec{P}(l)\, \psi(l) + \int_0^l dt\ \frac{d\vec{P}(t)}{dt}\, \psi(t).$$

Completing the integration over $d_2 r_\perp$ we get the same balance equation as in Eq.(5.22), which proves that the balance obtained is section 5.2 is conservative only if the matrix $\bar{\bar{\mathcal{P}}}(\vec{\Omega})$ is computed numerically, using the same trajectory discretization used for the transport sweep. The numerical matrix reported in Eq.(5.20) coincides, in fact, with the following term of the previous equation, integrated over $d_2 r_\perp$:

$$\int_{\partial r_\perp} d_2 r_\perp \int_0^l dt'\ \vec{P}(t') \otimes \vec{P}(t') = V_r(\vec{\Omega})\, \bar{\bar{\mathcal{P}}}_r(\vec{\Omega}).$$

B. Spherical harmonics relation

The relation of the complex spherical harmonics Y_l^m used in Section 5.3 is briefly recalled here. The full treatment can be found in [54]. The relation that came in use in our balance problem reads:

$$\mu\ Y_l^m(\theta,\varphi) = \gamma_{l+1}^m\ Y_{l+1}^m(\theta,\varphi) + \varepsilon_{l-1}^m\ Y_{l-1}^m(\theta,\varphi)$$

where:

$$\gamma_{l+1}^m = \sqrt{\frac{(l+m+1)(l-m+1)}{(2l+1)(2l+3)}} \qquad \varepsilon_{l-1}^m = \begin{cases} \sqrt{\frac{(l+m)(l-m)}{(2l+1)(2l-1)}} & if \quad |m| \leq l-1 \\ 0 & if \quad |m| > l-1 \end{cases}$$

that for the real spherical harmonics A_l^m reads:

$$\mu\ A_l^m(\theta,\varphi) = \alpha_{l+1}^m\ A_{l+1}^m(\theta,\varphi) + \beta_{l-1}^m\ A_{l-1}^m(\theta,\varphi) \tag{B.1}$$

where:

$$\alpha_{l+1}^m = \frac{\sqrt{2l+3}}{\sqrt{2l+1}}\ \gamma_{l+1}^m \qquad \beta_{l-1}^m = \begin{cases} \frac{\sqrt{2l-1}}{\sqrt{2l+1}}\ \varepsilon_{l-1}^m & if \quad |m| \leq l-1 \\ 0 & if \quad |m| > l-1 \end{cases}$$

$$A_l^m(\theta,\varphi) = \begin{cases} \sqrt{\frac{4\pi}{2l+1}}\ Y_l^0(\theta,\varphi) & m = 0 \\ \sqrt{\frac{4\pi}{2l+1}}\left(Y_l^m(\theta,\varphi) + \bar{Y}_l^m(\theta,\varphi)\right) & m > 0 \\ -i\sqrt{\frac{4\pi}{2l+1}}\left(Y_l^m(\theta,\varphi) - \bar{Y}_l^m(\theta,\varphi)\right) & m < 0 \end{cases}$$

where " $^-$ " denotes complex conjugation.

Expressing Eq.(B.1) in a matrix form we get:

$$\mu\ \vec{A}(\vec{\Omega}) = \bar{\bar{\alpha}}\ \vec{A}(\vec{\Omega}), \tag{B.2}$$

where $\bar{\bar{\alpha}}$ is a matrix with dimension $N_m \times N_m$. Each line of the matrix has just two non-zero elements, which are the coefficients α_{l+1}^m and β_{l-1}^m.

As written in Sec.5.3, the balance obtained via Eq.(5.31) requires a number of additional angular moments because of the spherical harmonics relation of Eq.(B.2). To better understand the exact number and type of additional harmonics needed, we treat a simple example by assuming *K=1* and $N_p = 2$, that is to say, for linear anisotropic scattering and a parabolic spatial expansion. Inserting Eq.(B.2) into the balance equation (Eq.(5.27)) we obtain that this latter equation must be expanded in function of the anisotropy order and of the maximum polynomial degree. To be more explicit we start from the highest polynomial order and we use here the double index notation for the spherical harmonics sum, as in

Eq.(2.9). The moment of the angular flux corresponding to a polynomial order of p and the spherical harmonics $A_k^l(\vec{\Omega})$ will be written as ${}'\Phi_{p,k}^{\ l}$. We do not write the complete balance equation, but just the dependence between different terms:

$$p=2 \rightarrow \begin{cases} {}'\Phi_{2,0}^{0} \rightarrow {}'\Phi_{1,1}^{0} \\ \\ {}'\Phi_{2,1}^{-1} \rightarrow {}'\mathbf{\Phi_{1,2}^{-1}} \\ {}'\Phi_{2,1}^{0} \rightarrow {}'\mathbf{\Phi_{1,2}^{0}} + {}'\Phi_{1,0}^{0} \\ {}'\Phi_{2,1}^{1} \rightarrow {}'\mathbf{\Phi_{1,2}^{1}} \end{cases} \tag{B.3}$$

$$p=1 \rightarrow \begin{cases} {}'\Phi_{1,0}^{0} \rightarrow {}'\Phi_{0,1}^{0} \\ \\ {}'\Phi_{1,1}^{-1} \rightarrow {}'\mathbf{\Phi_{0,2}^{-1}} \\ {}'\Phi_{1,1}^{0} \rightarrow {}'\mathbf{\Phi_{0,2}^{0}} + {}'\Phi_{0,0}^{0} \\ {}'\Phi_{1,1}^{1} \rightarrow {}'\mathbf{\Phi_{0,2}^{1}} \\ \\ {}'\mathbf{\Phi_{1,2}^{-1}} \rightarrow {}'\mathbf{\Phi_{0,3}^{-1}} + {}'\Phi_{0,1}^{-1} \\ {}'\mathbf{\Phi_{1,2}^{0}} \rightarrow {}'\mathbf{\Phi_{0,3}^{0}} + {}'\Phi_{0,1}^{0} \\ {}'\mathbf{\Phi_{1,2}^{1}} \rightarrow {}'\mathbf{\Phi_{0,3}^{1}} + {}'\Phi_{0,1}^{1} \end{cases} \qquad p=0 \rightarrow \begin{cases} {}'\Phi_{0,0}^{0} \\ \\ {}'\Phi_{0,1}^{-1} \\ {}'\Phi_{0,1}^{0} \\ {}'\Phi_{0,1}^{1} \\ \\ {}'\mathbf{\Phi_{0,2}^{-1}} \\ {}'\mathbf{\Phi_{0,2}^{0}} \\ {}'\mathbf{\Phi_{0,2}^{1}} \\ \\ {}'\mathbf{\Phi_{0,3}^{-1}} \\ {}'\mathbf{\Phi_{0,3}^{0}} \\ {}'\mathbf{\Phi_{0,3}^{1}} \end{cases}$$

Where the bold font indicates the additional moments to be computed for each polynomial order. From this we can retrieve a general rule to identify the total dimension of the $\mathfrak{Z}$ matrix of Eq.(5.26). The number of angular moments related to the scattering operator are:

$$N_m = (K+1)^2,$$

which means that, if no coupling was present, the $\mathfrak{Z}$ would have dimensions $[\mathbf{N}^{sc} \times \mathbf{N}^{sc}]$, where:

$$\mathbf{N}^{sc} = N_m \times (N_p+1) = (K+1)^2 \times (N_p+1).$$

Using Eqs.(B.3) and Tab.B.1 we see that the $\mathfrak{Z}_D$ matrix that is actually used in Eq.(5.31) has a dimension $[\mathbf{N}^{tot} \times \mathbf{N}^{tot}]$, where:

$$\mathbf{N}^{tot} = \sum_{p=0}^{N_p} \left[(K+1)^2 + (2K+1) \times (N_p - p)\right] =$$

$$\mathbf{N}^{sc} + (2K+1) \times \sum_{p=0}^{N_p} (N_p - p) = \mathbf{N}^{sc} + (2K+1) \times \frac{N_p(N_p+1)}{2},$$

whereas if the same amount of memory had to be devoted to every polynomial order, under the *tensorial* hypothesis to use the maximum sub-matrix size, we would have had:

$$\mathbf{N}^{tens} = (K+1+N_p)^2 \times (N_p+1).$$

For the example under consideration ($K=1$ and $N_p=2$) we would have $\mathbf{N}^{tot}=25$ and $\mathbf{N}^{tens}=48$. Notice that the choice adopted so far in both the transport and the DP_N sections is to use the simpler but more memory demanding *tensorial* workaround.

		P=0, K=1		
		m		
		-1	0	1
l	0		$'\Phi_{0,0}^{0}$	
	1	$'\Phi_{0,1}^{-1}$	$'\Phi_{0,1}^{0}$	$'\Phi_{0,1}^{1}$

		P=1, K=1		
		m		
		-1	0	1
l	0		$'\Phi_{1,0}^{0}$	
	1	$'\Phi_{1,1}^{-1}$	$'\Phi_{1,1}^{0}$	$'\Phi_{1,1}^{1}$
	2	$\boldsymbol{'\Phi_{1,2}^{-1}}$	$\boldsymbol{'\Phi_{1,1}^{0}}$	$\boldsymbol{'\Phi_{1,1}^{1}}$

		P=2, K=1		
		m		
		-1	0	1
l	0		$'\Phi_{2,0}^{0}$	
	1	$'\Phi_{2,1}^{-1}$	$'\Phi_{2,1}^{0}$	$'\Phi_{2,1}^{1}$
	2	$\boldsymbol{'\Phi_{2,2}^{-1}}$	$\boldsymbol{'\Phi_{2,1}^{0}}$	$\boldsymbol{'\Phi_{2,1}^{1}}$
	3	$\boldsymbol{'\Phi_{2,3}^{-1}}$	$\boldsymbol{'\Phi_{2,3}^{0}}$	$\boldsymbol{'\Phi_{2,3}^{1}}$

Table B.1 – Dimension comparisons between different flux moment requirements in MOC polynomial calculations. Here a linear scattering is taken into consideration, while varying the dimension of the polynomial base. In the polynomial and angular flux moments $\Phi_{p,l}^{m}$ (l, m) are angular moment index, while p is the polynomial order. For each sub-table the capital letters (P and K) stand for the polynomial and scattering orders. The sequence of sub-tables has to be interpreted cumulatively. This means that when considering only the constant moment case (P=0) only the first sub-table has to be considered. For the linear polynomial case, the moment of the second sub-table has to be added to the first, and so on.

C. Numerical issues and heuristic solutions

This appendix presents the most important numerical problems that we encountered during this work implementation, as well as the solutions we adopted in order to circumvent them. Both the transport and the acceleration MOC equations suffered from numerical instabilities due to optically thin media or poor numerical discretization. The quality of the numerical approximation of the angular, surface or volume integrals depends on the number of angles, the chosen quadrature formula and the radial and axial distances between trajectories. All these parameters are decided by the user and are generally chosen to obtain a compromise between precision and computational time. As a consequence smaller regions will be approximated with a larger relative error in comparison with larger ones, since the number of trajectories crossing a region is proportional to the region size.

Small total cross sections

Numerical instabilities associated to small total cross section values are quite common in the neutron transport field. The polynomial method implemented in this work is even more sensible to this kind of problem than the SC one. Looking at Eq.(5.22), it is possible to understand the origin of this problem. This equation expresses the neutron angular balance for a given region. If we express each polynomial term more explicitly we obtain:

$$\begin{aligned}
\Sigma_r\ {}'\psi_{r,0}(\vec{\Omega}) &= \left(\bar{\bar{\mathcal{P}}}_r(\vec{\Omega})\ \vec{q}_r(\vec{\Omega})\right)_0 - \Delta\vec{J}_{r,0}(\vec{\Omega}) \\
\Sigma_r\ {}'\psi_{r,1}(\vec{\Omega}) &= \left(\bar{\bar{\mathcal{P}}}_r(\vec{\Omega})\ \vec{q}_r(\vec{\Omega})\right)_1 - \Delta\vec{J}_{r,1}(\vec{\Omega}) + \mu\frac{1}{\Delta z/2}\ {}'\psi_{r,0}(\vec{\Omega}) \\
\Sigma_r\ {}'\psi_{r,2}(\vec{\Omega}) &= \left(\bar{\bar{\mathcal{P}}}_r(\vec{\Omega})\ \vec{q}_r(\vec{\Omega})\right)_2 - \Delta\vec{J}_{r,2}(\vec{\Omega}) + \mu\frac{2}{\Delta z/2}\ {}'\psi_{r,1}(\vec{\Omega}) \\
&\vdots
\end{aligned}$$

The first equation corresponds to the SC angular balance. It should be noted that the currents term $\Delta\vec{J}_{r,0}(\vec{\Omega})$ is expected to be very small for a low total cross section region, since each trajectory contribution given by the difference between entering and exiting angular fluxes, can be very small, for small optical lengths. If no care is taken, and the current term is simply computed as the difference between the entering and exiting angular fluxes, this operation will strongly suffer of numerical cancellation. This ill calculated difference is then divided by the total cross section, in order to compute the constant flux term $\psi_{r,0}(\vec{\Omega})$, and this may result in a very large error. The classic workaround for the constant term is to compute the currents term contributions as:

$$\psi^-(\vec{\Omega}) - \psi^+(\vec{\Omega}) = (1 - e^{-\Sigma_r\, l})\ \left(\frac{q_{r,p}(\vec{\Omega})}{\Sigma_r} - \psi^-(\vec{\Omega})\right),$$

which corresponds to the strategy adopted in the SC, both for vacuum treatment, and for optimal number of floating-point operations. This allows obtaining a correct estimation of the constant terms. The polynomial method is more sensible to this problem than the SC one, since each high order term features the difference between a current term and the previous angular flux moment, divided by the total cross section. This means that the p-th order equation will be divided by Σ_r^{p+1}.

This kind of problem affects both the transport and the acceleration operators. Even if the acceleration equations do not use an angular version of the balance equation, the same problem holds after the angular integration.

Poor numerical discretization

The second numerical instability we encountered is due to poor numerical integration caused either by a lack of precision of the angular quadrature formula or by the fact that the chosen trajectory axial or radial spatial integration steps are too large. Both phenomena translate in a low number of trajectories crossing a region. As a consequence, the numerical quantities computed using the trajectory-based discretization can be very different when compared to the analytic counterparts. When this happens, we have observed numerical instabilities similar to the ones caused by small cross sections, which results in the divergence of the inner iterations. This seems to be caused mainly by the inaccurate computation of the numerical $\bar{\bar{\mathcal{P}}}_r(\vec{\Omega})$ angular matrices (5.20). Even if this numerical evaluation is necessary for particle conservation, as described in Appendix A, completely inaccurate values seems to lead to such important instabilities.

Issues regarding the acceleration matrices

The set of operations described in Section 6.3, ending up in the final DP_N balance expressed by Eq.(6.35), suffers from ill conditioning due to both small total cross sections and/or poor numerical integration. This can lead to acceleration matrices so wrongly computed, to cause the transport solution to oscillate of several orders of magnitude during iterations. Since the set of operations needed to arrive to the final balance is constituted by several phases, it is difficult to keep track of the numerical quality of the coefficients. The most evident numerical degradation of the coefficients appears after the application of the $\mathfrak{X}^{-1}$ matrix (6.34). However, the problem is likely to be not only related to the matrix inversion, but depending on the *quality* of all the matrices.

Proposed solutions

The method used to stabilize the problem and to assess the coefficients' *quality*, both for transport and acceleration, was actually derived by the debugging technique we have used to implement the DP_N acceleration: the infinite medium (known) solution is used to feed a source term of Eq.(5.28), for transport, and Eq.(6.35), for the acceleration. These equations are then used to recompute the infinite medium solution for a given 3D region and energy group. If the maximum relative difference between the analytical and the numerical infinite medium solutions, on all the polynomial and angular terms, is found to be larger than a certain threshold (here 1% for transport and 10% for acceleration have been used), the order of the polynomial used for the given region and energy group is reduced by one, and the matrices are recomputed. For the acceleration, this means starting again from Eq.(6.31). Note that in order to avoid division by zero all the errors are computed relatively to the constant term. This approach revealed to be a powerful tool to stabilize our method, and

	Transport cut-off		Acceleration cut-off	
Small cyclic assembly →	p = 0	21.74 %	p = 0	22.38 %
	p = 1	0.01 %	p = 1	6.52 %
	p = 2	78.25 %	p = 2	71.11 %

Table C.1 – Cut-off applied to the *small cyclic assembly* calculation presented in Table 8.2.

	Transport cut-off		Acceleration cut-off	
Half column assembly →	p = 0	7.42 %	p = 0	7.84 %
	p = 1	0.59 %	p = 1	4.77 %
	p = 2	91.99 %	p = 2	87.39 %

Table C.2 – Cut-off applied to the *half column assembly* calculation presented in Table 8.3.

	Transport cut-off		Acceleration cut-off	
Full column assembly →	p = 0	16.24 %	p = 0	22.26 %
	p = 1	13.70 %	p = 1	16.57 %
	p = 2	0.20 %	p = 2	13.13 %
	p = 3	69.85 %	p = 3	48.04 %

Table C.3 – Cut-off applied to the *full column assembly* calculation in nominal condition with 15 axial meshes and a polynomial order equal to 3, presented in Table 8.5.

	Transport cut-off		Acceleration cut-off	
Full column assembly →	p = 0	46.97 %	p = 0	48.25 %
	p = 1	6.76 %	p = 1	8.02 %
	p = 2	0.14 %	p = 2	7.38 %
	p = 3	46.14 %	p = 3	36.35 %

Table C.4 – Cut-off applied to the *full column assembly* calculation in voided condition with 15 axial meshes and a polynomial order equal to 3, presented in Table 8.6.

it has confirmed that the greatest errors are located in small regions containing low cross sections. This procedure is referred to in section 8.4 as *cutoff*. As an example, tables C.1 through C.4 show the cut-off percentage for some of the polynomial calculations presented in tables 8.2, 8.3, 8.5 and 8.6.

The last, and a little bit more heuristic, stabilization strategy consists in evaluating the *quality* of the $\bar{\bar{\mathcal{P}}}_r(\vec{\Omega})$ matrix numerically computed using Eq.(5.20). Even if the method described in the previous paragraph operates a reduction of the polynomial value for certain regions and energy group, it is based on angular integrated information. As a consequence, if for a given angle the $\bar{\bar{\mathcal{P}}}_r(\vec{\Omega})$ matrix is poorly computed, but in average its values are acceptable, the previous strategy can reveal to be too gentle. To reduce the damages caused by a bad angular integration, the *quality* of this matrix per each 3D region and angle is assessed by computing the inverse matrix by Gauss elimination, even if the inverse is not used for the real calculation. If the matrix is found to be singular, the accepted

polynomial order for the given region and angle is reduced and the inversion is repeated. The matrix is assumed to be singular when the maximum pivot is smaller than a certain tolerance, here imposed equal to 10^{-3}, to be compared with 1, which is the first element of the $\bar{\bar{\mathcal{P}}}_r(\vec{\Omega})$ matrix. The tolerance value has been chosen in a heuristic manner and proved to stabilize the iteration without causing a large reduction of the polynomial order, unless a catastrophic angular and spatial integration is performed. If no trajectories cross a certain region for a given angle, the polynomial degree is considered automatically reduced to zero, and all the elements of the $\bar{\bar{\mathcal{P}}}_r(\vec{\Omega})$ matrix are replaced by zero, except for the first one, which is set to 1. An equivalent procedure is applied to the $\mathfrak{Z}_{\alpha_v^{\pm}}$ matrices defined by Eq.(6.16). In this case preventing a bad inversion is mandatory, since the $\mathfrak{Z}_{\alpha_v^{\pm}}$ inverse matrices are applied in Eqs.(6.25). This procedure has already been described in details in [49].

D. Flux reconstruction

We present in this appendix the scheme used to export the results of the polynomial method into a more refined axial mesh, mainly for graphical reasons such as flux plots and to be able to compare the polynomial results with a piecewise constant solution, such as the results of the SC or a Monte Carlo calculation scored on a certain mesh. Here r denotes a region in the polynomial meshes, while i refers to a region on the *output* discretization, as explained in Fig. 35.

We are interested in the moments of the flux defined as in Eq.(2.10):

$$\Phi^n(\vec{r}) = \oint \frac{d\vec{\Omega}}{4\pi}\, A_n(\vec{\Omega})\, \psi(\vec{r},\vec{\Omega}).$$

We use the polynomial expansion of Eq.(5.2) and we look for an average value in a region i, getting:

$$\begin{aligned}\Phi_i^n = \frac{1}{V_r}\int_i d\vec{r}\; \Phi^n(\vec{r}) =& \frac{1}{\Delta z_i}\int_{\bar{z}_i-\frac{\Delta z_i}{2}}^{\bar{z}_i+\frac{\Delta z_i}{2}} dz \oint \frac{d\vec{\Omega}}{4\pi}\, A_n(\vec{\Omega})\, \vec{P}(\tilde{z}_r)\cdot\vec{\psi}_r(\vec{\Omega}) \\ =& \frac{1}{\Delta z_i}\int_{\bar{z}_i-\frac{\Delta z_i}{2}}^{\bar{z}_i+\frac{\Delta z_i}{2}} dz\; \vec{P}(\tilde{z}_r)\cdot\vec{\Phi}_r^n.\end{aligned}$$

Integrating analytically:

$$\int_{\bar{z}_i-\frac{\Delta z_i}{2}}^{\bar{z}_i+\frac{\Delta z_i}{2}} dz\; P_p(\tilde{z}) = \int_{\bar{z}_i-\frac{\Delta z_i}{2}}^{\bar{z}_i+\frac{\Delta z_i}{2}} dz\; \left(\frac{z-\bar{z}_r}{\Delta z_r/2}\right)^p = \frac{\Delta z_r}{2}\int_{\tilde{z}_i^{low}}^{\tilde{z}_i^{up}} d\tilde{z}\; \tilde{z}^p = \frac{\Delta z_r}{2}\frac{1}{p+1}\left[(\tilde{z}_i^{up})^{p+1} - \left(\tilde{z}_i^{low}\right)^{p+1}\right].$$

In a compact form:

$$\Phi_i^n = \vec{X}_{r,i}\cdot\vec{\Phi}_r^n$$

where:

$$\vec{X}_{r,i} = \{X_{r,i,p}\}, \qquad X_{r,i,p} = \frac{1}{2}\frac{\Delta z_r}{\Delta z_i}\frac{1}{p+1}\left[(\tilde{z}_i^{up})^{p+1} - \left(\tilde{z}_i^{low}\right)^{p+1}\right],$$

and

$$\tilde{z}_i^{up} = \frac{\bar{z}_i-\bar{z}_r}{\Delta z_r/2} + \frac{\Delta z_i}{\Delta z_r}, \qquad \tilde{z}_i^{low} = \frac{\bar{z}_i-\bar{z}_r}{\Delta z_r/2} - \frac{\Delta z_i}{\Delta z_r}.$$

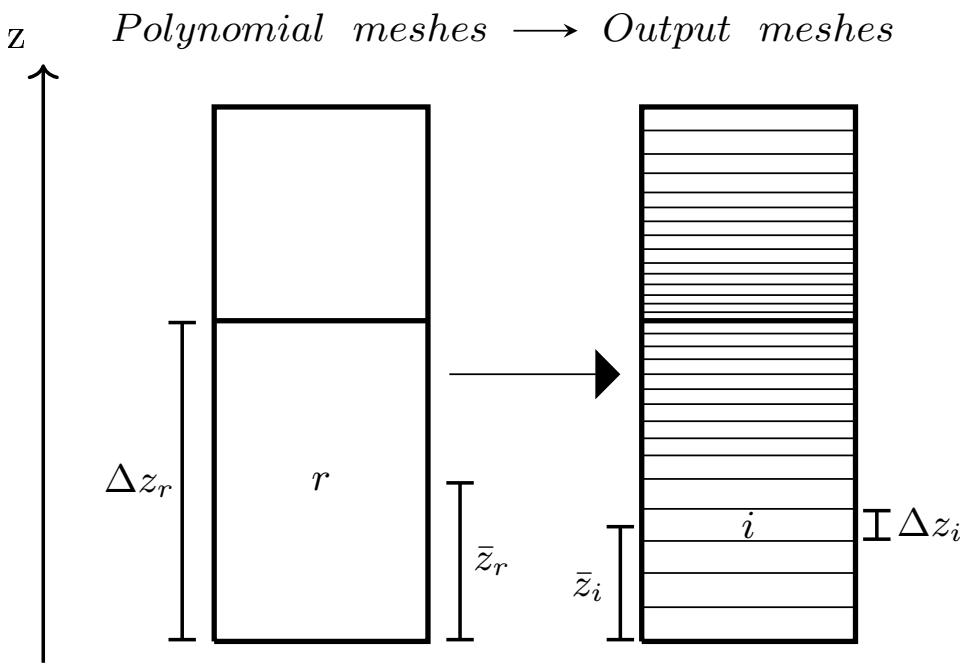

z
Polynomial meshes ⟶ Output meshes
Δz_r
r
$\bar{z}_r$
$\bar{z}_i$
i
Δz_i

Figure 35

Résumé en français

L'objectif de ce travail de thèse est le développement d'une approximation polynomiale axiale dans un solveur basé sur la Méthode des Caractéristiques (MOC). Le contexte, est celui de la solution stationnaire de l'équation de transport des neutrons pour des systèmes critiques, et l'implémentation pratique a été réalisée dans le solveur "two/Three Dimensional Transport" (TDT), faisant partie du projet APOLLO3®.

Un solveur MOC pour des géométries en trois dimensions a été implémenté dans ce code pendant un projet de thèse antécédent, se basant sur une approximation constante par morceaux du flux et des sources des neutrons. Les développements présentés dans la suite représentent la continuation naturelle de ce travail.

L'équation de transport des neutrons décrit fidèlement le comportement de la population des neutrons à l'intérieur du réacteur. La solution de cette equation peut être calculée de façon exacte seulement pour des cas symplifiees. Pour des vrai géométries de calcul on est obligés d'utiliser des solutions numeriques.

Le développement de ces méthodes numériques comporte forcement l'introduction d'approximations plus ou moins importantes. Dans les dernières décennies on assiste à des importants progrès des outils de calculs. Grâce à ces importants avancements technologiques certaines approximations introduites peuvent être relâchées et on cherche de plus en plus d'obtenir des simulations précises, pour des domaines des calculs plus larges. Obtenir des solutions plus précises pour des domaines des calculs plus importants pourrait être obtenu juste en augmentant la taille des machines des calcul utilisées. Par contre, ceci comporterait des plus importantes dépenses.

Une deuxième façon d'aborder le sujet est d'utiliser des meilleur instrument numériques pour essayer de décrire les phénomènes qu'on veut simuler. Ce travail de thèse s'inscrit dans ce contexte. L'approximation polynomiale qu'on a introduite a comme objectif de réduire le cout computationnel de la méthode numérique, pour faire en sorte de pouvoir traiter des geoetries plus grandes, avec les mêmes ressources computationnelles.

Les solveurs basés sur la méthode des caractéristiques en trois dimensions sont capables de produire des résultats précis pour des géométries complexes. Bien que précis, le coût computationnel associé à ce type de solveur est très important. Une représentation polynomiale en direction axiale du flux angulaire des neutrons a été utilisée pour réduire ce coût computationnel.

La méthode des caractéristiques utilise la forme intégrale de l'équation du transport des neutrons. Cette équation permet de calculer l'atténuation du flux neutronique le long d'une ligne caractéristique. Etant donné qu'on traite des particules sans charge électrique, ces lignes caractéristiques sont des simples lignes droites, qu'on appelle en général trajectoires. L'application concrète des méthodes des caractéristiques demande de tracer une série de trajectoires jusqu'à recouvrir tout le domaine de calcul. Successivement le flux neutronique

est propagé à partir d'une frontière du système, jusqu'à une autre. La forme intégrale de l'équation du transport, généralement appelée transmission, doit être résolue à chaque fois qu'une trajectoire sort d'une région de calcul, pour rentrer dans la prochaine.

Des nombreuses méthodes existent pour calculer la solution de l'équation du transport des neutrons. Pour calculer la solution de cette équation pour un entier cœur d'un réacteur nucléaire, le problème est en général découpé en plusieurs étapes. Dans une première partie on fait un calcul réseau. Ce calcul se constitue d'un calcul fin à niveau spatial et énergétique pour des petits sous-motifs représentatifs du réacteur.

Les résultats obtenus avec ce premier calcul sont ensuite utilisés pour obtenir des variables moyennes représentatives de ces sous-motifs. Avec ces variables grossières est possible obtenir une solution pour toute les géométries de calculs avec un cout computationnel raisonnable.

La méthode des caractéristiques est principalement utilisée dans la première étape du calcul. Le principal avantage de cette méthodes est de pouvoir être appliquée presque a tout type de géométries, du moment qu'on est capable de calculer les intersections d'une séries des lignes droites et le surfaces des régions de calcul. Cet important avantage fait en sorte que la méthode des caractéristiques soit une des plus utilisée pour les calculs de réseau.

Le fait de pouvoir calculer presque n'importe quel type de géométries c'est un important avantage, en particulier dans le domaine du nucléaire. Typiquement les géométries des réacteurs sont caractérisées par un certain nombre d'éléments qui peuvent être décrit avec des arcs de cercles. En utilisant des autres méthodes, comme des éléments fini, pour exemple, on serait obligé a représenter les bords des cercles comme une séries de segments droits, en résultant en une approximation plus importante, ou a un cout computationnel plus important.

La méthode des caractéristiques est principalement utilisée pour des calcul à deux dimensions, typiques des calculs réseau. Cette méthode peut être appliquée aussi pour des calculs en trois dimensions. La principale limite de cette approche est le cout computationnel associé. Dans un travail de thèse précèdent à celui-ci, une version 3D du solveur MOC TDT a été implémentée. Ce travail s'est concentré sur l'utilisation d'une approximation constante du flux dans les régions de calcul. Cette approximation est plutôt courante dans la méthode des caractéristiques, parce que c'est une approximation assez simple à implémenter et peut livrer des résultats très précis. La limite principale de cette approche est que les résultats seront précis seulement si la taille des regions de calcul est petites en comparaison avec les gradients du flux.

Les résultats de ce travail précédent montrent qu'un important nombre de mailles axiales est nécessaire pour représenter correctement les gradients axiaux du flux de neutrons, même à l'intérieur d'un seul matériel. Autrement dit, on nécessite d'une importante discrétisation spatiale en direction axiale pour réussir à représenter les gradients du flux. Une importante discrétisation en direction axiale se traduit dans un important cout computationnel et aussi de mémoire vive utilisée au sein du calcul. Ce dernier aspect est en train de devenir de plus en plus important, pour faire en sorte d'exploiter au mieux les nouvelles machines de calculs.

Suite à ces considérations, ce travail a consisté dans le développement d'une approximation polynomiale pour représenter la dépendance axiale du flux angulaire. L'objectif de cette approche est de réduire le nombre de mailles axiales utilisées, tout en gardant la même précision. L'utilisation de méthode d'ordre supérieur est très courante dans ce genre de

problèmes. Une méthode d'ordre supérieur implique en général un surcout computationnel, parce qu'elle ajoute des inconnues et des opérations flottantes aux problèmes. D'autre part, uen méthode d'ordre supérieur fait en sorte en général d'avoir besoin d'une mineure discrétisation. L'effet dû à l'utilisation d'une méthode d'ordre supérieur sera donc la somme de ces deux effets opposés.

On a choisi d'utiliser une approximation polynomiale pour représenter la dépendance spatiale du flux angulaire mais seulement en direction axiale. Ce choix est justifié par le fait que les géométries typiques des réacteurs sont beaucoup plus hétérogènes en direction radiale, que non axiale. On a retenu qu'une expansion polynomiale aussi en direction radiale aurait comporté un surcout computationnel qui aurait été difficilement contrebalancé par la réduction du nombre d'inconnues, étant donné que le nombre d'inconnus croit de façon non-linéaire avec le degré du polynôme.

Le travail réalisé pendant cette thèse peut être considéré comme divisé en trois parties: *transport*, *accélération* et *autres*.

La première partie est constituée par l'implémentation de l'approximation polynomiale choisie dans les équations de transmission et de bilan typiques de la méthode des caractéristiques. L'équation de transmission polynomiale qu'on a implémenté s'est révélée être beaucoup plus chère du point de vu computationnel par rapport à l'équation de la méthode Step. En effet le nombre d'opération flottantes que cette équation implique est strictement liés au nombre de cordes 3D du système. Bien que la méthode polynomiale ait permis une importante réduction du nombre des mailles axiales, le nombre des cordes 3D n'est pas diminué de façon proportionnelle. Ce qui a résulté dans une phase de transmission beaucoup plus chère du point de vue computationnel. Par contre, la réduction du nombre d'inconnues a impliqué une importante réduction de la mémoire engagée pendant le calcul. Cette réduction de la mémoire a aussi un important impact sur les temps de calcul.

Cette première partie a aussi été caractérisée par le calcul d'une série de coefficients numériques qui se sont révélés nécessaires afin d'obtenir un algorithme stable. En effet, dans une première partie de ce travail une série de matrices nécessaires pour obtenir la version polynomiale de l'équation de bilan, étaient calculée de façon analytique. Du point de vu computationnel et de la mémoire cette approche est très avantageuse. Par contre le calcul des coefficients analytiques à causé des importants effets d'instabilités numériques. Apres une première partie d'incompréhension, on est arrivés à une version stable de l'algorithme. Cette version stable a été obtenue en calculant une série de coefficients en utilisant la même discrétisation des volumes propre de la méthode des caractéristiques.

Pendant la deuxième partie de ce travail de thèse, on a modifié et implémenté la solution des équations de la méthode d'accélération DP_N. Cette méthode était déjà utilisée pour l'accélération et des itérations internes et externes dans TDT pour les solveurs deux et trois dimensionnels avec l'approximation des flux plat, quand ce travail a commencé. L'introduction d'une approximation polynomiale a demandé plusieurs développements numériques regardant la méthode d'accélération.

L'idée de la méthode d'accélération est de résoudre un problème simplifié, en comparaison à la solution de l'équation du transport, mais de le faire avec un cout computationnel réduit. Cette solution approchée est utilisée pour accélérer la convergence de la solution de l'équation du transport, qui est recherchée de façon itérative. Une méthode d'accélération est toujours nécessaire pour ce type de méthode.

L'idée de l'accélération DP_N est de éxpandre la dépendance angulaire des flux surfaciques

est des sources volumiques avec une séries de harmoniques sphériques réelles. En plus, la surface de chaque région de calcul est considérée comme décomposé dans une série de surfaces. En utilisant l'approximation polynomiale, en plus de l'expansion de la dépendance angulaire, la dépendance spatiale en direction axiale des flux surfaciques et des sources est représentée avec la même expansion polynomiale utilisée pendant la solution de la partie transport. Ces approximations permettent d'écrire le courant partiel entrant et sortant pour chaque surface de chaque région comme un système linéaire. La solution de ce système linéaire est obtenue avec une méthode itérative de Krylov.

Dans la dernière partie de ce travail on a recherché des solutions pour un mélange de différents problèmes liés aux premières deux parties. En premier lieux, on a eu à faire avec des instabilités numériques associées à une discrétisation spatiale ou angulaire pas suffisamment précise, soit pour la partie transport que pour la partie d'accélération. Un autre type d'instabilité numérique qu'on a rencontré est lié à la présence de sections efficaces petites. La présence de matériels avec une très faible section efficaces cause des importantes instabilités dans l'algorithme qu'on a développé parce, causé par des successives divisions de quantités toujours plus petites. Pour remédier à cet inconvénient on a décidé de réduire le degré du polynôme utilisé localement, pour la région de calcul et le groupe donné.

Ensuite, on a essayé d'utiliser différentes méthodes pour réduire l'empreinte mémoire des coefficients d'accélération. La méthode d'accélération implémentée dans TDT demande le stoquage d'un set de coefficients pour chaque groupe d'énergie. Ces matrices peuvent être très importantes en mémoire. Le nombre de groupe typiquement utilisé peut varier entre une dizaine et quelque millier. Quand le nombre de groupe devient important, la plupart de la mémoire occupée par le calcul est constitué par le stoquage des matrices d'accélération. Ce phénomène peut limiter le type de calcul qui peut être réalisé parce que d'un côté ils sont trop longs sans accélération, mais de l'autre ils peuvent demander plus de mémoire par rapport à celle qui est disponible sur une certaine machine. Une approche possible consiste à recalculer ces coefficients pendant les itérations. Cette approche aurait l'avantage de ne nécessiter de aucun stoquage. D'autre côte, le calcul de ces coefficients est très important, donc cette approche résulterait dans un cout computationnel inacceptable.

L'approche qu'on a finalement choisie pour essayer de réduire l'empreinte mémoire de la méthode se base sur une régression non-linéaire au sens des moindres carrés de la dépendance en fonction des sections efficaces typique de ces coefficients. L'approche standard consiste dans le stockage d'une série de coefficients pour chaque groupe d'énergie. La méthode de régression permet de remplacer cette information avec une série de coefficients calculés pendant la régression qui sont utilisés pour reconstruire les matrices d'accélération au cours des itérations. La méthode sera donc avantageuse seulement si le nombre de coefficients utilisé est mineur au nombre des groupes originel. Cette procédure ajoute un certain coût computationnel à la méthode, mais nous pensons que la réduction de la mémoire rende ce surcoût acceptable.

La méthode de régression qu'on a implémenté s'est révélé être capable des représenter 99% des matrices d'accélération en utilisant 9 coefficients, avec une précision relative de 1%. Les valeurs qui ne sont pas correctement représenté par le modelé de régression doivent être stoqués.

En conclusion, le travail réalisé a été concentré sur l'application d'une simple approximation polynomiale avec l'objectif de réduire le cout computationnel et l'empreinte mémoire associées à un solveur basée sur les méthodes des caractéristiques qui est utilisé pour calculer le flux neutroniques pour des géométries à trois dimensions extrudées. Même si cela ne con-

stitue pas une amélioration radicale des performances, l'approximation d'ordre supérieur qu'on a introduit permet une réduction en termes de mémoire et de temps de calcul d'un facteur compris entre 2 et 5, selon le cas. Nous pensons que ces résultats constitueront une base fertile pour des futures améliorations.

En particulier, la perspective la plus intéressante de ce travail consiste dans le développement d'une approximation polynomiale pour les sections efficaces similaire à celle utilisée pour représenter la dépendance spatiale axiale du flux angulaire. Une approximation de telle sorte permettrait de réaliser des calculs d'évolution isotopique. Sans une approximation polynomiale des sections efficaces, on serait limités à calculer la solution du problème avec des concentrations nominales, mais on ne pourrait pas simuler l'évolution du combustible, parce que cette évolution nécessite de récupérer la dépendance polynomiale des moments du flux neutroniques.

List of Figures

List of Tables

Glossary

ASTRID Advanced Sodium Technological Reactor for Industrial Demonstration. 31, 32, 69, 78, 79, 81, 90, 91, 92, 93, xvii, xviii

CCM Chords Classification Method. 26, 28, 29, 53, 54, 67, 69, 70

CEA Commissariat à l'Énergie atomique et aux Énergies alternatives. 5, 24, 31

CMFD Coarse-Mesh Finite Difference. 56

Gen-IV Generation IV. 5, 78

HSS Hit Surfaces Sequence. 28, 29, 30, xvii

IDT Integro-Differential Transport. 22, 34

LEAF Legendre polynomial Expansion of the Angular Flux. 35

MOC Method Of Characteristics. 2, 4, 5, 18, 20, 21, 22, 24, 25, 26, 29, 30, 31, 32, 34, 35, 36, 37, 40, 44, 56, 67, 95, 96, v, vi, xii, xxv

NTE Neutron Transport Equation. 4, 8, xxv

SC Step Characteristics. 24, 31, 32, 36, 40, 44, 46, 53, 54, 55, 56, 63, 66, 67, 69, 71, 78, 81, 82, 83, 85, 95, vi, vii, x, xviii

SMOC Short Method Of Characteristics. 22

TDT two/Three Dimensional Transport. 4, 5, 12, 16, 20, 21, 22, 24, 25, 32, 34, 35, 36, 50, 56, 71, 95, xii, xiv, xxv

Bibliography

[1] J. K. Shultis and R. E. Faw, *Fundamentals of nuclear science and engineering.* CRC Press, 2002. (cited at pp: 9)

[2] G. I. Bell; and S. Glasstone, *Nuclear Reactor Theory.* New York: Van Nostrand Reinhold Company, 1970. (cited at pp: 12)

[3] R. Sanchez and A. Chetaine, "Synthetic acceleration for a two-dimensional characteristic method in unstructured meshes," *Nuclear Science and Engineering*, vol. 136, no. 1, pp. 122–139, 2000. (cited at pp: 12)

[4] R. Sanchez, "Assembly homogenization techniques for core calculations," *Progress in Nuclear Energy*, vol. 51, no. 1, pp. 14–31, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.pnucene.2008.01.009 (cited at pp: 15)

[5] A. Hébert, *Applied Reactor Physics.* Montréal, Canada: Presses internationales Polytechnique, 2009. (cited at pp: 17, 21)

[6] L. Mao, I. Zmijarevic, and R. Sanchez, "Resonance Self-Shielding Methods for Fast Reactor Calculations—Comparison of a New Tone's Method with the Subgroup Method in APOLLO3®," *Nuclear Science and Engineering*, vol. 188, pp. 15–32, 2017. (cited at pp: 17, 18)

[7] J. R. Askew, "A characteristics formulation of the neutron transport equation in complicated geometries," Atomic Energy Authority, Reactor Group, Winfrith (United Kingdom), 1972. (cited at pp: 20)

[8] M. Halsall, "CACTUS, a characteristics solution to the neutron transport equations in complicated geometries," UKAEA Atomic Energy Establishment, Winfrith (United Kingdom), 1980. (cited at pp: 20)

[9] R. Sanchez and N. J. McCormick, "A Review of Neutron Transport Approximations," *Nuclear Science and Engineering*, vol. 80, pp. 481–535, 1982. (cited at pp: 20)

[10] E. W. Larsen and R. E. Alcouffe, "Linear characteristic method for spatially discretizing the discrete ordinates equations in (X,Y)-geometry," in *ANS/ENS Joint Topical Meeting, Mathematical Methods in Nuclear Engineering*, Munich, 1981. (cited at pp: 20)

[11] W. L. Filippone, S. Woolf, and R. J. Lavigne, "Particle Transport Calculations with the Method of Streaming Rays," *Nuclear Science and Engineering*, vol. 77, no. 2, pp. 119–136, 1981. (cited at pp: 20)

[12] I. R. Suslov, "Solution of transport in 2- and 3-dimensional irregular geometry by the method of characteristics," in *Mathematical methods and supercomputing in nuclear applications*, Karlsruhe, Germany, 1993. (cited at pp: 20)

[13] D. Knott and M. Edenius, "Validation of the CASMO-4 Transport Solution," in *Mathematical methods and supercomputing in nuclear applications*, Karlsruhe, Germany, 1993. (cited at pp: 20)

[14] N.-Z. Cho and S.-G. Hong, "CRX: a transport theory code for cell and assembly calculations based on characteristic method," in *PHYSOR*, Taejeon, Japan, 1996. (cited at pp: 20)

[15] R. Roy, "The cyclic characteristics method," in *International conference on physics of nuclear science and technology*, Long Island, New York, 1998. (cited at pp: 20)

[16] A. Chetaine, R. Sanchez, and L. Erradi, "The use of the characteristics method to solve the transport equation in unstructured geometries," *Radiation Physics and Chemistry*, vol. 61, no. 3-6, pp. 763–765, 2001. (cited at pp: 20)

[17] R. Sanchez, L. Mao, and S. Santandrea, "Treatment of boundary conditions in trajectory - based deterministic transport methods," *Nuclear Science and Engineering*, vol. 140, no. 1, pp. 23–50, 2002. (cited at pp: 20, 31)

[18] N. Z. Cho, "Fundamentals and recent developments of reactor physics methods," *Nuclear Engineering and Technology*, vol. 37, no. 1, pp. 25–78, 2005. (cited at pp: 21)

[19] A. Yamamoto, A. Giho, Y. Kato, and T. Endo, "GENESIS : A Three-Dimensional Heterogeneous Transport Solver Based on the Legendre Polynomial Expansion of Angular Flux Method," *Nuclear Science and Engineering*, vol. 186, pp. 1–22, 2017. (cited at pp: 22, 35, 71)

[20] R. Ferrer and Y. Azmy, "A Robust Arbitrarily High-Order Transport Method of the Characteristic Type for Unstructured Grids," *Nuclear Science and Engineering*, vol. 172, no. 1, pp. 33–51, 2012. (cited at pp: 22, 36)

[21] E. Masiello, R. Sanchez, and I. Zmijarevic, "New Numerical Solution with the Method of Short Characteristics for 2-D Heterogeneous Cartesian Cells in the APOLLO2 Code: Numerical Analysis and Tests," *Nuclear Science and Engineering*, vol. 161, no. September, pp. 257–278, 2009. (cited at pp: 22, 34)

[22] Y. S. Ban, E. Masiello, R. Lenain, H. G. Joo, and R. Sanchez, "Code-to-code comparisons on spatial solution capabilities and performances between nTRACER and the standalone IDT solver of APOLLO3®," *Annals of Nuclear Energy*, vol. 115, pp. 573–594, 2018. [Online]. Available: https://doi.org/10.1016/j.anucene.2018.02.011 (cited at pp: 22)

[23] D. Sciannandrone, "Acceleration and higher order schemes of a characteristic solver for the solution of the neutron transport equation in 3D axial geometries," Ph.D. dissertation, Université Paris-Sud, 2015. (cited at pp: 24, 26, 28, 29, 30, 32, 40, 50, 53, 54, 69, 78, 79, 81, 95)

[24] D. Sciannandrone, S. Santandrea, R. Sanchez, L. Lei-Mao, J. Vidal, J. Palau, and P. Archier, "Coupled fine-group three-dimensional flux calculation and subgroups method for a FBR hexagonal assembly with the APOLLO3® core physics analysis code," in *Mathematics and Computations, Supercomputing in Nuclear Applications and Monte Carlo International Conference, M&C+SNA+MC 2015*, Nashville, Tennessee, 2015. (cited at pp: 24)

[25] D. Sciannandrone, S. Santandrea, and R. Sanchez, "Optimized tracking strategies for step MOC calculations in extruded 3D axial geometries," *Annals of Nuclear Energy*, vol. 87, pp. 49–60, 2016. (cited at pp: 24, 25)

[26] S. Santandrea, D. Sciannandrone, R. Sanchez, L. Lei-Mao, and L. Graziano, "A neutron transport characteristics method for 3D axially extruded geometries coupled with a fine group self-shielding environment," *Nuclear Science and Engineering*, vol. 186, no. 3, pp. 239–276, 2017. (cited at pp: 24, 29, 56, 66, 72)

[27] E. E. L. Warren F. Miller, *Computational Methods of Neutron Transport*, 1st ed., 1993. (cited at pp: 26)

[28] A. Hébert, "High-Order Linear Discontinuous and Diamond Differencing Schemes Along Cyclic Characteristics," *Nuclear Science and Engineering*, vol. 184, no. Lc, 2016. (cited at pp: 34)

[29] T. Mazumdar and S. Degweker, "Solution of the neutron transport equation by the Method of Characteristics using a linear representation of the source within a mesh," *Annals of Nuclear Energy*, vol. 108, pp. 132–150, 2017. (cited at pp: 34)

[30] R. M. Ferrer and J. D. Rhodes III, "A Linear Source Approximation Scheme for the Method of Characteristics," *Nuclear Science and Engineering*, vol. 182, no. 2, pp. 1–15, 2016. (cited at pp: 34)

[31] R. Le Tellier and A. Hébert, "On the integration scheme along a trajectory for the characteristics method," *Annals of Nuclear Energy*, vol. 33, no. 14-15, pp. 1260–1269, 2006. (cited at pp: 34)

[32] R. Sanchez and S. Santandrea, "Convergence analysis for the method of characteristics in unstructured meshes," *Nuclear Science and Engineering*, vol. 183, pp. 196–213, 2016. (cited at pp: 34)

[33] S. Santandrea, J. Jaboulay, P. Bellier, F. Fevotte, and H. Golfier, "Improvements and validation of the linear surface characteristics scheme," *Annals of Nuclear Energy*, vol. 36, no. 1, pp. 46–59, 2009. (cited at pp: 34)

[34] E. Masiello, R. Clemente, and S. Santandrea, "High-Order Method of Characteristics for 2-D Unstructured Meshes," in *International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009) Reactor Physics*, Saratoga Springs, New York, 2009. (cited at pp: 34)

[35] G. Gunow, J. Tramm, B. Forget, K. Smith, and T. He, "SimpleMOC - A PERFORMANCE ABSTRACTION FOR 3D MOC," in *ANS MC2015 – Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA), and the Monte Carlo (MC) Method*, Nashville, Tennessee, 2015. (cited at pp: 35)

[36] G. Gunow, S. Shaner, B. Forget, and K. Smith, "Reducing 3D MOC Storage Requirements with Axial On-the-fly Ray Tracing," in *PHYSOR*, Sun Valley, Idaho, 2016. (cited at pp: 35)

[37] S. Shaner, W. Boyd, B. Forget, and K. Smith, "Accuracy and Performance of 3D MOC for Full-Core PWR Problems," in *M&C 2017 - International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering*, Jeju, Korea, 2017. (cited at pp: 35)

[38] X. M. Chai, K. Wang, and D. Yao, "The linear source approximation in three dimension characteristics method," in *International Conference on Mathematics, Computational Methods and Reactor Physics (M&C 2009)*, Saratoga Springs, New York, 2009, pp. 1–14. (cited at pp: 35)

[39] M. Hursin, B. Collins, Y. Xu, and T. Downar, "The Development and Implementation of a One-Dimensional Sn Method in the 2D-1D Integral Transport Solution," *Nuclear Science and Engineering*, vol. 176, no. 2, pp. 186–200, 2014. (cited at pp: 35)

[40] B. Collins, S. Stimpson, B. W. Kelley, M. T. H. Young, B. Kochunas, A. Graham, E. W. Larsen, T. Downar, and A. Godfrey, "Stability and accuracy of 3D neutron transport simulations using the 2D/1D method in MPACT," *Journal of Computational Physics*, vol. 326, pp. 612–628, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.jcp.2016.08.022 (cited at pp: 35)

[41] Y. S. Jung, C. B. Shim, C. H. Lim, and H. G. Joo, "Practical numerical reactor employing direct whole core neutron transport and subchannel thermal/hydraulic solvers," *Annals of Nuclear Energy*, vol. 62, pp. 357–374, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.anucene.2013.06.031 (cited at pp: 35)

[42] K. A. Mathews, R. L. Miller, and C. R. Brennan, "Split-Cell, Linear Characteristic Transport Method for Unstructured Tetrahedral Meshes," *Nuclear Science and Engineering*, vol. 136, pp. 178–201, 2000. (cited at pp: 36)

[43] C. R. Brennan, R. L. Miller, and K. A. Mathews, "Split-Cell Exponential Characteristic Transport Method for Unstructured Tetrahedral Meshes," *Nuclear Science and Engineering*, vol. 138, pp. 26–44, 2001. (cited at pp: 36)

[44] R. Sanchez, "Prospects in deterministic three-dimensional whole-core transport calculations," *Nuclear Engineering and Technology*, vol. 44, no. 5, pp. 113–150, 2012. (cited at pp: 36, 37, 44, 45, 95, i)

[45] S. Santandrea, "An Integral Multidomain DP_N Operator as Acceleration Tool for the Method of Characteristics in Unstructured Meshes," *Nuclear Science and Engineering*, pp. 223–235, 2007. (cited at pp: 42, 56)

[46] L. Graziano, S. Santandrea, and D. Sciannandrone, "Polynomial axial expansion in the Method of Characteristics for neutron transport in 3D extruded geometries," in *ICRS13-RPSD2016*, Paris, France, 2016. (cited at pp: 45)

[47] M. L. Adams and E. W. Larsen, "Fast iterative methods for discrete-ordinates particle transport calculations," *Progress in Nuclear Energy*, vol. 40, no. 1, pp. 3–159, 2002. (cited at pp: 56)

[48] I. R. Suslov, "An Algebraic Collapsing Acceleration Method for the Acceleration of the Inner (Scattering) Iterations in Long Characteristics Transport Theory," in *Supercomputing in Nuclear Applications (SNA)*, Commissariat à l'Énergie Atomique, Paris, France, 2003. (cited at pp: 56, 58)

[49] S. Santandrea, L. Graziano, and D. Sciannandrone, "Accelerated Polynomial axial expansions for full 3D neutron transport MOC in the APOLLO3® code system as applied to the ASTRID fast breeder reactor," *Annals of Nuclear Energy*, vol. 113, no. November, pp. 194–236, 2017. [Online]. Available: https://doi.org/10.1016/j.anucene.2017.11.010 (cited at pp: 63, ix)

[50] K. Madsen, H. B. Nielsen, and O. Tingleff, *Methods for Non-Linear Least Squares Problems*, 2004, vol. 2. (cited at pp: 75)

[51] Kenneth Levenberg, "A Method for the Solution of Certain Non-Linear Problems in Least Squares," *Quarterly of Applied Mathematics*, vol. 2, pp. 164–168, 1944. (cited at pp: 75)

[52] B. Fontaine, N. Devictor, P. Le Coz, A. Zaetta, D. Verwaerde, and J.-M. Hamy, "The french R&D on SFR core design and ASTRID project," in *Global*, Makuhari, Japan, 2011. (cited at pp: 78)

[53] P. Archier, J.-M. Palau, J. Vidal, S. Santandrea, and D. Sciannandrone, "Validation of the Newly Implemented 3D TDT-MOC Solver of APOLLO3® Code on a Whole 3D SFR Heterogeneous Assembly," in *PHYSOR*, Sun Valley, Idaho, 2016. (cited at pp: 85)

[54] C. Cohen-Tannoudji, F. Laloe, and B. Diu, *Mécanique quantique*, hermann ed., 1998. (cited at pp: iii)

université PARIS-SACLAY | ÉCOLE DOCTORALE PHENIICS

Titre: Méthode accélérée aux caractéristiques pour la solution de l'équation du transport des neutrons, avec une approximation polynomiale axiale

Mots clés: Méthode des caractéristiques, equation du transport des neutrons, approximation polynomiale, TDT, 3D MOC

Résumé: L'objectif de ce travail de thèse est le développement d'une approximation polynomiale axiale dans un solveur basé sur la Méthode des Caractéristiques (MOC). Le contexte, est celui de la solution stationnaire de l'équation de transport des neutrons pour des systèmes critiques, et l'implémentation pratique a été réalisée dans le solveur "two/Three Dimensional Transport" (TDT), faisant partie du projet APOLLO3®. Un solveur MOC pour des géométries en trois dimensions a été implémenté dans ce code pendant un projet de thèse antécédent, se basant sur une approximation constante par morceaux du flux et des sources des neutrons. Les développements présentés dans la suite représentent la continuation naturelle de ce travail. Les solveurs MOC en trois dimensions sont capables de produire des résultats précis pour des géométries complexes. Bien que précis, le coût computationnel associé à ce type de solveur est très important. Une représentation polynomiale en direction axiale du flux angulaire des neutrons a été utilisée pour réduire ce coût computationnel.

Le travail réalisé a été concentré sur l'application d'une simple approximation polynomiale avec l'objectif de réduire le cout computationnel et l'empreinte mémoire associées à un solveur basée sur la méthodes des caractéristiques qui est utilisé pour calculer le flux neutroniques pour des géométries à trois dimensions extrudées. Même si cela ne constitue pas une amélioration radicale des performances, l'approximation d'ordre supérieur qu'on a introduit permet une réduction en termes de mémoire et de temps de calcul d'un facteur compris entre 2 et 5, selon le cas. Nous pensons que ces résultats constitueront une base fertile pour des futures améliorations.

Title: An axial polynomial expansion and acceleration of the characteristics method for the solution of the Neutron Transport Equation

Keywords: Method of characteristics, neutron transport equation, polynomial approximation, TDT, 3D MOC

Abstract: The purpose of this PhD is the implementation of an axial polynomial approximation in a three-dimensional Method Of Characteristics (MOC) based solver. The context of the work is the solution of the steady state Neutron Transport Equation (NTE) for critical systems, and the practical implementation has been realized in the Two/Three Dimensional Transport (TDT) solver, as a part of the APOLLO3® project. A three-dimensional MOC solver for 3D extruded geometries has been implemented in this code during a previous PhD project, relying on a piecewise constant approximation for the neutrons fluxes and sources. The developments presented in the following represent the natural continuation of this work. Three-dimensional neutron transport MOC solvers are able to produce accurate results for complex geometries. Although accurate, the computational cost associated to this kind of solvers is very important. An axial polynomial representation of the neutron angular fluxes has been used to lighten this computational burden.

This work has focused on applying a simple polynomial approximation in order to reduce the computational cost and memory footprint associated to a MOC solver used to compute the neutron fluxes in three dimensional extruded geometries. Even if this does not constitute a radical improvement, the high order approximation that we have introduced allows a reduction in terms of memory and computational times of a factor between 2 and 5, depending on the case. We think that these results will constitute a fertile ground for further improvements.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France