Polynomial Characteristics Method for Neutron Transport in 3D extruded geometries

I. Zmijarevic, L. Graziano, S. Santandrea, D. Sciannandrone

M&C 2017 - International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering, Jeju, Korea, April 16-20, 2017, on USB (2017)

Polynomial Characteristics Method for Neutron Transport in 3D extruded geometries

Laurent Graziano, Simone Santandrea, Daniele Sciannandrone, Igor Zmijarevic [*]

[*]*CEA Saclay*
laurent.graziano@cea.fr, simone.santandrea@cea.fr, daniele.sciannandrone@cea.fr, igor.zmijarevic@cea.fr

Abstract - *A polynomial expansion in the axial direction has been implemented in the Method Of Characteristics framework, for the solution of the mono-group, steady state neutron transport equation for 3D extruded geometries. This work has been realized in the context of the APOLLO3® project and, in particular, in the Two & three Dimensional Transport (TDT) module. In this paper we weigh up benefits and disadvantages of this approach, when comparing to the classical Step Characteristics (SC) approximation, already available in APOLLO3®.*

I. INTRODUCTION

The solution of the neutron transport equation for 3D extruded geometries is available inside the TDT module of the APOLLO3® code. A solver based on the Method Of Characteristics (MOC) allowing the treatment of such geometries has been recently developed. The classic SC approximation is used in this work. For each region of the domain the source is approximated as a constant function. The results of this previous work showed very good agreement with the Monte-Carlo solution and good computational performances[1][2][3][4].

However, since the SC is adopted, a quite large number of meshes is need for a good solution convergence. Some results of the previous work and fluxes profiles of a 3D heterogeneous assembly of the ASTRID reactor will be shown to enhance this concept. Moreover, these results show that flux gradients in the axial direction can be efficiently represented using polynomial functions. Their shape is, in fact, quite regular. On the other hand in the radial plane a typical reactor geometry presents, in our opinion, too strong heterogeneities for a polynomial representation to be at the same time accurate and efficient.

The idea of a polynomial expansion of the method arises from these considerations. The seminal ideas of this method are already given in [3]. They are here developed in a different form.

II. PURPOSE OF THE METHOD

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), developed at the CEA in France. This project of a fast breeder reactor features a particular axial design in order to try to circumvent the typical issue of positive void reactivity coefficient of fast sodium cooled reactors. Fig. 1 shows an axial section of the reactor core, while Fig. 2 shows a radial and an axial section of the assembly used for the computation as well as the boundary condition imposed. It is easy to see how the strong axial heterogeneity makes of this reactor a perfect candidate to test the efficiency of a 3D transport solver.

Fig 3 shows the energy spectrum of the flux for a fuel pin inside the fissile material, while Fig 4 shows an axial profile of the flux computed using the SC method. This figure shows that the axial flux gradient are quite strong, both in fast, and in thermal groups. Moreover it allows to see how many axial meshes are needed to properly represent the flux shapes, using the flat source approximation. The idea to develop the source using a polynomial function arises from these considerations.

In the same framework we cite work realized by MIT researcher [5]. Here a quadratic function is used to represent the axial behaviour of the sources.

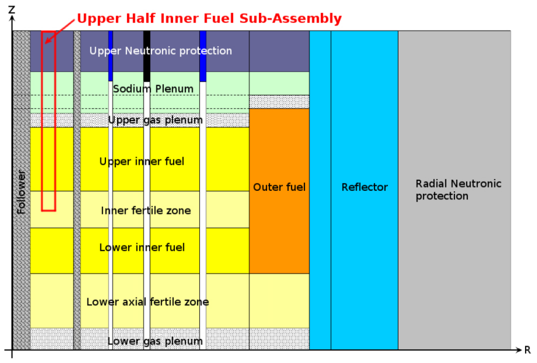

Fig. 1: Axial cut of the ASTRID reactor. The enlightened area represents the computational domain.

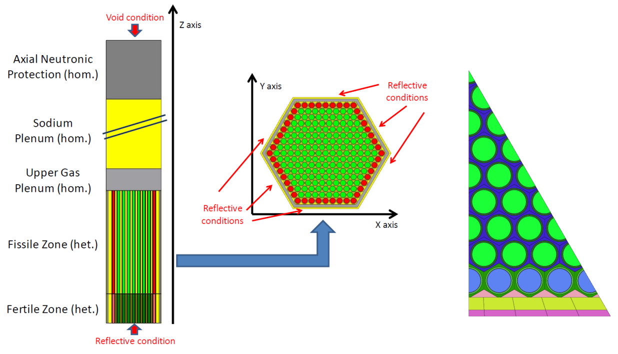

Fig. 2: Radial and axial sketch of the ASTRID fuel assembly with $2\pi/12$ symmetry. Even if not representative of the real geometry, a reflective boundary condition is also used for the lower part of the domain. On the right a 2D section of the actual computational domain. Different colors indicate different self-shielding zones.

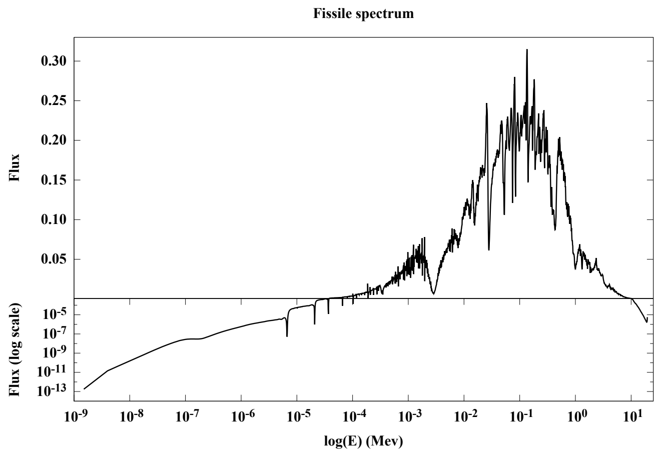

Fig. 3: Energy spectrum inside the fissile material resulting from the 1968 energy groups computation. The upper part of the graph is in linear scale, while the lower part is in logarithmic scale.

III. METHOD DESCRIPTION

The steady state neutron transport equation is the starting point of our problem.

$$\begin{aligned}&\left[\vec{\Omega}\nabla + \Sigma_t(\vec{r},E)\right]\Psi(\vec{r},\vec{\Omega},E) = q(\vec{r},\vec{\Omega},E) && \vec{r}\in D,\quad \vec{\Omega}\in S_{4\pi}\\ &\Psi(\vec{r},\vec{\Omega},E) = \Psi_{in} && \vec{r}\in\partial D,\ \vec{\Omega}\in S^-_{2\pi}\end{aligned} \tag{1}$$

The emission density q accounts for transfer and fission:

$$\begin{aligned}&q(\vec{r},\vec{\Omega},E) = \mathcal{H}\Psi(\vec{r},\vec{\Omega},E) + \frac{1}{k_{eff}}\mathcal{F}\Psi(\vec{r},\vec{\Omega},E)\\ &\mathcal{H}\Psi(\vec{r},\vec{\Omega},E) =\\ &\qquad\int_0^\infty dE' \oint \frac{d\vec{\Omega}'}{4\pi}\,\Sigma_s(\vec{r},E'\to E,\vec{\Omega}\cdot\vec{\Omega}')\Psi(\vec{r},\vec{\Omega}',E')\\ &\mathcal{F}\Psi(\vec{r},\vec{\Omega},E) =\\ &\qquad\chi(E)\int_0^\infty dE'\ \nu(E')\Sigma_f(\vec{r},E')\oint\frac{d\vec{\Omega}}{4\pi}\,\Psi(\vec{r},\vec{\Omega}',E')\end{aligned}$$

The energy dependence is treated with a multi-group approximation and the transfer cross section is expanded as customary using the real spherical harmonics:

$$q(\vec{r},\vec{\Omega},E) \simeq q^g(\vec{r},\vec{\Omega}) = q(\vec{r},\vec{\Omega})$$

with the assumption that when the group superscript is not explicitly written we are treating the generic g group.

$$\begin{aligned}&\mathcal{H}\Psi(\vec{r},\vec{\Omega}) = \sum_{n=1}^{N_m} A_n(\vec{\Omega})\sum_{g'}\Sigma_{s,n}^{g'\to g}(\vec{r})\oint\frac{d\vec{\Omega}'}{4\pi}\,A_n(\vec{\Omega}')\Psi^{g'}(\vec{r},\vec{\Omega}')\\ &\mathcal{F}\Psi(\vec{r},\vec{\Omega}) = \chi^g\sum_{g'}\nu^{g'}\Sigma_f^{g'}(\vec{r})\oint\frac{d\vec{\Omega}'}{4\pi}\,A_n(\vec{\Omega}')\Psi^{g'}(\vec{r},\vec{\Omega}')\end{aligned} \tag{2}$$

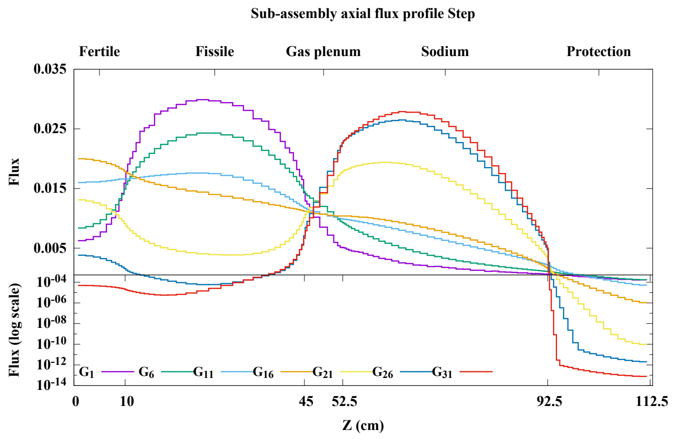

Fig. 4: Axial flux profile in a fuel pin obtained with a SC computation using 110 axial planes. The upper part of the graph is in linear scale, while the lower part is in logarithmic scale, in order to shows the different flux gradients in the neutron protection. The multigroup flux has been condensed on 33 energy group and fluxes on each groups have been normalized to fit in the same graph.

Where the sum over N_m is the short version of:

$$\sum_{n=1}^{N_m} A_n(\vec{\Omega}) = \sum_{l=0}^{l=K}\sum_{m=-l}^{m=l} A_l^m(\vec{\Omega})$$

where K is the anisotropy order, A_l^m are the real spherical harmonics and $N_m = (K+1)\times(K+1)$ represents the total number of angular moments.

Polynomial basis definition

A local polynomial basis can be used to express sources and flux moments:

$$\vec{P}(\tilde{z}_r) = \{\tilde{z}_r^p = \left(\frac{z_r - \bar{z}_r}{\Delta z_r/2}\right)^p, \quad 0\le p\le N_p\} \tag{3}$$

where Δz_r is the height of the given region, $z_r \in [-\Delta z_r/2, \Delta z_r/2]$ is the local coordinate for the r_{th} region, $\bar{z}_r$ is the value of the axial coordinate at the region center and N_p is the chosen order for the polynomial expansion.

Angular flux expansion

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

$$\boxed{\Psi(\vec{r},\vec{\Omega}) = \sum_{p=1}^{N_p} P_p(\tilde{z}_r)\cdot\Psi_{r,p}(\vec{\Omega}) = \vec{P}(\tilde{z}_r)\cdot\vec{\Psi}_r(\vec{\Omega})} \tag{4}$$

Where we assume that $\vec{\Psi}_r(\vec{\Omega})$ is a vector constant per region.

Replacing this is Eqs.(2) we get:

$$\mathcal{H}\Psi(\vec{r},\vec{\Omega}) = \sum_{n=1}^{N_m} A_n(\vec{\Omega}) \sum_{p=1}^{N_p} P_p(\tilde{z}_r) \sum_{g'} \Sigma_{s,n}^{g'\to g}(\vec{r})\, \Phi_{r,p}^{n,g'} \tag{5}$$

$$\mathcal{F}\Psi(\vec{r},\vec{\Omega}) = \sum_{p=1}^{N_p} P_p(\tilde{z}_r)\, \chi^g \sum_{g'} \nu^{g'} \Sigma_f^{g'}(\vec{r})\, \Phi_{r,p}^{1,g'} \tag{6}$$

where the flux moments are defined as:

$$\boxed{\Phi_{r,p}^{n,g'} = \oint \frac{d\vec{\Omega}'}{4\pi} A_n(\vec{\Omega}')\, \Psi_{r,p}^{g'}(\vec{\Omega}')} \tag{7}$$

Source expansion

Following the definition of Eqs. (5) and (6), the source term can be written in a similar way to the customary real spherical harmonics expansion:

$$q(\vec{r},\vec{\Omega}) = \sum_{n=1}^{N_m} A_n(\vec{\Omega}) \sum_{p=1}^{N_p} P_p(\tilde{z}_r)\cdot q_{r,p}^n \tag{8}$$

or in the more compact form:

$$\boxed{q(\vec{r},\vec{\Omega}) = \vec{\mathcal{Z}}(\tilde{z},\vec{\Omega})\cdot \vec{\boldsymbol{q}}_r} \tag{9}$$

where:

$$\begin{aligned}\vec{\mathcal{Z}}(\tilde{z},\vec{\Omega}) &= \{A^0(\vec{\Omega})P_0(\tilde{z}), A^1(\vec{\Omega})P_0(\tilde{z}), ...,\\ &\quad A^0(\vec{\Omega})P_1(\tilde{z}), A^1(\vec{\Omega})P_1(\tilde{z}), ...\}\end{aligned} \tag{10}$$

$$\vec{\boldsymbol{q}} = \{q_p^n\} = \{\underbrace{q_0^0, q_0^1, q_0^2, ...}_{p=0}, \underbrace{q_1^0, q_1^1, q_1^2, ...}_{p=1}, \underbrace{q_2^0, q_2^1, q_2^2, ...}_{p=2}, ...\}$$

$\vec{\mathcal{Z}}$ and $\vec{\boldsymbol{q}}_r$ have dimensions $N_p \times N_m$ and $q_{r,p}^n$ can be directly retrieved from Eqs. (5) and (6).

Another useful relation used in the next section for the transmission equation later can be obtained recalling Eq. (8) and inverting the sums order:

$$\begin{aligned}q(\vec{r},\vec{\Omega}) &= \sum_{n}^{N_m} A_n(\vec{\Omega}) \sum_{p}^{N_p} P_p(\tilde{z}_r)\cdot q_{r,p}^n\\ &= \sum_{p}^{N_p} P_p(\tilde{z}_r)\cdot q_{r,p}(\vec{\Omega}) = \vec{P}(\tilde{z}_r)\cdot \vec{q}_r(\vec{\Omega})\end{aligned} \tag{11}$$

where:

$$q_{r,p}(\vec{\Omega}) = \sum_{n}^{N_m} A_n(\vec{\Omega})\cdot q_{r,p}^n$$

Transmission equation

The standard MOC transmission equation, coming from the integral form of the transport equation, reads:

$$\Psi(t^{out},\vec{\Omega}) = \Psi(t^{in},\vec{\Omega})\cdot e^{-\Sigma_r l} + \int_{t^{in}}^{t^{out}} dt'\; q\left(\vec{r}(t'),\vec{\Omega}\right)\, e^{-\Sigma_r(t^{out}-t')}$$

where t denotes the local coordinate along the trajectory, t^{in} and t^{out} represent the entering and exiting point of the trajectory inside a region and l is the chord length. Fig. 5 gives a visual interpretation of the classical trajectory-based domain discretization used in the Method Of Characteristics for a small 2D domain, as well as the meaning of entering, exiting fluxes and trajectory t.

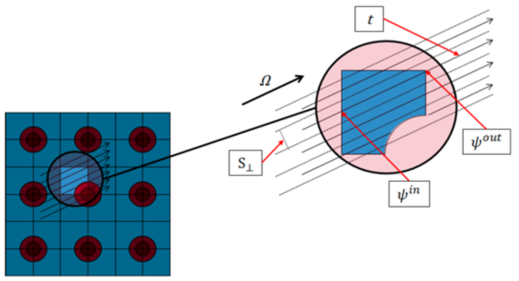

Fig. 5: Example of a 2D tracking.

The source term of the transmission equation can be expanded using Eq.(11), giving a numerical transmission equation for the polynomial method:

$$\Psi(t^{out},\vec{\Omega}) = \Psi(t^{in},\vec{\Omega})\cdot e^{-\Sigma_r l} + \vec{P}(\tilde{z}^{in})\cdot \vec{T} \tag{12}$$

where $\vec{P}(\tilde{z}^{in})$ is the values of the polynomial basis at the entering point of the trajectory inside a given region and the vector $\vec{T}$ reads:

$$T_k = \sum_{p=k}^{N_p} \mathbf{F}_{p,k} \frac{\left(\vec{q}_r(\vec{\Omega})\right)_p}{\Sigma_r} \tag{13}$$

$$\mathbf{F}_{p,k} = c_{p,k}\, \mu^{p-k} \left(\frac{2}{\Delta z_r}\right)^{p-k} E_{p-k}(\tau) \tag{14}$$

$$E_{p-k}(\tau) = \frac{1}{\Sigma_r^{(p-k)}} \int_{\tau(t^{in})}^{\tau(t^{out})} d\tau' \tau'^{p-k} e^{(\tau'-\tau(t^{out}))} \tag{15}$$

$c_{p,k}$ is the binomial coefficient, $\mu = cos(\theta)$ (where θ is the polar component of $\vec{\Omega}$) and $\tau = \Sigma_r l$ is the optical length.

The *escape coefficient* $E_{p-k}(\tau)$ of can be computed directly integrating Eq.(15). However integrating by parts this equation a useful recursive relation can be obtained, that allows to compute only the 0_{th} order coefficient and to retrieve the others:

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

$$E_p(\tau) = l^p - \frac{p}{\Sigma_r} E_{p-1}(\tau) \qquad 1 \le p \le N_p$$

A. Importance of the Chords Classification method

The Chords Classification method is a particular procedure that helps in strongly reducing the number of floating point operations that have to be performed during the transport sweep. It was introduced in [3], in association with the *Hit Surface Sequence* storage technique (HSS). The HSS allows to avoid the storage of the complete 3D tracking, exploiting the regularities of 3D extruded geometries. In [6] the authors propose an alternative strategy that allows an even bigger memory reduction, always exploiting big regularities typical of reactors geometries.

The sweep is one of the most computationally demanding part of the MOC, since it requires to sequentially solve the transmission equation for each chords on a trajectory, in order to compute the entering and exiting flux (Eq.(12)) and cumulate the term needed later for the balance equation (Eq.(19)). It is easy to see that the term $\vec{T}$ of Eq.(12) hide a big computational effort, as Eqs. from (13) to (15) show. The lowest *escape coefficient* can be pre-tabulated once and for all and the other can be computed from the first one. This operation can be performed *on-the-fly*. If this approach is followed the *escape coefficients* are computed the during the trajectory sweep for each chord. Once $E_p(\tau)$ is known, Eqs. (13) and (14) are applied to compute $\vec{T}$.

This strategy is clearly not efficient, but the Chords Classification method is particularly helpful in this situation. The basic idea of the method is that in a 3D extruded geometry typical of a nuclear reactor a lot of chords have the same lengths and belong to the same material. This means that they all share the same *optical length*. We will say that all these chords belong to the same *class*.

In particular, all *3D* chords parallel to a certain angle, sharing the same *2D* projection, intersecting two vertical surfaces and that belong to the same region will belong to the same *class*. The same thing applies to all chords intersecting two successive horizontal surfaces, belonging to the same region and parallel to a certain direction. We will call *V-classes* the first type and *H-classes* the second one. We refers to [3] for a detailed explanation of the method.

However the fundamental part is that thanks to the Chords Classification method is possible to compute the values of the $\vec{T}$ vector in Eq.(12) only for the *classes*, without needing to compute it for each chord. Each term that composes $\vec{T}$ depends, in fact, only on the *optical length*, the source coefficients, the region height and the trajectory angle. Moreover, since using the polynomial method big axial meshes are sufficient to approximate the axial behaviour of the flux, a small amount of *classes* is able to be representative of a huge part of the 3D chords present.

This means that the $\vec{T}$ vector can be computed only for a small percentage of the total number of chords. This is the reason why Eq.(12) is written in this form: during the transport sweep the only operation left to perform for each chords is the product with the value of the polynomial at the trajectory entering point $\vec{P}(\tilde{z}^{in})$, while $\vec{T}$ is pre-computed with a small (even if not negligible) computational cost.

Angular balance equation

An angular balance equation can be obtained to compute the moments of the angular flux defined as follows:

$$ {}'\vec{\Psi}_r(\vec{\Omega}) = \frac{1}{V_r}\int_r d\vec{r}\; \vec{P}(\tilde{z})\Psi(\vec{r},\vec{\Omega}) \tag{16}$$

Multiplying by $\vec{P}(\tilde{z})$ and integrating over the volume of each region the transport equation (1) we get:

$$\frac{1}{V_r}\int_r \vec{dr}\vec{P}(\tilde{z})\left(\vec{\Omega}\nabla\Psi(\vec{r},\vec{\Omega})\right) + \frac{\Sigma_r}{V_r}\int_r \vec{dr}\vec{P}(\tilde{z})\Psi(\vec{r},\vec{\Omega}) = \frac{1}{V_r}\int_r \vec{dr}\vec{P}(\tilde{z})\; q(\vec{r},\vec{\Omega})$$

With some adjustments we get a balance equation to compute the angular flux moments of Eq.(16):

$$\Sigma_r\; {}'\vec{\Psi}_r(\vec{\Omega}) = \bar{\bar{P}}P(\vec{\Omega})\cdot\vec{q}_r(\vec{\Omega}) - \Delta\vec{J}_r(\vec{\Omega}) + \mu\;\bar{\bar{C}}\;{}'\vec{\Psi}_r(\vec{\Omega}) \tag{17}$$

Going through the different terms of this equation separately we have:

$$\bar{\bar{P}}P(\vec{\Omega}) = \frac{1}{V_r}\int_r d\vec{r}\;\vec{P}(\tilde{z})\otimes\vec{P}(\tilde{z}) \tag{18}$$

$\bar{\bar{P}}P(\vec{\Omega})$ is a matrix with dimension N_p^2 and has an analytic and a numerical version. The analytic one reads:

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

The numerical counterpart is obtained expressing the axial coordinate as a function of the local coordinate along the trajectory, and numerically integrating Eq.(18) coherently with the trajectory-based spatial discretization.

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

$$\begin{aligned}\bar{\bar{P}}P_{(pp')} &= \frac{1}{V_r}\int_{\partial r} d_2r_\perp\int_0^L dt\left[\frac{z_r^{in}-\bar{z}+\mu\, t}{\Delta z_r/2}\right]^p\otimes\left[\frac{z_r^{in}-\bar{z}+\mu\, t}{\Delta z_r/2}\right]^{p'}\\ &= \sum_{k,k'=0}^{p,p'} c_{pk}\cdot c_{p'k'}\cdot\frac{\Delta_t}{V_r}\sum_{\substack{t\|\vec{\Omega}\\ t\cap r}}\left[\frac{z_r^{in}-\bar{z}}{\Delta z_r/2}\right]^{p+p'-k-k'}\left(\frac{2\mu}{\Delta z}\right)^{k+k'}\frac{l^{(k+k'+1)}}{k+k'+1}\end{aligned}$$

where z_r^{in} is the z coordinate of the entering point of the trajectory in the r_{th} region. The numerical version of $\bar{\bar{P}}P(\vec{\Omega})$ is a region and angle dependent matrix.

In order to have a conservative system the numerical approach has to be applied to compute this matrix. In a first realization of our method we didn't realized this [7] and the results where often affected by big numerical instabilities, especially when the numerical integration was far from the analytical one. Later, we found this very important detail explained in [8]. This lead a radical change in the method performances.

The remaining term of the balance equation (17) are:

$$\Delta\vec{J}_r(\vec{\Omega}) = \frac{\Delta_\perp(\vec{\Omega})}{V_r}\sum_{\substack{t\|\vec{\Omega}\\ t\cap r}}\left[\vec{P}(\tilde{z}^{out})\Psi_r(t^{out}) - \vec{P}(\tilde{z}^{in})\Psi_r(t^{in})\right] \quad (19)$$

$$\bar{\bar{C}} = diag\left[\frac{p}{\Delta z/2}\right]\cdot\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\\\vdots&&\ddots\end{bmatrix} = \begin{bmatrix}0&0&0\\\frac{1}{\Delta z/2}&0&0\\0&\frac{2}{\Delta z/2}&0\\\vdots&&\ddots\end{bmatrix}$$

$\Delta\vec{J}_r(\vec{\Omega})$ is computed during the transport sweep using the transmission equation (Eq.(12)) and $\Delta_\perp(\vec{\Omega})$ is the integration weight, corresponding to the perpendicular area associated to the trajectory. The $\bar{\bar{C}}$ is an under-diagonal matrix and thanks to this profile is possible to solve Eq.(17) as a lower triangular system.

Angular flux coefficients and flux moments computation

Once the angular flux moments are obtained, we can retrieve the corresponding coefficient of the expansion, combining Eq.(16) with Eq.(4) and obtaining:

$$\begin{aligned}{}'\vec{\Psi}_r(\vec{\Omega}) &= \frac{1}{V_r}\int_r d\vec{r}\,\vec{P}(\tilde{z})\otimes\vec{P}(\tilde{z})\cdot\vec{\Psi}_r(\vec{\Omega})\\ \vec{\Psi}_r(\vec{\Omega}) &= \bar{\bar{P}}P^{-1}\cdot\ {}'\vec{\Psi}_r(\vec{\Omega})\end{aligned} \quad (20)$$

To close the system we must compute the flux angular moments of Eq.(7):

$$\Phi^n_{r,p} = \oint\frac{d\vec{\Omega}}{4\pi}\,A_n(\vec{\Omega})\,\Psi_{r,p}(\vec{\Omega}) \quad (21)$$

IV. COMPARISON WITH THE SC METHOD

We present here a comparison between the SC and the Polynomial method. The two methods should be able to give a similar solution in terms of precision, since the basic idea is the same. Moreover the spatial approximation in the radial plane is exactly the same for the two methods. On the other hand the different approach for the axial flux approximation should results in a different number of meshes needed for the two methods, but in a similar result when the axial discretization is converged.

Disadvantages of the Polynomial method

Before presenting the results we want to underline some important differences in terms of computational cost between the two methods. The transmission equation (Eq.(12)) is one of the most important in the method of characteristics framework. It is solved during the transport sweep for each 3D chords of the system. Since the number of 3D chords is easily of the order of $10^7 - 10^8$ for classical 3D assemblies, the computational cost of the transmission equation is a very important parameter, when analysing solver performances. Moreover, during the transmission phase the current information is stored (Eq.(19)), to be later used in the balance equation. Comparing the transmission equation and the current term for the Polynomial and the SC method we get:

$$Poly \rightarrow \begin{cases}\Psi(t^{out},\vec{\Omega}) = \Psi(t^{in},\vec{\Omega})\cdot e^{-\Sigma_r l} + \vec{P}(\tilde{z}^{in})\cdot\vec{T}\\ \vec{\delta}_r(\vec{\Omega}) = \sum\limits_{\substack{t\|\vec{\Omega}\\ t\cap r}}\left[\vec{P}(\tilde{z}^{out})\Psi_r(t^{out}) - \vec{P}(\tilde{z}^{in})\Psi_r(t^{in})\right]\end{cases} \quad (22)$$

$$Step \rightarrow \begin{cases}\Psi_r(t^{out}) - \Psi_r(t^{in}) = \left(\frac{1-e^{-\Sigma_r l}}{\Sigma_r}\right)\cdot\left(q_r(\vec{\Omega}) - \Psi_r(t^{in})\right)\\ \delta_r(\vec{\Omega}) = \sum\limits_{\substack{t\|\vec{\Omega}\\ t\cap r}}\left[\Psi_r(t^{out}) - \Psi_r(t^{in})\right]\end{cases} \quad (23)$$

where we can assume for a simplified comparison that the exponential terms are already computed when the sweep is performed, thanks to the Chords Classification method [3], as well as the $\vec{T}$ vector.

Comparing the two sets of Eqs.(22) and (23) we see similarities and differences. They both requires a subtraction and a quite expensive memory access for the δ_r term. On the other hand the number of floating point operations(FLOPS) that they require is quite different: Eqs.(22) shows that $2\cdot N_p + 1$ FLOPS are required for the polynomial method (3 for the transmission and 2 for δ_r). On the other hand the SC method roughly needs only 1 FLOPS. Supposing a polynomial of second order this would easily lead to 5 FLOPS against 1 per chords in the SC method. The computational effort required by the Polynomial method is therefore clearly higher, as we would expect for a higher order scheme.

B. Vectorization possibilities

The Polynomial method is well suited for some vectorization options, that allows to partially compensate for the computational overhead just described in the above section. Looking at the set of Eqs. (22) we see that the trajectory sweep must be performed sequentially, since the entering flux for each chord is needed. This is a common fact of the MOC. Nevertheless, some of the operations of Eqs.(22) can be treated in a vectorial manner, in particular those which do not depend on the entering flux value.

In a first phase the product $\vec{P}(z^{in})\cdot\vec{T}$ is performed as long as the trajectory is reconstructed, through the HSS method. Here the product can be vectorized on the number of subsequent surfaces of the same kind encountered (mostly vertical since the axial meshes are quite elongated). To do this the value of $\vec{P}(z^{in})$ must be available for each chord of the sequence. This can be obtained through direct storing and retrieving, or on-line computing, depending on the chosen option. In both cases at the end of the HSS $\vec{P}(z^{in})$ will be available for each chord of the concerned trajectory.

In a second phase the transmission equation is solved and the entering fluxes values temporarily stored in some auxiliary quantities. Here no vectorization is possible. As a result two vectors are available:

$$\vec{\Psi}^{f/b} = \left\{\Psi^{f/b}_i,\ i = 1, N(t)+1\right\} \quad (24)$$

where N(t) is the number of chords of a given "t" trajectory and "f/b" stands for forward and backward directions.

Then, the term $\vec{\delta}_r(\vec{\Omega})$ must be computed and cumulated. Using some additional auxiliary vectors, its computation can be vectorized. For each chord we should write:

$$\begin{aligned} \gamma_{i,p}^{in}(t) &= P_p(\tilde{z}_i^{in})\Psi_i \\ \gamma_{i,p}^{out}(t) &= P_p(\tilde{z}_i^{out})\Psi_{i+1} \qquad\qquad i = 1,\ N(t) \\ \tilde{\delta}_{i,p} &= \gamma_{i,p}^{out} - \gamma_{i,p}^{in} \end{aligned} \tag{25}$$

However it is worth noting that thanks to choice of the polynomial basis (Eq. (3)) for every even moments:

$$P_p(\tilde{z}_{i+1}^{in}) = P_p(\tilde{z}_i^{out})$$

Unfortunately this is not true for odd terms. In this case the only difference arises when a horizontal surface is crossed and the value of the polynomial basis switches from -1 to +1 or vice-versa. As a consequence Eqs.(25) becomes:

$$\begin{aligned} \gamma_{i,p} &= P_p(\tilde{z}_i^{in})\Psi_i && i = 1,\ N(t)+1 \\ \tilde{\delta}_{i,p} &= \gamma_{i+1,p} - \gamma_{i,p} && for\ even\ terms \\ \tilde{\delta}_{i,p} &= \gamma_{i+1,p} - s_i \cdot \gamma_{i,p} && for\ odd\ terms \end{aligned}$$

where s_i a two-bit information stored for each chords, and it is equal to -1 if the i-th chord crosses a horizontal surface and equal to +1 otherwise. All these operations can be performed vectorially, for each chords of the trajectory t.

Finally the cumulation must be performed as is Eq.(22):

$$\delta_{r,p}(\vec{\Omega}) = \sum_{\substack{t\|\vec{\Omega} \\ t\cap r}} \left[P_p(\tilde{z}^{out})\Psi_r(t^{out}) - P_p(\tilde{z}^{in})\Psi_r(t^{in}) \right] = \sum_{\substack{t\|\vec{\Omega} \\ t\cap r}} \tilde{\delta}_{i,p}$$

and it is practically implemented as:

$$\delta_{r(i),p}(\vec{\Omega}) = \delta_{r(i),p}(\vec{\Omega}) + \tilde{\delta}_{i,p} \tag{26}$$

In the last formula r(i) denotes the region of the i-th intersection. Even if the cumulation phase has only a sum operation, the memory accesses that it requires is very expensive. Unfortunately, the same region can be intersected more than once by one trajectory and for a given angle. For this reason vectorization of this phase has not be implemented.

Advantages of the Polynomial method

Thanks to the polynomial approximation the number of axial meshes need is greatly reduced. This lead to very elongated regions, with a predominance of chords going from vertical to vertical surfaces. This also mean that there are more chances for a trajectory to be interrupted by a horizontal plane, which implies a lower number of 3D chords. This reduction in terms of total number of chords in comparison with the SC method depends on the study case, but we observed values around $\sim 10-15\%$. The second, and more important advantage, lies in the number of unknowns needed in the two cases. Decreasing the number of axial plane directly decreases the multi-group fluxes sizes, as well as the size of a set of working variables such as q_r and δ_r. Of course all these elements will have as many components as the chosen polynomial degree. The memory gain factor will then be:

$$\frac{N_z^{step}}{N_z^{pol} \cdot N_p}$$

where N_z^{step} and N_z^{pol} are the numbers of axial meshes need using the SC and in the Polynomial method, respectively. In addition to decrease the total memory needed, it also has a direct impact of computational time since this is strongly influenced by the size of the objects we are working with.

Iterative schemes

A classical iterative scheme, described in Algorithm 1, is used to converge to the solution. As customary this is divided in three steps: a initial guess is made for the fission integral and for the fluxes. Keeping the fission integral constant we iterate on the flux of each group updating the scattering term. Another subdivision is made: keeping the scattering from all the other group to the g groups constant we iterate on the self-scattering source for the group g. These are generally called *inners iterations*. Once the fluxes in the g group are converged, we update the transfer from others groups (*thermal iterations*). Finally when the transfer integral has converged, the fission source is updated. These iterations are generally called *external*. The procedure is repeated until convergence is reached.

An *OpenMP* based parallel strategy is implemented to speed-up the transport sweep phase, which is one of the most time consuming part. This part is indicated in Algorithm 1 as *Paralled trajectories sweep*. While the sweep of one trajectory must be sequential, different trajectories can be swept in parallel. A series of precautions must be adopted to avoid race condition and to equally distribute work among threads. The same strategy presented in presented in [3] for the SC method is also used in the Polynomial case. Algorithm 2 gives a simplified view of how the parallel sweep is handled and how the classic sweep sequence is modified to implement the vectorization explained in subsection B.. As the algorithm shows, there are private copies of the δ_r array for each thread. A certain amount of trajectories is attributed to each thread(*TASK*). For each trajectory, only a part of the operations described in the Algorithm 2 are performed. In particular, reduction of the private δ_r on the shared counterpart δ_r^{sh} is operated only when the basic angle has changed from a trajectory to another. Moreover, computation of the coefficients for classified chords T (Eq.(13)) and β is needed only if the basic angle or the 2D line on which the 3D trajectory lies has changed.

From these considerations we can see how a proper load balancing technique is important and can help in considering computational time reductions. The basic idea is to sort trajectories in order to minimize the basic angle change. Trajectories are then subdivided in *TASKS*. A greedy algorithm is then used to equally distribute work between tasks. For the details about the load balancing techniques used we recall [3][9], since no changes have been implemented.

```
External iterations:
while the fission term is not converged do
    Compute $q_{r,p}^{1,ext} = \chi^g \sum_{g'} \nu^{g'} \Sigma_f^{g'}(\vec{r})\, \Phi_{r,p}^{1,g'}$
            $q_{r,p}^{n,ext} = 0 \quad for \quad n > 1$
    Thermal iterations:
    while $\Phi_{r,p}^{1,g}$ not converged in each group do
        Compute $q_{r,p}^{n,ext} = q_{r,p}^{n,ext} + \sum_{g' \neq g} \Sigma_{s,n}^{g' \to g}(\vec{r})\, \Phi_{r,p}^{n,g'}$
        Internal iterations:
        while $\vec{\Phi}_r^{1,g}$ not converged in g group do
            Compute $q_{r,p}^{n} = q_{r,p}^{n,ext} = +\Sigma_{s,n}^{g \to g}(\vec{r})\, \Phi_{r,p}^{n,g}$
            Mono group solution

              Parallel trajectories sweep

            Balance (Eqs.(17),(20),(21))
            for every region r do
                $\Delta\vec{J}_r(\vec{\Omega}) = \frac{\Delta_\perp(\vec{\Omega})}{V_r}\vec{\delta}_r(\vec{\Omega})$
                $'\vec{\Psi}_r(\vec{\Omega}) =$
                $\frac{1}{\Sigma_r}\left[\bar{\bar{P}}P(\vec{\Omega}) \cdot \vec{q}_r(\vec{\Omega}) - \Delta\vec{J}_r(\vec{\Omega}) + \mu\, \bar{\bar{C}}\; '\vec{\Psi}_r(\vec{\Omega})\right]$
                $\vec{\Psi}_r(\vec{\Omega}) = \bar{\bar{P}}P^{-1} \cdot \; '\vec{\Psi}_r(\vec{\Omega})$
                $\vec{\Phi}_r^{n,g} = \oint \frac{d\vec{\Omega}}{4\pi} A_n(\vec{\Omega})\, \vec{\Psi}_r(\vec{\Omega})$
            end
        end
    end
end
```

Algorithm 1: Iterative scheme

```
Parallel trajectories sweep:
$\vec{\delta}_r^{sh}(\vec{\Omega}) = 0 \to$ initialize shared current vector
!$OMP PARALLEL
$\vec{\delta}_r(\vec{\Omega}) = 0 \to$ initialize private current vector
while Every trajectory is not swept do
    ThreadNum= $OMP\_GET\_THREAD\_NUM$
    Task= $GET\_TASK(ThreadNum)$
    for all trajectories in Task do
        if Basic angle has changed then
            Reduce on $\vec{\delta}_r^{sh}(\vec{\Omega})$
            $\vec{\delta}_r^{sh}(\vec{\Omega}) = \vec{\delta}_r^{sh}(\vec{\Omega}) + \vec{\delta}_r(\vec{\Omega})$
            $\vec{\delta}_r(\vec{\Omega}) = 0$
        end
        if 2D line or Basic angle has changed then
            Classified coefficient computation
              For each class:
              Compute $\vec{T} \quad \to$ Eq. (13)
              Compute $\beta = e^{-\Sigma_r l}$
        end
        HSS trajectory reconstruction
        for i=1, N(t) do
            if classed chord then
                Retrieve $\beta, \vec{T}^{f/b}$
            else
                Compute $\beta, \vec{T}^{f/b}$
            end
            Compute $\vec{P}(z^{out})$
            Store $\beta_i, \vec{P}_{i+1} = \vec{P}(z^{out})$
            Compute and Store:
              $\tilde{T}_i^f = \vec{P}_i \cdot \vec{T}^f$
              $\tilde{T}_i^b = s_i \cdot \vec{P}_{i+1} \cdot \vec{T}^b$
        end
        Trajectory sweep (sequential)
        for i=1, N(t) do
            $\Psi_{i+1}^{f/b} = \beta_i \Psi_i^{f/b} + \tilde{T}_i^{f/b} \quad \to$ Eq.(12)
        end
        Compute $\tilde{\delta}_{i,p}$ (vectorial)
        for i=1, N(t) do
            $\gamma_{i,p}^f = s_i \cdot P_{i,p} \Psi_i^f$
            $\tilde{\delta}_{i,p}^f = \gamma_{i,p}^f - \gamma_{i+1,p}^f$              even terms
            $\tilde{\delta}_{i,p}^f = \gamma_{i,p}^f - s_{i+1}^f \cdot \gamma_{i+1,p}^f$    odd terms
            $\gamma_{i,p}^b = P_{i,p} \Psi_i^b$
            $\tilde{\delta}_{i,p}^b = \gamma_{i+1,p}^b - \gamma_{i,p}^b$              even terms
            $\tilde{\delta}_{i,p}^b = \gamma_{i+1,p}^b - s_i^b \cdot \gamma_{i,p}^b$      odd terms
        end
        Cumulate $\vec{\delta}_r(\vec{\Omega})$ (sequential)
        for i=1, N(t)+1 do
            $\vec{\delta}_{r(i)}(\vec{\Omega}) = \vec{\delta}_{r(i)}(\vec{\Omega}) + \vec{\tilde{\delta}}_i^{f/b} \quad \to$Eq. (26)
        end
    end
end
!$OMP END PARALLEL
```

Algorithm 2: Inners iterative scheme

V. RESULTS AND ANALYSIS

Precision and performances of the polynomial method have been compared with the SC method and with a Monte Carlo simulation obtained with TRIPOLI4® [10]. Also in this case, the ASTRID assembly presented in Fig.2 has been used for the calculation. This is the same case presented in [4].

The multi-group cross sections have been self-shielded with the Subgroup and Tone methods of APOLLO3®. The self-shielding procedure and the 2D-3D equivalence applied is explained in [9][2][3]. In a few word the self-shielding is performed only in 2D sections of the domain. The number of axial planes on which perform a different self-shielding calculation is a input parameter and can be different in comparison with the axial meshes needed for the MOC solution. In our calculation only one self-shielding layer is computed for each different material. Radially, different zones can be defined in order to account for the impact of the heterogeneous flux on the self-shielded cross section, even when the same material is present. An internal and an external layer are used, as depicted in Fig.2. A 1968 groups discretization of the energy is used for the multi-group calculation.

Table I shows two results obtained with converged parameters, one using the SC and the second with the Polynomial method. The meaning of this table in only to show that with converged angular and spatial integration and axial meshes both the SC and the Polynomial method converge to a similar result, with a certain discrepancy in comparison with the Monte-Carlo reference calculation.
Table II shows the results in terms of reactivity and computational time for different SC and Polynomial calculation with different axial meshes. Results shows a discrete precision when compared to the Monte-Carlo simulation, with a error around +70/80 pcm of k_{eff}.

The presence of very different materials in the axial direction imply strong flux gradients, as Fig.4 shows. For this reason the SC method needs a quite big number of axial meshes to converge. The Polynomial method on the other side requires only a few more meshes than the one already present due to the five different materials. The zone where a subdivision is more important is the neutronic protection, where the gradients are very strong (Fig.4) . Thanks to a significant lower amount of axial meshes needed the Polynomial method also enjoys a lower number of chords and a very high classification rate.

Thanks to an important classification rate, the expensive transmission coefficients of Eq. (13) can be computed only for the *classes*, instead of for the total 3D chords, as explained in section A., and thanks to a high classification the classes will be representative of a percentage of the total number of chords which is referred to in table II as *Classification*.

VI. CONCLUSIONS

A polynomial axial expansion of the angular fluxes has been implemented for the 3D Method of Characteristics in the TDT module of APOLLO3®. Results show that the Polynomial and the SC methods converge to a similar solution, discretely close to the Monte Carlo simulation, in terms of k_{eff}. Reactions rates have not been compared yet, but they will.

Method	**Step**	**Polynomial (N_p=2)**
Δr (cm)	0.04	0.04
Δs (cm)	0.2	0.2
Axial meshes	**600**	**19**
# chords	510.92 M	270.50 M
# classes	152.98 M	13.94 M
Classification	53.74 %	94.21 %
Self-shielding	**Tone:**	**Tone:**
K_{eff}	1.165749	1.165782
ρ err/T4 (pcm)	**+50.16**	**+52.42**

TABLE I: First comparison between Step and Polynomial method. k_{eff} and reactivity (ρ) error are presented and compared to the Monte-Carlo reference value. M stands for millions. The Tone self-shielding method hase been used. Since some discrepancies are present between the SC, the Polynomial and the Monte-Carlo simulation these two calculation have been run with very strict integration parameters, and with a better angular quadrature formula in comparison with the results presented in Tab.II. These two first results do not intend to compare performances, but they only want to show which is the results obtained with converged spatial and angular integration and with a very refined axial meshes.

The tracking size for the Polynomial case, thanks to a lower amount of meshes, is smaller and the percentage of classified chords is bigger. The computational cost per chord of the Polynomial method results higher when compared to the SC method. Nevertheless results shows that the overall computational time is lower using the Polynomial method. This represents a significant advantage when comparing to the results obtained in [7]. This is due to some performance-enhancing recently realized in the Polynomial method, mainly focused on the vectorization of some of the sweep operations that is not possible in the SC method (see Sec.B.). Moreover the case presented here is bigger than the one presented in [7] and the flux gradients are steeper, so the memory gain coming from the use of a less refined axial mesh strongly advantages the Polynomial method.

The Polynomial method currently lacks a suitable acceleration technique, but an adapted version of the DP_N acceleration developed for the SC method is under construction. This is the reason why the results presented are run without the acceleration option, even for the SC method.

ACKNOWLEDGMENTS

The authors would like to acknowledge Electricité de France (EDF) for their long term partnership and support in the development of the APOLLO3® nuclear reactor analysis code. We thanks J.F. Vidal, J.M. Palau and P. Archier for having provided the 3D FBR assembly case that have been used to benchmark our developments.

Method	**Step**					**Polynomial** (N_p**=2)**		
Δr (cm)	0.05					0.05		
Δs (cm)	1.0					1.0		
Axial meshes	**57**	**110**	**180**	**257**	**600**	**5**	**6**	**12**
# chords	29.57 M	31.88 M	34.93 M	38.29 M	53.24 M	27.31 M	27.35 M	27.61 M
# classes	17.07 M	21.12 M	21.80 M	20.33 M	18.16 M	1.90 M	2.28 M	4.56 M
Classification	83.82 %	74.67 %	66.68 %	59.56 %	52.39 %	98.4 %	98.09 %	96.21 %
Self-shielding	**Tone:**					**Tone:**		
K_{eff}	1.163761	1.165075	1.165412	1.165672	1.165744	1.165584	1.165805	1.165889
ρ **err/T4 (pcm)**	**-96.45**	**+0.44**	**+25.43**	**+44.66**	**+49.72**	**+37.76**	**+54.00**	**+60.30**
Time	12 210 s	22 719 s	37 785 s	55 472 s	127 809 s	15 384 s	16 326 s	23 900 s
Self-shielding	**Sub-Groups:**					**Sub-Groups:**		
K_{eff}	1.164114	1.16543	1.165767	1.166027	1.166094	1.165927	1.166147	1.166243
ρ **err/T4 (pcm)**	**-70.54**	**+26.38**	**+ 51.22**	**+70.28**	**+75.42**	**+ 63.01**	**+ 79.27**	**+86.22**
Time	12 575 s	24 031 s	38 454 s	56 500 s	124 470 s	16 174 s	17 316 s	23431 s

TABLE II: Comparison between Step and Polynomial method. k_{eff} and reactivity (ρ) error are presented and compared to the Monte-Carlo reference value. Difference in the results caused by the use of the two different self-shielding option available in APOLLO3® are also presented. M stands for millions. The *Classification* percentage shows the portion of the total number of *chords* that can are represented by the *classes*. In other words, computing the coefficients depending on the *optical length* for each *class*, we will represents a percentage of the total number of *chords* corresponding to the *classification* percentage.

REFERENCES

1. D. SCIANNANDRONE, S. SANTANDREA, and R. SANCHEZ, "Optimized tracking strategies for step MOC calculations in extruded 3D axial geometries," *Annals of Nuclear Energy*, **87**, 49–60 (2016).
2. 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," *Mathematics and Computations, Supercomputing in Nuclear Applications and Monte Carlo International Conference, M&C+SNA+MC 2015*, **3**, *October 2016* (2015).
3. 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. thesis, Université Paris-Sud (2015).
4. 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).
5. 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 C), Supercomputing in Nuclear Applications (SNA), and the Monte Carlo (MC) Method," Nashville, Tennessee (2015), Mc.
6. 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).
7. 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 (2016).
8. R. SANCHEZ, "Prospects in deterministic three-dimensional whole-core transport calculations," *Nuclear Engineering and Technology*, **44**, *5*, 113–150 (2012).
9. 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," *Accepted for publication in Nuclear Science and Engineering* (2017).
10. E. BRUN, F. DAMIAN, C. M. DIOP, E. DUMONTEIL, F. X. HUGOT, C. JOUANNE, Y. K. LEE, F. MALVAGI, A. MAZZOLO, O. PETIT, J. C. TRAMA, T. VISONNEAU, and A. ZOIA, "Tripoli-4®, CEA, EDF and AREVA reference Monte Carlo code," *Annals of Nuclear Energy*, **82**, 151–160 (2015).