International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)
Sun Valley, Idaho, USA, May 5-9, 2013, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2013)

MICADO : Parallel Implementation of a 2D–1D Iterative Algorithm for the 3D Neutron Transport Problem in Prismatic Geometries

FÉVOTTE François and LATHUILIÈRE Bruno
EDF R&D
francois.fevotte@edf.fr; bruno.lathuiliere@edf.fr

ABSTRACT

The large increase in computing power over the past few years now makes it possible to consider developing 3D full-core heterogeneous deterministic neutron transport solvers for reference calculations. Among all approaches presented in the literature, the method first introduced in [1] seems very promising. It consists in iterating over resolutions of 2D and 1D MOC problems by taking advantage of prismatic geometries without introducing approximations of a low order operator such as diffusion. However, before developing a solver with all industrial options at EDF, several points needed to be clarified. In this work, we first prove the convergence of this iterative process, under some assumptions. We then present our high-performance, parallel implementation of this algorithm in the MICADO solver. Benchmarking the solver against the Takeda case shows that the 2D–1D coupling algorithm does not seem to affect the spatial convergence order of the MOC solver. As for performance issues, our study shows that even though the data distribution is suited to the 2D solver part, the efficiency of the 1D part is sufficient to ensure a good parallel efficiency of the global algorithm. After this study, the main remaining difficulty implementation-wise is about the memory requirement of a vector used for initialization. An efficient acceleration operator will also need to be developed.

Key Words: 3D heterogeneous transport; 2D–1D coupling algorithm; method of characteristics; prismatic geometries; MICADO; parallel algorithm.

1. INTRODUCTION

The large increase in computing power over the past few years now makes it possible to consider developing 3D full-core heterogeneous deterministic neutron transport solvers for reference calculations. The method of characteristics (MOC) [2] is very popular in 2D as it can easily deal with complex geometries. However, for 3D problems the method of characteristics is still too expensive to run computations on whole core geometries due to the huge amount of memory requirement and/or computation time needed. As reported in [3], many studies have been published on these issues. A first approach [4] consists in reducing the amount of memory by increasing the polynomial order of the source approximation, which allows reducing the number of mesh cells in the spatial discretization. Other studies [5] have shown that memory requirements for the ray tracing could be optimized in lattice geometries consisting of a large number of basic geometric patterns. Finally, most reactor core geometries are prismatic, which allows computing 3D tracks inline and reducing the required storage to relatively small 2D tracking data [6]. All these methods mainly address the issue of memory requirement for a 3D MOC.

An other approach consists in avoiding direct 3D MOC computations by using a set of 2D MOC solvers fed by axial leakage source terms. These kinds of methods are called Fusion Method or 2D–1D coupling method. In code DeCART, axial leakage is computed by a global 3D CMFD diffusion problem [7–9] or by an SPN solver [10]. In the CHAPELET-3D [11], CRX [12,13] and AGENT [14] codes, after an homogenization at the scale of a pin (Fuel and moderator) 1D computation (Finite Difference, Nodal Expansion or

MOC) are performed to compute the leakage. However, all these methods introduce approximations, either by using a diffusion equation or by homogenizing cross sections. With the method described in [1] under the name *New formulation of the 2D/1D method*, axial leakage sources are computed with a 1D MOC solver without homogenization. As this method does not require diffusion nor simplified transport approximations, we think it is very promising for reference calculations and aim at assessing in this work the potential benefits and drawbacks of such a method in an industrial setting.

After providing a description of the method, this paper addresses in particular three main concerns: do the coupled 2D–1D iterations converge? Does the 2D–1D coupling affect the spatial convergence order of the MOC? Finally, can this method be efficiently implemented in parallel? All numerical results are based on the MICADO solver (Modern parallel Implementation of transport equation resolution based on Characteristic And Discrete Ordinate) developed at EDF in order to perform reference calculations.

2. THE 2D–1D ITERATIVE COUPLING METHOD

2.1. Construction of the 2D–1D iterative system

After discretization of the energetic and angular variables in the neutron transport problem, the spatial equation left to solve can be written as:

$$\varepsilon\frac{\partial\psi}{\partial x}(x,y,z)+\eta\frac{\partial\psi}{\partial y}(x,y,z)+\mu\frac{\partial\psi}{\partial z}(x,y,z)+\Sigma\psi(x,y,z)=Q(x,y,z), \tag{1}$$

where ε, η, μ are the three components of direction $\mathbf{\Omega}$ respectively along axes x, y and z. The angular flux at a given point is denoted by $\psi(x,y,z)$, Σ is the total macroscopic cross-section, and $Q(x,y,z)$ represents the source term including fission and scattering.

The radial and axial parts of this equation can be decoupled, which is especially interesting in cases where the geometry is prismatic. This can be done by reformulating equation (1) as the following set of coupled equations:

$$\begin{cases} \varepsilon\dfrac{\partial\psi_R}{\partial x}(x,y,z)+\eta\dfrac{\partial\psi_R}{\partial y}(x,y,z)+\Sigma\psi_R(x,y,z)=Q(x,y,z)-\mu\dfrac{\partial\psi_Z}{\partial z}(x,y,z), & \text{(2a)}\\ \mu\dfrac{\partial\psi_Z}{\partial z}(x,y,z)+\Sigma\psi_Z(x,y,z)=Q(x,y,z)-\varepsilon\dfrac{\partial\psi_R}{\partial x}(x,y,z)-\eta\dfrac{\partial\psi_R}{\partial y}(x,y,z), & \text{(2b)}\\ \psi_R=\psi_Z. & \text{(2c)} \end{cases}$$

In the system above, ψ_R and ψ_Z are the unknowns of the new system, and all other notations are consistent with equation (1). The first equation in this system is very similar to those proposed in [7–14]: for each fixed value of z, equation (2a) can be seen as a standard 2D neutron transport problem of unknown ψ_R, with a modified source term including axial leakage to couple planes. However, unlike most other codes, axial leakage terms are evaluated using a 1D axial transport problem. Indeed, for each fixed position (x,y), equation (2b) is a 1D transport ordinary differential equation of unknown ψ_Z, whose source term includes radial leakage terms coming from ψ_R. The third equation (2c) ensures that both the 1D and 2D equations are equivalent to the global 3D problem.

A natural step then consists in transforming the set of equations (2) into an iterative system (figure 1), in which the 2D and 1D equations are solved in turn, and axial and radial leakage source terms are computed

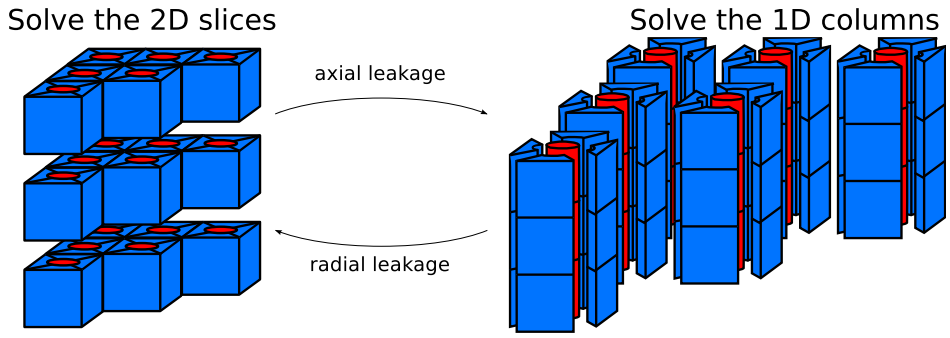


Figure 1. Alternating resolution on 2D slices and 1D columns

using the most recently computed values of ψ_Z and ψ_R respectively:

$$\begin{cases} \varepsilon \dfrac{\partial \psi_{R,n+1}}{\partial x}(x,y,z) + \eta \dfrac{\partial \psi_{R,n+1}}{\partial y}(x,y,z) + \Sigma \psi_{R,n+1}(x,y,z) = Q(x,y,z) - \mu \dfrac{\partial \psi_{Z,n}}{\partial z}(x,y,z), & \text{(3a)} \\ \mu \dfrac{\partial \psi_{Z,n+1}}{\partial z}(x,y,z) + \Sigma \psi_{Z,n+1}(x,y,z) = Q(x,y,z) - \varepsilon \dfrac{\partial \psi_{R,n+1}}{\partial x}(x,y,z) - \eta \dfrac{\partial \psi_{R,n+1}}{\partial y}(x,y,z), & \text{(3b)} \end{cases}$$

where n is the iteration index. We expect this system to converge to a fixed point with $\|\psi_{R,n} - \psi_{Z,n}\| \to 0$, which yields back equation (2c).

2.2. Discretization

In prismatic geometries, space can be discretized as the Cartesian product of a radial mesh and an axial one, very consistently with iterative formulation (3). In the following, let us denote by $z_j,\ j \in [\![1; N_j + 1]\!]$, the boundaries of the axial mesh cells. This mesh decomposes the entire geometric domain into N_j *slices* denoted as S_j, comprised between planes $z = z_j$ and $z = z_{j+1}$ and of thickness $h_j = |z_{j+1} - z_j|$. Equation (3a) can be integrated over the axial direction between z_j and z_{j+1} to yield

$$\varepsilon \frac{\partial \psi^j_{R,n+1}}{\partial x}(x,y) + \eta \frac{\partial \psi^j_{R,n+1}}{\partial y}(x,y) + \Sigma \psi^j_{R,n+1}(x,y) = Q^j(x,y) - Q^j_{Z,n}(x,y), \tag{4}$$

where $\psi^j_{R,n+1}$ is the axially averaged flux at iteration $n+1$ in slice S_j, Q^j is the axially averaged source term comprising scattering and fission, and $Q^j_{Z,n}$ is the axially averaged leakage source term computed using $\psi_{Z,n}$:

$$Q^j(x,y) = \frac{1}{h_j} \int_{z_j}^{z_{j+1}} Q(x,y,z)\, dz, \qquad Q^j_{Z,n}(x,y) = \frac{1}{h_j} \int_{z_j}^{z_{j+1}} \mu \frac{\partial \psi_{Z,n}}{\partial z}(x,y,z)\, dz.$$

For each slice, equation (4) is a standard 2D transport equation, which can be discretized using any adequate numerical scheme. In order to perform heterogeneous reference calculations, a method of characteristics with the Step Characteristics (SC) or Diamond Difference (DD) scheme has been chosen in our work. Let us denote by $C_i,\ i \in [\![1; N_i]\!]$, the N_i cells constituting the unstructured radial mesh, and define $|C_i|$ as the area of cell C_i. As usual in the MOC–SC sources are supposed to be constant within each cell:

$$Q^j(x,y) = \sum_{i=1}^{N_i} \mathbb{1}_{C_i}(x,y)\, Q^{i,j}, \qquad \text{with} \quad Q^{i,j} = \frac{1}{|C_i|\, h_j} \int_{(x,y)\in C_i} \int_{z_j}^{z_{j+1}} Q(x,y,z)\, dz\, dy\, dx. \tag{5}$$

In the equation above, $\mathbb{1}_{C_i}$ denotes the indicator function of radial cell C_i, and $Q^{i,j}$ is the fully averaged source term in cell C_i of slice S_j.

The fully discretized radial equation in slice S_j with cell-wise constant representation for the source terms therefore reads:

$$\varepsilon\frac{\partial\psi^j_{R,n+1}}{\partial x}(x,y)+\eta\frac{\partial\psi^j_{R,n+1}}{\partial y}(x,y)+\Sigma\psi^j_{R,n+1}(x,y)=\sum_{i=1}^{N_i}\mathbb{1}_{C_i}(x,y)\ (Q^{i,j}-Q^{i,j}_{Z,n}), \tag{6a}$$

where $Q^{i,j}_{Z,n}$ denotes the fully averaged axial leakage source term at iteration n in cell C_i and slice S_j, defined in the same way as $Q^{i,j}$.

The same discretization can be applied to the axial equation of the system: integrating equation (3b) over the radial directions (x,y) in cell C_i yields

$$\mu\frac{\partial\psi^i_{Z,n+1}}{\partial z}(z)+\Sigma\psi^i_{Z,n+1}(z)=\sum_{j=1}^{N_j}\mathbb{1}_{S_j}(z)\ \left(Q^{i,j}-Q^{i,j}_{R,n+1}\right), \tag{6b}$$

where $\psi^i_{Z,n+1}$ is the radially averaged flux at iteration $n+1$ in cell C_i, and the sources are represented using the same cell-wise flat approximation as in the radial equation. $Q^{i,j}_{R,n+1}$ denotes the averaged radial leakage source term, at iteration $n+1$ in cell C_i and slice S_j.

Using the discretized unknowns ψ^j_R and ψ^i_Z, averaged leakage source terms can be expressed as

$$\begin{aligned}
Q^{i,j}_R &= \frac{1}{|C_i|\ h_j}\int_{C_i}\int_{z_j}^{z_{j+1}}\left[\varepsilon\frac{\partial\psi_R}{\partial x}(x,y,z)+\eta\frac{\partial\psi_R}{\partial y}(x,y,z)\right]dz\ dy\ dx,\\
&= \frac{-1}{|C_i|\ h_j}\int_{z_j}^{z_{j+1}}\int_{\Gamma_i}(\varepsilon\, n_x+\eta\, n_y)\ \psi_R(x,y,z)\ d\Gamma_i\ dz, \qquad (7a)\\
&= \frac{-1}{|C_i|}\int_{\Gamma_i}(\varepsilon\, n_x+\eta\, n_y)\ \psi^j_R(x,y)\ d\Gamma_i,\\
\text{and}\quad Q^{i,j}_Z &= \frac{1}{|C_i|\ h_j}\int_{C_i}\int_{z_j}^{z_{j+1}}\mu\frac{\partial\psi_Z}{\partial z}(x,y,z)\ dz\ dy\ dx=\frac{\mu}{h_j}\left[\psi^i_Z(z_{j+1})-\psi^i_Z(z_j)\right], \qquad (7b)
\end{aligned}$$

where in the first equation Γ_i denotes the boundary of radial cell C_i, and (n_x,n_y) are the coordinates of the normal vector to this boundary. Iteration indices n were dropped in these equations for the sake of readability.

Another way of expressing averaged axial and radial source consists in averaging the radial and axial transport equations (6a) and (6b), to obtain balance equations

$$\begin{cases}
Q^{i,j}_{R,n+1}+\Sigma\psi^{i,j}_{R,n+1}=Q^{i,j}-Q^{i,j}_{Z,n}, & (8a)\\
Q^{i,j}_{Z,n+1}+\Sigma\psi^{i,j}_{Z,n+1}=Q^{i,j}-Q^{i,j}_{R,n+1}, & (8b)
\end{cases}$$

where $\psi^{i,j}_{R,n+1}$ and $\psi^{i,j}_{Z,n+1}$ are the radially and axially averaged values of $\psi_{R,n+1}$ and $\psi_{Z,n+1}$ in cell C_i and slice S_j.

2.3. Convergence of the iterative system

This paragraph studies the question of the convergence of iterative system (3) to the solution ψ of the complete 3D problem (1). After iteration n, the errors introduced by the iterative system can be defined as $e_{R,n} = \psi_{R,n} - \psi$ and $e_{Z,n} = \psi_{Z,n} - \psi$.

To prove convergence of the iterative system, our goal here is to study the limit of sequences $(e_{R,n})_{n\in\mathbb{N}}$ and $(e_{Z,n})_{n\in\mathbb{N}}$ as n tends to infinity. One can obtain the equations verified by $e_{R,n}$ and $e_{Z,n}$ by subtracting (1) to both equations of (3):

$$\begin{cases} \varepsilon \dfrac{\partial e_{R,n+1}}{\partial x}(x,y,z) + \eta \dfrac{\partial e_{R,n+1}}{\partial y}(x,y,z) + \Sigma e_{R,n+1}(x,y,z) = -Q_{Z,n}(x,y,z), \\ \mu \dfrac{\partial e_{Z,n+1}}{\partial z}(x,y,z) + \Sigma e_{Z,n+1}(x,y,z) = -Q_{R,n+1}(x,y,z), \end{cases}$$

where we defined $Q_{Z,n} = \mu \frac{\partial \psi_{Z,n}}{\partial z}$ and $Q_{R,n} = \varepsilon \frac{\partial \psi_{R,n}}{\partial x} + \eta \frac{\partial \psi_{R,n}}{\partial y}$. These equations are similar to those verified by the flux and can thus be discretized in the way described in paragraph 2.1:

$$\begin{cases} \varepsilon \dfrac{\partial e^j_{R,n+1}}{\partial R}(R) + \Sigma e^j_{R,n+1}(R) = -\displaystyle\sum_{i=1}^{N_i} \mathbb{1}_{C_i}(R)\; Q^{i,j}_{Z,n}, & \forall j \in [\![1, N_j]\!], \quad (9a) \\ \mu \dfrac{\partial e^i_{Z,n+1}}{\partial z}(z) + \Sigma e^i_{Z,n+1}(z) = -\displaystyle\sum_{j=1}^{N_j} \mathbb{1}_{S_j}(z)\; Q^{i,j}_{R,n+1}, & \forall i \in [\![1, N_i]\!], \quad (9b) \end{cases}$$

where the notations are consistent with equations (6a) and (6b). Most numerical methods used to solve such a set of equations can be expressed as a matricial operation similar to

$$\begin{cases} \bar{e}_{R,n+1} = -M_R\; \bar{Q}_{Z,n}, \\ \bar{e}_{Z,n+1} = -M_Z\; \bar{Q}_{R,n+1}, \end{cases}$$

where $\bar{e}_{R,n+1}$, $\bar{e}_{Z,n+1}$, $\bar{Q}_{Z,n}$ and $\bar{Q}_{R,n+1}$ are vectors respectively storing fully averaged coefficients $e^{i,j}_{R,n+1}$, $e^{i,j}_{Z,n+1}$, $Q^{i,j}_{Z,n}$ and $Q^{i,j}_{R,n+1}$, while M_R and M_Z are matrices representing the discretized inversion of radial and axial transport operators.

Furthermore, similarly to equations (8) for the flux, integrating the radial and axial transport above respectively over the radial and axial mesh cells yields the two following balance equations:

$$\begin{cases} \bar{Q}_{R,n+1} + \Sigma \bar{e}_{R,n+1} = -\bar{Q}_{Z,n}, \\ \bar{Q}_{Z,n+1} + \Sigma \bar{e}_{Z,n+1} = -\bar{Q}_{R,n+1}. \end{cases}$$

By combining the two discretized transport inversion equations and the two balance equations above, one can deduce the following recurrence relation:

$$\bar{e}_{Z,n+1} = M_Z\; (\Sigma M_R - I)\; (\Sigma M_Z - I)\; M_Z^{-1}\; \bar{e}_{Z,n} = M_Z^{it}\; \bar{e}_{Z,n},$$

where I denotes the identity matrix, and we defined the iteration matrix M_Z^{it} linking each axial error term to the next.

Using the method of characteristics to discretize system (3) yields matrices M_R and M_Z which are lower-triangular, under the assumption that all mesh cells are convex. More generally, this property would be verified by any tracking-based method preserving the notion of "flow" in the hyperbolic transport equation. During the solution of the 1D axial problem using MOC–SC-like equations, for each axial slice S_j the diagonal coefficient of matrix M_Z is

$$\frac{1}{\Sigma}\left(\frac{e^{-\Sigma h_j}-1}{\Sigma h_j}+1\right).$$

Therefore, the diagonal coefficients of matrix $\Sigma M_Z - I$ can be written as $\frac{e^{-\Sigma h_j}-1}{\Sigma h_j}$, which can be proven to be between -1 and 0 if $\Sigma h_j > 0$. Equivalent results could have been obtained for the DD scheme.

Following the same line of reasoning, similar results could have been obtained for the 2D radial MOC matrices. Since all matrices involved in the expression of M_Z^{it} are triangular, it can be concluded that $\rho(M_Z^{it}) < 1$, where $\rho(M)$ denotes the spectral radius of matrix M. This proves the convergence of iterative scheme (6). It is interesting to note that $\rho(M_Z^{it})$ approaches 1 when the mesh cells optical thickness is small. Slow convergence speeds will therefore be expected for finely discretized geometries or void regions.

2.4. Algorithm

The 2D–1D coupling scheme leads to algorithm 1, in which transport equations (6a) and (6b) are solved in turn and coupled using axial leakage source terms computed using balance equations (8a) and (7b). These equations are solved for each direction $\mathbf{\Omega}_k$ in the quadrature formula S_N. This scheme can easily be implemented using any 2D and 1D transport methods consistent with the cell-wise constant source representation of equation (5).

As the convergence of this loop is integrated in a complex inner-outer iterative algorithm (inverse power iteration, Gauss-Seidel for up scattering iterations, free or MR iterations for scattering), it is difficult to choose an appropriate stopping criteria. Based on our experience of neutronic solvers, we choose a fixed, small number of inner iterations and the outer algorithms are used to reach convergence. However, better stopping criteria based on $\|\psi_R - \psi_Z\|$ could be studied. To be effective, such stopping criteria would have to save a large number of iterations as they imply storing ψ_R and ψ_Z in two different vectors.

The output of the solver is the last computed value of ψ_Z, but axial leakage source terms Q_Z also need to be kept to initialize following calls to the 2D–1D iterative solver. Storing such a group- and direction-dependent vector drastically increases the memory requirement of the method. However, this drawback can be partially mitigated in parallel implementations of the algorithm, especially on distributed memory systems, in which this extra required memory can be shared between processes.

3. SPATIAL CONVERGENCE ANALYSIS

To perform the experimental spatial convergence analysis, the TAKEDA benchmark[15] is chosen. Indeed, as the physical sections are defined on a Cartesian mesh, it is easy to refine. With the C5G7[16] benchmark it would be more difficult for the finer mesh to respect the geometry of the fuel.

We used a Gauss–Legendre angular quadrature formula with 32 directions (8 azimuthal times 4 polar directions). The stopping criterion is fixed to $|\Delta k_{eff}| < 10^{-8}$, which leads to $\|\Delta S\phi\|_2 \simeq 10^{-6}$ at the end of

Algorithm 1: The 2D–1D coupling algorithm

while $\psi_R \neq \psi_Z$ **do**

$\triangleright$ *Solve 2D MOC equations*

forall $j \in [\![1, N_j]\!]$ **do**

forall $\mathbf{\Omega}_k \in S_N$ **do**

$$\int_{z_j}^{z_{j+1}} \left(\varepsilon_k \frac{\partial \psi_{k,R}}{\partial x} + \eta_k \frac{\partial \psi_{k,R}}{\partial y} + \Sigma \psi_{k,R} \right) dz = \int_{z_j}^{z_{j+1}} \left(Q - \mu_k \frac{\partial \psi_{k,Z}}{\partial z} \right) dz \; ;$$

$\triangleright$ *Solve 1D MOC equations*

forall $i \in [\![1, N_i]\!]$ **do**

forall $\mathbf{\Omega}_k \in S_N$ **do**

$$\int_{\mathcal{C}_i} \left(\mu \frac{\partial \psi_{k,Z}}{\partial z} + \Sigma \psi_{k,Z} \right) dxdy = \int_{\mathcal{C}_i} \left(Q - \varepsilon_k \frac{\partial \psi_{k,R}}{\partial x} - \eta_k \frac{\partial \psi_{k,R}}{\partial y} \right) dxdy \; ;$$

Algorithm 2: Basic sequential resolution of 1D MOC equation

forall $\mathbf{\Omega}_k \in S_N$ **do**

  forall $i \in [\![1, N_i]\!]$ **do**

    $\psi^- = 0$;

    for $j \in [\![1, N_j]\!]$ *with* j *increasing (resp. decreasing) for* $\mu_k > 0$ *(resp.* $\mu_k < 0$*)* **do**

      $\beta = e^{-h_j \Sigma[j,i]/\mu_k}$;

      $\psi^+ = \beta \psi^- + (1 - \beta) \dfrac{Q[j,i] - Q_R[k,j,i]}{\Sigma[j,i]}$;

      $Q_Z[k,j,i] = \dfrac{\mu_k \left(\psi^+ - \psi^- \right)}{h_j}$;

      $\psi_Z[k,j,i] = \dfrac{Q[j,i] - Q_R[k,j,i] - Q_Z[k,j,i]}{\Sigma[j,i]}$;

      $\psi^- = \psi^+$;

the computation. The reference computation is run on a well-converged $500 \times 500 \times 500$ spatial mesh. A comparison of these reference results with published benchmark results is presented on table I.

Figure 2 presents result differences between the reference calculation and computations on coarser meshes as a function of the mesh size. Scalar flux differences are computed using a L^2-norm on the fine reference mesh (*i.e.* fluxes obtained on coarser meshes are first projected on the $500 \times 500 \times 500$ spatial mesh before being compared to the reference flux). Excluding the last few points (which are too close to the reference calculation), the measured order of convergence for the flux is close to 1, as expected of a MOC–SC scheme.

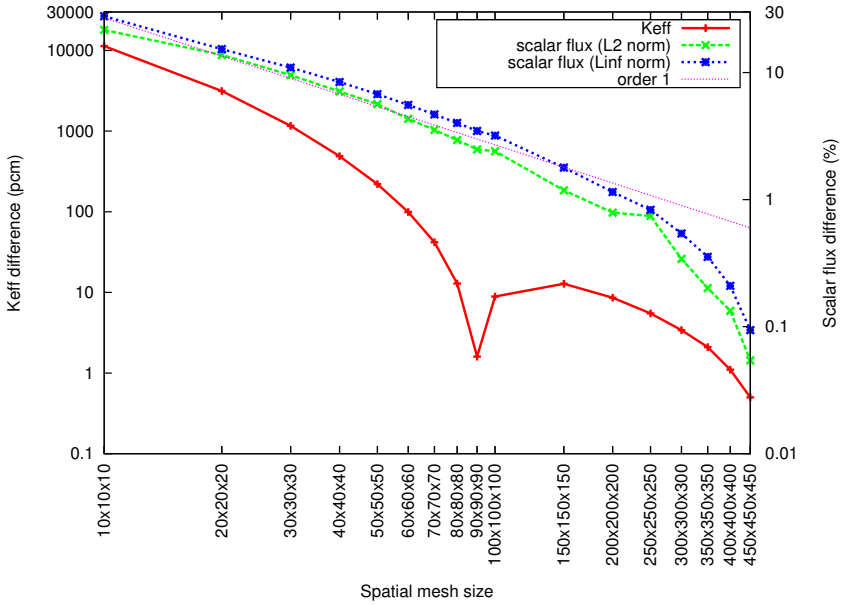


Figure 2. Spatial convergence of the solver

Table I. Difference between tabulated TAKEDA benchmark results and our reference calculation

	Difference	MCNP Error bar
k_{eff}	1.3 pcm	60 pcm
flux core (group 1 / group 2)	-0.38 % / 0.12 %	0.13 % / 0.10 %
flux reflector (group 1 / group 2)	0.32 % / 0.29 %	0.23 % / 0.21 %
flux control rod (group 1 / group 2)	1.12 % / 0.04 %	0.72 % / 0.48 %

4. HPC CAPABILITIES

4.1. Parallel algorithm in distributed memory

As 2D characteristic solvers are more difficult to parallelize and require more computational resources than 1D solvers, the data are distributed along radial slices. Hence the parallelization in a distributed environment (through MPI API) is pretty simple. As the data distribution is orthogonal to the direction of 1D characteristics solvers, the parallelization of this step of the algorithm is more complicated. That's why the parallel capabilities of the 2D–1D algorithm has to be assessed before developing a solver with all options of an industrial solver.

A basic sequential algorithm implementation of the 1D solver is based on the algorithm 2 for the SC scheme. With the first two loops (lines 1 and 2) the characteristics for all directions and columns are treated. Lines 3 to 9 correspond to the sweep of one characteristic : incoming fluxes are initialized line 3 with void boundary condition; then the loop over slices (line 4) takes care of the angular direction. The transmission equation is applied with lines 5, 6 and 9. Line 7 (specific to the 2D–1D coupling algorithm) computes the radial leakage. Finally line 8 the angular flux is computed. This vector is used to compute the scalar flux. Depending on the stopping criterion used in the 2D–1D coupling, an optimized implementation can avoid storing this vector.

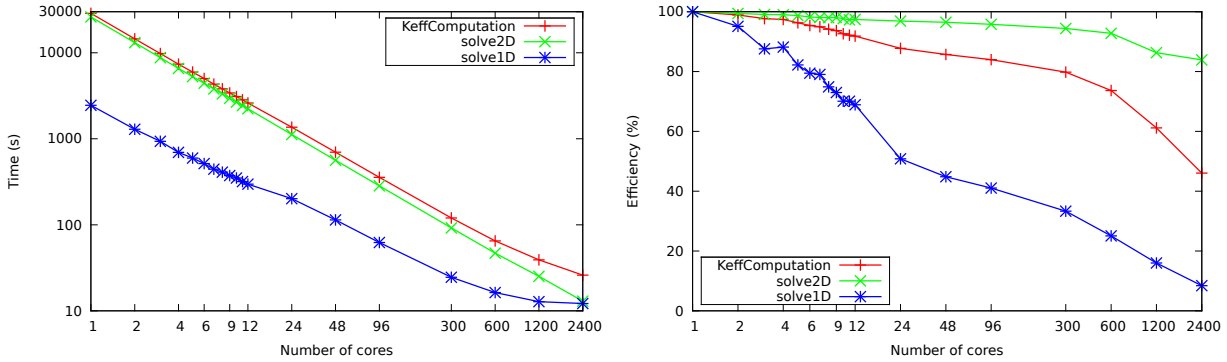
Algorithm 3 describes the parallelization of the 1D part for the SC scheme for the processor p which is responsible of the slice $jBeg$ to $jEnd$. First the communication of ψ^- are added (lines 7 and 17). As we want to avoid too large a number of communications, these communications are packed by groups of $blockSize$. Variable ψ^- thus becomes a vector of size $blockSize$ and the loop over Ni (line 2 of algorithm 2) is split into two loops (lines 3 and 9). The second loop (line 9) is under the loop over j (line 8) to benefit from the storage of the vectors which give a priority to the continuity into a slice. This optimization also improves the performances of the algorithm in sequential context (*i.e.* when $jBeg = 1$ and $jEnd = N_j$). The choice of $blockSize$ has an influence on the parallel efficiency. If $blockSize$ is too small there is too much communications and the latency of the network downgrades the performance. If $blockSize$ is too big the granularity of the pipeline is too big and the last (or the first depending on the sign of μ_k) processors are wasting time waiting for data. Complementary work is required to automatically compute an optimal $blockSize$ value. To avoid breaking the pipeline between each direction, the directions of loop 2 have to be sorted. First the directions with $\mu_k > 0$ have to be treated and then those with $\mu_k < 0$. The pipeline is therefore broken only once, when the sign of μ_k changes.

Algorithm 3: Parallel resolution of 1D MOC equation

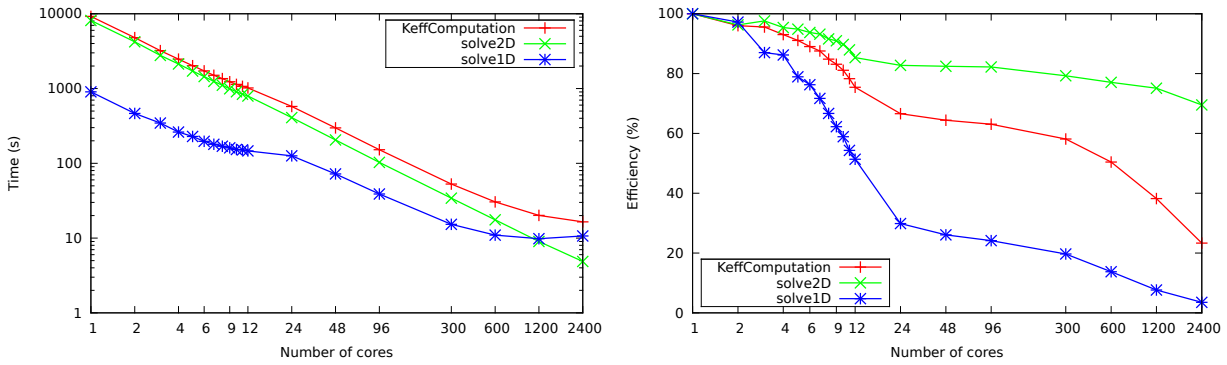
▷ *Executed by the processor p dealing with the slices* $[\![jBeg, jEnd]\!]$
2 **forall** $\mathbf{\Omega}_k \in S_N$ **do**
3 **forall** $blocI \in [\![1, \frac{Ni}{blockSize}]\!]$ **do**
 if *(*$jBeg = 1$ *and* $\mu_k > 0$ *) or (*$jEnd = N_j$ *and* $\mu_k < 0$*)* **then**
5 $\psi^- = 0$; ▷ *Vector of size* $blockSize$
 else
7 receiving ψ^- from processor $p - sign(\mu_k)$;
8 **for** $j \in [\![jBeg, jEnd]\!]$ *with j increasing (resp. decreasing) for* $\mu_k > 0$ *(resp.* $\mu_k < 0$*)* **do**
9 **forall** $bloci \in [\![1, blockSize]\!]$ **do**
 $i = bloci + (blocI - 1) * blockSize$;
 $\beta = e^{-h_j \Sigma[j,i]/\mu_k}$;
 $\psi^+ = \beta\psi^-[bloci] + (1-\beta)\dfrac{Q[j,i] - Q_R[k,j,i]}{\Sigma[j,i]}$;
 $Q_Z[k,j,i] = \dfrac{\psi^+ - \psi^-[bloci]}{\Delta_k^z}$;
 $\psi_Z[k,j,i] = \dfrac{Q[j,i] - Q_R[k,j,i] - Q_Z[k,j,i]}{\Sigma[j,i]}$;
15 $\psi^-[bloci] = \psi^+$;
 if *non((*$jEnd = N_j$ *and* $\mu_k > 0$*) or (jBeg=1 and* $\mu_k < 0$ *))* **then**
17 sending ψ^- to processor $p + sign(\mu_k)$;

4.2. Parallel algorithm in shared memory

To use more parallelism than the distributed algorithm presented in the previous paragraph, the shared memory parallelism is expressed through the intel TBB API. For the 2D solver part, the parallelism on the angular directions and on the slices (if a MPI node is responsible of more than two slices) are used. As for the 1D solver, in order to avoid difficulty coming from the support of `MPI_THREAD_MULTIPLE` by the MPI implementation, we parallelized lines 8–15 of algorithm 3. Once again the line 9 is split into two loops. The upper one is moved above the loop over j (line 8) and is defined as a parallel loop.



(a) **with Step Characteristic Scheme**



(b) **with DD Scheme**

Figure 3. Parallel performance of MICADO

4.3. Parallel Efficiency

The parallel efficiency was assessed on the Takeda benchmark with a $200 \times 200 \times 200$ spatial mesh, 10 characteristics per side of cell, and a Gauss-Legendre quadrature formula with 8×4 directions. As we are only interested in computational performance we choose a fixed number of iteration for all iterative algorithms (100 for the power iterations, 1 for the upscattering iterations, 1 for the free scattering iterations and 1 for 2D–1D coupling iterations) without checking convergence. This experiment was performed on EDF's Ivanoe cluster [17] and used 200 nodes with 12 cores on each node. Since a buffered tracking is chosen, the cores are exploited with shared memory algorithm and TBB implementation and only one MPI process per node is launched. Only one tracking storage per node is therefore required. To avoid problem with load balancing, the performance is only evaluated for numbers of nodes which evenly divide the 200 slices. All 12 cores are used except for the one-node case where performances are evaluated from 1 to 12 cores. The $blockSize$ parameter is fixed to 2000 (resp. 4000) for the SC scheme (resp. DD scheme). Figure 3 reports the time and parallel efficiency of the power algorithm (KeffComputation), the 2D solver (solve2D) and the 1D solver (solve1D).

For the SC scheme the results confirm our assumptions. In sequential computations, the 1D part is negligible and justifies our choice to favor the data structure and parallelism of the 2D part over the 1D part. As expected the parallel efficiency of the 2D part is nearly optimal. The performance drop for 100 and 200 nodes is due to load balancing and the small number of parallel tasks in shared memory (the efficiency is theoretically bounded to 89%). As expected, the performance of the 1D part is not so impressive but as the

time spent in 1D part remains lower than the 2D part the global efficiency remains good (better than 80% below 300 cores and better than 60% up to 1200 cores). With 2400 cores which is the maximum of usable cores for this test case with only one slice per node, the limit of the strong scalability study is reached with an efficiency of 46% with only $0.26s$ for one power iteration.

As expected due to the lower arithmetic intensity (no exponential to compute) *, the results of the DD scheme are better in terms of computation time, but worse in terms of parallel of efficiency. For example for the 2D part, a performance drop of 15% due to memory accesses is observed for the use of 12 cores in share memory. As the computation time is reduced and the communication time remains similar, it was expected that for a large number of nodes the 1D part would catch up the 2D part. Until 300 nodes for this test case, the efficiency remains really acceptable.

5. CONCLUSIONS

We presented in this work a neutron transport solver aiming at dealing with 3D whole-core heterogeneous calculations. This solver is based on the 2D–1D coupling method first introduced in [1]: the geometry is split into 2D slices; 2D MOC resolutions on these slices are coupled by axial leakage source terms coming from a set of 1D axial transport problems. Under an assumption of convex mesh cells and non-null cross-sections, a proof of convergence of this iterative process was established.

We then presented our high performance, parallel implementation of this algorithm in the MICADO solver. Benchmarking this solver against the Takeda case shows that the 2D-1D coupling algorithm does not affect the spatial convergence order of the MOC solver. As for performance issues, our study shows that even though the data distribution was designed to improve the 2D solver efficiency, the 1D solver is effective enough to ensure a good parallel efficiency of the global algorithm.

A few drawbacks of the method were also uncovered in this process. In particular, the need for storing axial leakage sources Q_z between iterations could become a very limiting factor in terms of memory consumption for 3D whole-core computations with many energy groups. This problem could be solved be three different ways. First, distributing angular directions and/or energy groups over different parallel processes would ensure that no single computing node needs to store the full multigroup angular field. Second, axial source terms Q_z could be projected on a smaller space to only store an approximated value. Such a method would however require a good acceleration method to counterbalance the loss of information in the initialization of the iterative system. The third way could consist in using higher order schemes to reduce the number of required mesh cells in the 2D slices.

In terms of parallel efficiency some improvements can be done. A major improvement could come from the use of SIMD instructions, which are well suited to the parallelism implied by polar angles. Indeed, preliminary results with the use of SIMD instructions are very promising but need to be confirmed. Such parallelization however reduces the number of available directions for the shared memory parallelism, which would then need to focus on another part of the algorithm, for example the characteristics sweeping in the 2D solver.

* *arithmetic intensity* [18] is the ratio of floating point operations to the number of memory accesses from the main memory. A high arithmetic intensity reduces the effect of memory contention for an application in shared memory.

REFERENCES

[1] G. Lee and N. Cho. "2D/1D fusion method solutions of the three-dimensional transport OECD benchmark problem C5G7 MOX." *Progress in Nuclear Energy*, **vol. 48(5)**, pp. 410 – 423 (2006)

[2] M. J. Halsall. "CACTUS, a characteristics solution to the neutron transport equations in complicated geometries." Tech. Rep. AEEW-R 1291, Atomic Energy Establishment, Winfrith, Dorchester, Dorset, United Kingdom (1980)

[3] R. Sanchez. "Prospects in deterministic three-dimensional whole-core transport calculations." *Nuclear Engineering and Technology*, **vol. 44(2)**, pp. 113–150 (2012)

[4] X. Chai, K. Wang, and D. Yao. "The linear source approximation in three dimension characteristics method." *Proceedings of Mathematics, Computational Methods & Reactor Physics* (2009)

[5] Z. Liu, H. Wu, L. Cao, Q. Chen, and Y. Li. "A new three-dimensional method of characteristics for the neutron transport calculation." *Annals of Nuclear Energy*, **vol. 38(2–3)**, pp. 447–454 (2011)

[6] M.-A. Lajoie. "A generalization of 3D prismatic characteristics along a nonuniform projection mesh." Personal Communication (2012)

[7] H. G. Joo, J. Y. Cho, K. S. Kim, C. C. Lee, and S. Zee. "Methods and performance of a three-dimensional whole-core transport code decart." *Proceedings of The Physics of Fuel Cycles and Advanded Nuclear Systems (CDROM)* (2004)

[8] J. Y. Cho and H. G. Joo. "Solution of the C5G7MOX benchmark three-dimensional extension problems by the DeCART direct whole core calculation code." *Progress in Nuclear Energy*, **vol. 48**, pp. 456–466 (2006)

[9] M. Hursin. *Full Core, Heterogeneous, Time Dependent Neutron Transport Calculations with the 3D Code DeCART*. Ph.D. thesis, University of california at Berkeley (2010)

[10] J. Y. Cho, K. S. Kim, C. C. Lee, S.-Q. Zee, and H.-G. Joo. "Axial SPN and radial MOC coupled whole core transport calculation." *Journal of Nuclear Science and Technology*, **vol. 44(9)**, pp. 1156–1171 (2007)

[11] S. Kosaka and T. Takeda. "Verification of 3d heterogeneous core transport calculation utilizing nonlinear iteration technique." *Journal of Nuclear Science and Technology*, **vol. 41(6)**, pp. 645–654 (2004)

[12] N. Cho, G. Lee, and C. Park. "Fusion of method of characteristics and nodal method for 3-D whole-core transport calculation." **vol. 86**, pp. 322–324 (2002)

[13] N. Cho, G. Lee, and C. Park. "2-D and 3-D whole-core transport calculations of the OECD benchmark problem C5G7 MOX by CRX." *PHYSOR* (2002)

[14] M. Hursin, S. Xiao, and T. Jevremovic. "Synergism of the method of characteristic, R-functions and diffusion solution for accurate representation of 3D neutron interactions in research reactors using the agent code system." *Annals of Nuclear Energy*, **vol. 33(13)**, pp. 1116–1133 (2006)

[15] T. Takeda and H. Ikeda. "Final report on the 3-D neutron transport benchmarks." Tech. rep., OECD/NEA Comittee on Reactor Physics (1991)

[16] "Benchmark on deterministic transport calculations without spatial homogenisation: a 2-D/3-D MOX fuel assembly benchmark." Tech. Rep. NEA/NSC/DOC(2003)16, NEA/OECD (2003)

[17] "Ivanoe." http://i.top500.org/system/177030 (2010)

[18] D. A. Patterson and J. L. Hennessy. *Computer Organization and Design, Fourth Edition: The Hardware/Software Interface* (2008)