International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009)
Saratoga Springs, New York, May 3-7, 2009, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2009)

A Non Overlapping Parallel Domain Decomposition Method Applied to The Simplified Transport Equations

LATHUILIERE Bruno and BARRAULT Maxime
EDF R&D
bruno.lathuiliere@edf.fr; maxime.barrault@edf.fr

RAMET Pierre and ROMAN Jean
INRIA Bordeaux - Sud-Ouest and LaBRI UMR5800
ramet@labri.fr; roman@labri.fr

ABSTRACT

A reactivity computation requires to compute the highest eigenvalue of a generalized eigenvalue problem. An inverse power algorithm is used commonly. Very fine modelizations are difficult to tackle for our sequential solver, based on the simplified transport equations, in terms of memory consumption and computational time. So, we propose a non-overlapping domain decomposition method for the approximate resolution of the linear system to solve at each inverse power iteration. Our method brings to a low development effort as the inner multigroup solver can be re-use without modification, and allows us to adapt locally the numerical resolution (mesh, finite element order). Numerical results are obtained by a parallel implementation of the method on two different cases with a pin by pin discretization. This results are analyzed in terms of memory consumption and parallel efficiency.

Key Words: Simplified Transport Equation, Domain Decomposition, Uzawa-MR

1. Introduction

The operation of a PWR based nuclear plant requires to change the fuel each year in the reactor. It is done while ensuring the safety and the productivity of the plant in service. More precisely, EDF uses the numerical simulation to achieve this goal. In our context, the neutron transport inside a nuclear reactor have to be simulated. So, EDF has developed a fast sequential solver [1] based on the simplified transport equations [2] *. We report on **Table I** its performances, obtained while considering the pin by pin IAEA-3D benchmark [3] with the SP1/SP3 model and the RT0/RT1 finite element approximation [4]:

This sequential code suffers of two limitations. On the one hand, we are not able to run efficiently large scale computation due to the memory consumption and/or the computational time. On the other hand, it is necessary to refine a large part of the mesh as soon as a better numerical approximation is needed in a local part of the core. So, we propose a non overlapping domain decomposition to tackle these problems. In the following, the domain decomposition method is presented. Then, numerical results on the IAEA-3D benchmark and on data coming from real core computation are provided.

*This equations are called SPN

Type of computation	SP1-RT0	SP1-RT1	SP3-RT0	SP3-RT1
DOF Number	31 948 950	254 747 720	63 897 900	509 495 440
Memory consumption	1.5 Go	13.4 Go	2.5 Go	23 Go
Time	143.55 s	$\times$	347.59 s	$\times$

Table I. Performance of the sequential solver for different computations with the same convergence parameters (see section 4 for a description of the target cluster). $\times$ indicates that the computation can't be run because of memory consumption.

2. Description of the SPN problem

The simplified transport equations for a reactivity calculation lead to the following algebraic problem:

Problem 1 *Find the highest k_{eff} value such that: $AX = \frac{1}{k_{eff}} FX$, where A and F are the transport and fission matrices.*

A Generalized Power Inverse Iteration Algorithm is used [5] to solve **Problem 1**. It brings to the resolution of linear systems of type $AX = S$ with several Right Hand Side vectors S. In order to benefit from the pattern of A (See **Figure 1(a)**), we consider the following iterative strategy to solve these linear systems:

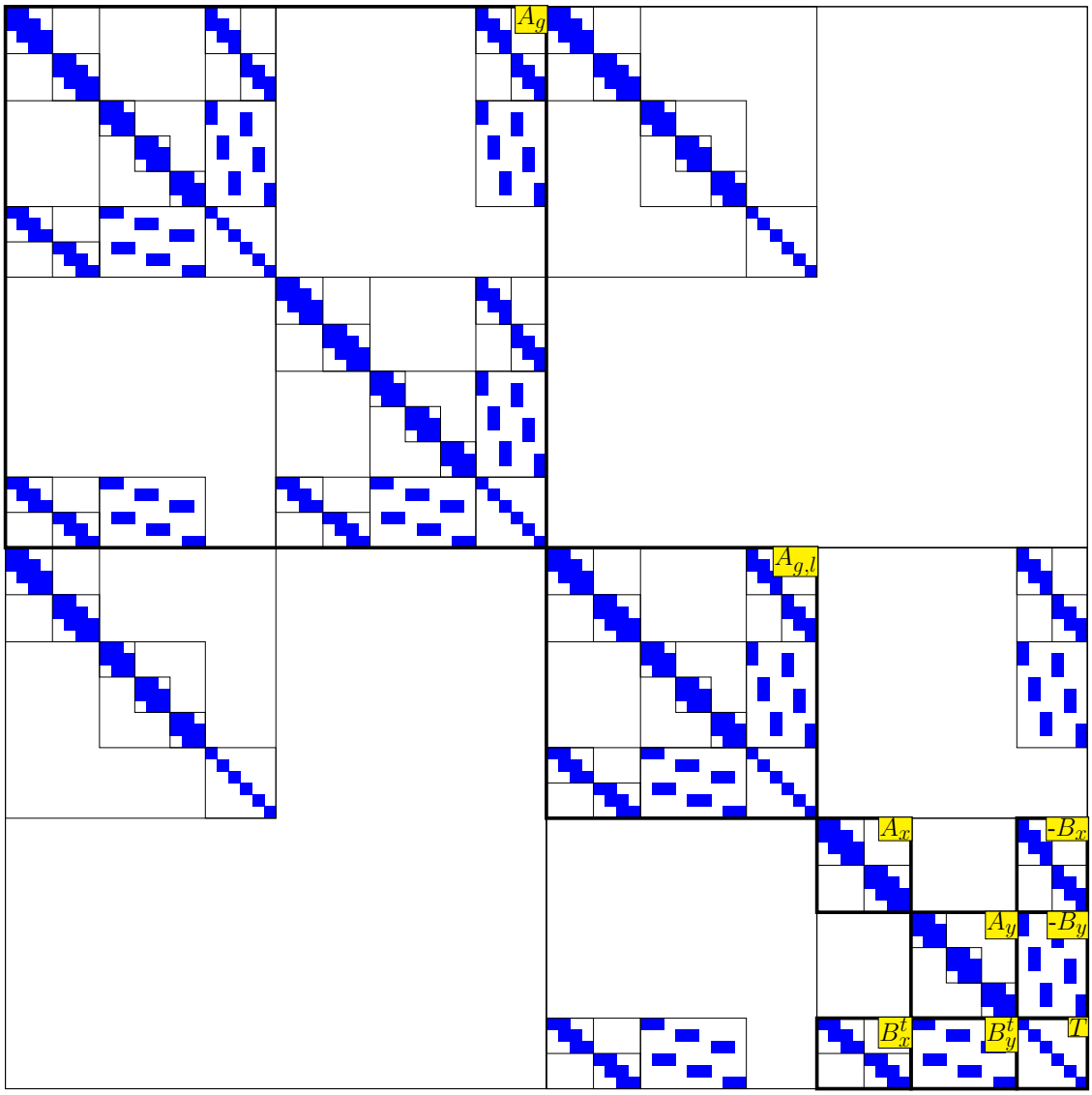
1. to solve $AX = S$, we use an iterative block Gauss-Seidel algorithm. A block corresponds to one energy level defined by the subscript g;
2. for each linear system involving the diagonal blocks A_g of A, we use an iterative block Gauss-Seidel algorithm [†]. A block corresponds to one harmonic defined by the subscript l;
3. for each linear system involving the diagonal blocks $A_{g,l}$ of A_g, an alternating direction method based on a block Gauss-Seidel algorithm is used.

The matrix $A_{g,l}$ comes from the discretization of the mixed dual formulation of a diffusion problem [6]:

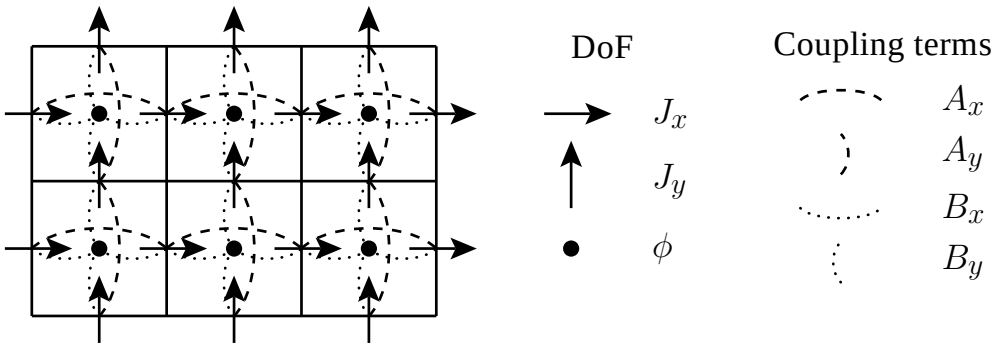
Problem 2 *Find ϕ and $\vec{J}$ respectively the flux and the current such that:*

$$\left\{ \begin{array}{rcccl} \mathrm{div}\,(\vec{J}) & + & \Sigma\phi & = & f \quad \textit{in } \Omega \\ \dfrac{\vec{J}}{D} & + & \vec{\nabla}\phi & = & \vec{0} \quad \textit{in } \Omega \\ & & \phi & = & 0 \quad \textit{on } \partial\Omega \end{array} \right. . \tag{1}$$

[†] In SP1 case, this step is skipped.



(a) Pattern of the matrix A for two energy groups and 2 harmonics (SP3).



(b) Mesh, degrees of freedom(DoF) and coupling terms.

Figure 1. Discretization with the finite element RT0.

where $\Sigma,\ D \in C^0(\Omega)$ *are borned and positive functions.*

A cartesian mesh is used to discretize this problem. For each cell K_n, we compute D_n (resp. Σ_n) the value of the function D (resp. Σ). The Raviart-Thomas finite element (RTk) defines the spaces V_h and Q_h [7, 8]. The discrete mixed dual variational formulation of **Problem 2** is:

Problem 3 *Find* $(\phi, \vec{J}) \in V_h \times Q_h$ *such that:*

$$\left\{ \begin{array}{ll} a_h(\overrightarrow{J}, \overrightarrow{q}) - b_h(\phi, \overrightarrow{q}) = 0 & \forall \overrightarrow{q} \in Q_h \\ b_h(v, \overrightarrow{J}) + t_h(\phi, v) = s_h(v) & \forall v \in V_h \end{array} \right. \quad \textit{with} \quad \left| \begin{array}{l} a_h(\overrightarrow{J}, \overrightarrow{q}) = \displaystyle\sum_n \frac{1}{D_n} \int_{K_n} \overrightarrow{J} . \overrightarrow{q}\, d\Omega \\ b_h(\phi, \overrightarrow{q}) = \displaystyle\int_\Omega \phi \ \mathrm{div}\ \left(\overrightarrow{q}\right) d\Omega \\ t_h(\phi, v) = \displaystyle\sum_n \Sigma_n \int_{K_n} \phi\ v\ d\Omega \\ s_h(v) = \displaystyle\int_\Omega f\ v\ d\Omega \end{array} \right.$$

Thanks to the Raviart-Thomas finite element properties, a very sparse algebraic system is obtained. The following unknowns and coupling terms (See **Figure 1(b)**) are defined:

- J_d contains the degrees of freedom of current in direction d;
- ϕ contains the degrees of freedom of flux;
- S contains the source terms;
- $A_{dd'}$ is the matrix which contains the coupling terms between J_d and J'_d. As $A_{dd'} = 0$ when $d \neq d'$, A_{dd} is written A_d. A_d is a symmetric positive definite banded matrix (tridiagonal in RT0 and pentadiagonal in RT1);
- T is a positive definite diagonal matrix;
- B_d is the matrix which contains the coupling terms between J_d and ϕ.

So in a 2D case, a linear system of the following form has to be solved:

$$\begin{pmatrix} A_x & & -B_x \\ & A_y & -B_y \\ B_x^t & B_y^t & T \end{pmatrix} \begin{pmatrix} J_x \\ J_y \\ \phi \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ S \end{pmatrix}. \tag{2}$$

The flux $\phi = T^{-1}(S - B_x^t J_x - B_y^t J_y)$ is eliminated to solve (2). Hence the current unknowns are obtained by a Block Gauss Seidel algorithm on the following symmetric positive definite system:

$$\begin{pmatrix} W_x & B_x T^{-1} B_y^t \\ B_y T^{-1} B_x^t & W_y \end{pmatrix} \begin{pmatrix} J_x \\ J_y \end{pmatrix} = \begin{pmatrix} B_x T^{-1} S \\ B_y T^{-1} S \end{pmatrix} \quad \text{with} \left| \begin{array}{l} \forall d \in \{x, y\} \\ W_d = A_d + B_d T^{-1} B_d^t \end{array} \right. .$$

The matrices W_d and A_d share the same pattern. The resolution of linear systems involving W_d is based on a Cholesky factorization. The dense matrices $B_d T^{-1} B_{d'}^t$ are not build in memory as we consider three successive sparse products.

Finally, the overall algorithm is made of four iterative imbricated loops (The power algorithm and the three inner loops presented page 2). For our applications, we obtain, for a given accuracy, the best performance in terms of CPU time by fixing the number of iterations of all Gauss-Seidel loops to one.

3. Description of the domain decomposition method using Lagrange multipliers

The first parallelization of this algorithm [9] used the independence of the current lines (see **Figure 1(b)**). From an algebraic point of view, the banded matrices W_d are block diagonal. Each block can be treated in parallel. Due to the structure of the matrix B_y (see **Figure 1(a)**) a large amount of communications was required. So, the speed-up was limited. To avoid this limitation, domain decomposition methods are well fitted. In a first time, the nodale synthesis approaches [9, 10] did not give the excepted results in term of computational time, so the next attempts was to solve only the linear systems $AX = S$ by a domain decomposition strategy. A success was obtained with a domain decomposition method based on a Schwarz multiplicative algorithm with Robin transfer conditions [10]. A first drawback of this method is the introduction of a small overlapping (one cell) between subdomains due to the non-continuity of the flux on the interface [‡]. A second and main drawback of this algorithm comes from the good estimation of the parameter α for the Robin transfer conditions. *Trial and error* is the only known method to choose this parameter in our context.

In order to face up these difficulties, we study a non-overlapping domain decomposition method based on Lagrange multipliers, where:

- the local solvers are identical to the global sequential solver. In terms of code *reuseability* this is an important point: we do not need to develop and maintain a new solver;
- this method will allow us to deal with non-matching grids.

First, we cut the space Ω into two non-overlapping subdomains [§] Ω_1 and Ω_2 such as $\Omega = \Omega_1 \cup \Omega_2$ and $\Gamma = \partial\Omega_1 \cap \partial\Omega_2 = \Omega_1 \cap \Omega_2$. The interface Γ is directed by the normal vector $\overrightarrow{n}$. To obtain the continuous multidomain formulation (**Problem 4**), the equations (1) are rewritten in each subdomain with a new boundary condition on the interface Γ. To get the same solution, we add an equation (4) to ensure the continuity of the normal component of the current $\overrightarrow{J}$:

[‡] Coming from the use of the Finite Element RTk.

[§] For the sake of simplicity, the presentation of the method is limited to two subdomains. The generalization to more subdomains is pretty obvious.

Problem 4 *Find* $(\phi_1, \vec{J_1})$ *and* $(\phi_2, \vec{J_2})$ *such that:*

$$\left\{\begin{array}{rcll} \mathrm{div}\,(\vec{J_1}) & + & \Sigma\phi_1 = f & \textit{in } \Omega_1 \\ \dfrac{\vec{J_1}}{D} & + & \vec{\nabla}\phi_1 = \vec{0} & \textit{in } \Omega_1 \\ \phi_1 & = & \phi_\Gamma & \textit{on } \Gamma \\ \phi_1 & = & 0 & \textit{on } \partial\Omega_1 \cap \partial\Omega \end{array}\right. \quad \left\{\begin{array}{rcll} \mathrm{div}\,(\vec{J_2}) & + & \Sigma\phi_2 = f & \textit{in } \Omega_2 \\ \dfrac{\vec{J_2}}{D} & + & \vec{\nabla}\phi_2 = \vec{0} & \textit{in } \Omega_2 \\ \phi_2 & = & \phi_\Gamma & \textit{on } \Gamma \\ \phi_2 & = & 0 & \textit{on } \partial\Omega_2 \cap \partial\Omega \end{array}\right. \tag{3}$$

$$\vec{J_1} \cdot \vec{n} = -\vec{J_2} \cdot \vec{n} \quad \textit{on } \Gamma \tag{4}$$

The new variable ϕ_Γ in (3) leads to the introduction of Lagrange Multipliers Λ. So the variational formulation of **Problem 4** is :

Problem 5 *Find* $(\phi_1, \vec{J_1}) \in L^2(\Omega_1) \times H(\mathrm{div}\,, \Omega_1)$, $(\phi_2, \vec{J_2}) \in L^2(\Omega_2) \times H(\mathrm{div}\,, \Omega_2)$ *and* $\Lambda \in H^{1/2}(\Gamma)$ *such that:*

$$\left\{\begin{array}{rcll} \displaystyle\int_{\Omega_1} \mathrm{div}\,(\vec{J_1})\, u_1 \, d\Omega_1 + \int_{\Omega_1} \Sigma_1\phi_1 u_1 \, d\Omega_1 = \int_{\Omega_1} f_1 u_1 \, d\Omega_1 & & \forall\, u_1 \in L^2(\Omega_1) \\ \displaystyle\int_{\Omega_1} \frac{\vec{J_1}}{D_1} \cdot \vec{v}_1 \, d\Omega_1 - \int_{\Omega_1} \phi_1 \,\mathrm{div}\,(\vec{v}_1)\, d\Omega_1 + \int_\Gamma \Lambda \vec{v}_1 \cdot \vec{n} \, d\Gamma = 0 & & \forall\, \vec{v}_1 \in H(\mathrm{div}\,, \Omega_1) \\ \displaystyle\int_{\Omega_2} \mathrm{div}\,(\vec{J_2})\, u_2 \, d\Omega_2 + \int_{\Omega_2} \Sigma_2\phi_2 u_2 \, d\Omega_2 = \int_{\Omega_2} f_2 u_2 \, d\Omega_2 & & \forall\, u_2 \in L^2(\Omega_2) \\ \displaystyle\int_{\Omega_2} \frac{\vec{J_2}}{D_2} \cdot \vec{v}_2 \, d\Omega_2 - \int_{\Omega_2} \phi_2 \,\mathrm{div}\,(\vec{v}_2)\, d\Omega_2 - \int_\Gamma \Lambda \vec{v}_2 \cdot \vec{n} \, d\Gamma = 0 & & \forall\, \vec{v}_2 \in H(\mathrm{div}\,, \Omega_2) \\ \displaystyle\int_\Gamma \left(\vec{J_1} - \vec{J_2}\right) \cdot \vec{n}\, \mu \, d\Gamma = 0 & & \forall\, \mu \in H^{1/2}(\Gamma) \end{array}\right. .$$

In each subdomain a cartesian mesh is considered. On each cell K_n^i, D_i (resp. Σ_i) has a constant value D_i^n (resp. Σ_i^n). **Problem 5** becomes after discretization:

Problem 6 *Find* $(\phi_1, \vec{J}_1) \in V_h^1 \times Q_h^1$, $(\phi_2, \vec{J}_2) \in V_h^2 \times Q_h^2$ *and* $\Lambda_h \in V_h^\Gamma$ *such that:*

$$\begin{cases} a_h^1(\overrightarrow{J_1}, \overrightarrow{q_1}) - b_h^1(\phi_1, \overrightarrow{q_1}) = -c_h^1(\Lambda_h, \overrightarrow{q_1}) & \forall \overrightarrow{q_1} \in Q_h^1 \\ b_h^1(v_1, \overrightarrow{J_1}) + t_h^1(\phi_1, v_1) = s_h^1(v_1) & \forall v_1 \in V_h^1 \\ \\ a_h^2(\overrightarrow{J_2}, \overrightarrow{q_2}) - b_h^2(\phi_2, \overrightarrow{q_2}) = -c_h^2(\Lambda_h, \overrightarrow{q_2}) & \forall \overrightarrow{q_2} \in Q_h^2 \\ b_h^2(v_2, \overrightarrow{J_2}) + t_h^2(\phi_2, v_2) = s_h^2(v_2) & \forall v_2 \in V_h^2 \\ \\ c_h^1(\mu_h, \overrightarrow{J_1}) + c_h^2(\mu_h, \overrightarrow{J_2}) = 0 & \forall \mu_h \in V_h^\Gamma \end{cases}$$

with

$$\begin{cases} a_h^i(\overrightarrow{J_i}, \overrightarrow{q_i}) = \displaystyle\sum_n \frac{1}{D_n^i} \int_{K_n^i} \overrightarrow{J_i}.\overrightarrow{q_i}\, d\Omega \\ \\ b_h^i(\phi_i, \overrightarrow{q_i}) = \displaystyle\int_{\Omega_i} \phi_i \,\mathrm{div}\,(\overrightarrow{q_i})\, d\Omega \end{cases} \quad \textit{and} \quad \begin{cases} t_h^i(\phi_i, v_i) = \displaystyle\sum_n \Sigma_n^i \int_{K_n^i} \phi_i\, v_i\, d\Omega \\ \\ s_h^i(v_i) = \displaystyle\int_{\Omega_i} f_i\, v_i\, d\Omega \\ \\ c_h^i(\Lambda_h, \overrightarrow{q_i}) = (-1)^{i+1} \displaystyle\int_\Gamma \Lambda_h \vec{q_i} \cdot \vec{n}\, d\Gamma \end{cases} .$$

The approximation spaces V_h^i and Q_h^i come from the finite element discretization scheme. In case of matching grids (and of the same finite element order used in each subdomain), V_h^Γ is chosen as the trace of Q_h^i. This choice gives the equivalence between **Problem 3** and **Problem 6**. In case of non-matching grids, a mortar technique [11, 12] is used: V_h^Γ is the trace of Q_h^1 or Q_h^2. The chosen subdomain is called the master subdomain. In our case the master subdomain is the subdomain with the finest mesh and the highest finite element order [¶]. The numerical applications in this paper concern only matching grids case.

So, **Problem 6**, in a 2D case, leads to the following algebraic system [‖]:

$$\left(\begin{array}{ccc|ccc|c} A_x^1 & & -B_x^1 & & & & C_{\Lambda\to1}^x \\ & A_y^1 & -B_y^1 & & & & 0 \\ B_x^{1t} & B_y^{1t} & T^1 & & & & \\ \hline & & & A_x^2 & & -B_x^2 & C_{\Lambda\to2}^x \\ & & & & A_y^2 & -B_y^2 & 0 \\ & & & B_x^{2t} & B_y^{2t} & T^2 & \\ \hline {C_{\Lambda\to1}^x}^t & 0 & & {C_{\Lambda\to2}^x}^t & 0 & & \end{array}\right) \left(\begin{array}{c} J_x^1 \\ J_y^1 \\ \phi^1 \\ \hline J_x^2 \\ J_y^2 \\ \phi^2 \\ \hline \Lambda \end{array}\right) = \left(\begin{array}{c} 0 \\ 0 \\ S^1 \\ \hline 0 \\ 0 \\ S^2 \\ \hline 0 \end{array}\right). \tag{5}$$

The system (5) can be written as a saddle point problem:

$$\begin{pmatrix} A_{d=1} & 0 & C_{\Lambda\to1} \\ 0 & A_{d=2} & C_{\Lambda\to2} \\ C_{\Lambda\to1}^t & C_{\Lambda\to2}^t & 0 \end{pmatrix} \begin{pmatrix} X_{d=1} \\ X_{d=2} \\ \Lambda \end{pmatrix} = \begin{pmatrix} S_{d=1} \\ S_{d=2} \\ 0 \end{pmatrix} \tag{6}$$

[¶] When a subdomain has the finest mesh and the other one has the highest finite element order, we do not introduce a master subdomain: V_h^Γ is taken as the trace of the space defined by the finest mesh and the highest finite element order.

[‖] The null term explicitly written in the matrix are null as there is no interface over direction y.

where for each subdomain i, $A_{d=i}$ is the local matrix, $X_{d=i}$ the local unknowns, and $S_{d=i}$ the local source term.

Through matrix reordering, the domain decomposition can be placed at each level of the inner/outer iterative process. In order to be less intrusive as possible and to get a coarse grain parallelism, we have chosen to emplace the domain decomposition over the multigroup solver. It means that $A_{d=i}$ is the multigroup matrix of the subdomain Ω_i.

To solve (6) a Krylov based Uzawa algorithm was proposed [13]. In practice, for our context of inner-outer iterations, a Krylov algorithm is less efficient than **Algorithm 1** based on the Minimal Residual algorithm (MR) [14]. To present this algorithm called Uzawa-MR, the following notations are needed:

$$C = \begin{pmatrix} C_{\Lambda\to 1} \\ C_{\Lambda\to 2} \end{pmatrix}; \quad A_{DD} = \begin{pmatrix} A_{d=1} & 0 \\ 0 & A_{d=2} \end{pmatrix}; \quad S = \begin{pmatrix} S_{d=1} \\ S_{d=2} \end{pmatrix}.$$

We have to solve the generalized [**] saddle point system:

$$\begin{pmatrix} A_{DD} & C \\ C^t & 0 \end{pmatrix} \begin{pmatrix} X \\ \Lambda \end{pmatrix} = \begin{pmatrix} S \\ 0 \end{pmatrix};$$

Λ is the solution of the interface problem:

$$C^t A_{DD}^{-1} C\Lambda = C^t A_{DD}^{-1} S. \tag{7}$$

The Uzawa-MR algorithm corresponds to the application of MR to (7) with a small adjustment: instead of computing X once Λ is computed by MR, X is updated by an AXPY operation [††] at each MR iteration.

Algorithm 1: Interface algorithm

$\Lambda = \Lambda_0$;
$X = A_{DD}^{-1}(S - C\Lambda)$;
$r = C^t X$;
while *! Convergence* **do**
  $\hat{X} = A_{DD}^{-1}(Cr)$;
  $M_r = C^t \hat{X}$;
  $\rho = -\dfrac{< r, M_r >}{< M_r, M_r >}$;
  $\Lambda = \Lambda + \rho r$;
  $r = r + \rho M_r$;
  $X = X - \rho \hat{X}$;
end

Remarks:

- Λ is initialized with the value Λ_0 obtained at the previous inverse power iteration.
- To compute the product by A_{DD}^{-1}, we use in parallel the sequential solver in each subdomain. The parallel implementation and data distribution is described in [13].
- The local solvers called at the line ($X = A_{DD}^{-1}(S - C\Lambda)$) are initialized with the value obtained at the previous inverse power iteration.
- The local solvers called at the line ($\hat{X} = A_{DD}^{-1}(Cr)$) are initialized with the null vector.

Compared to the sequential monodomain algorithm, we finally add one loop to get convergence of **Algorithm 1**. In the following, we consider a fixed number i of iterations: the algorithm is

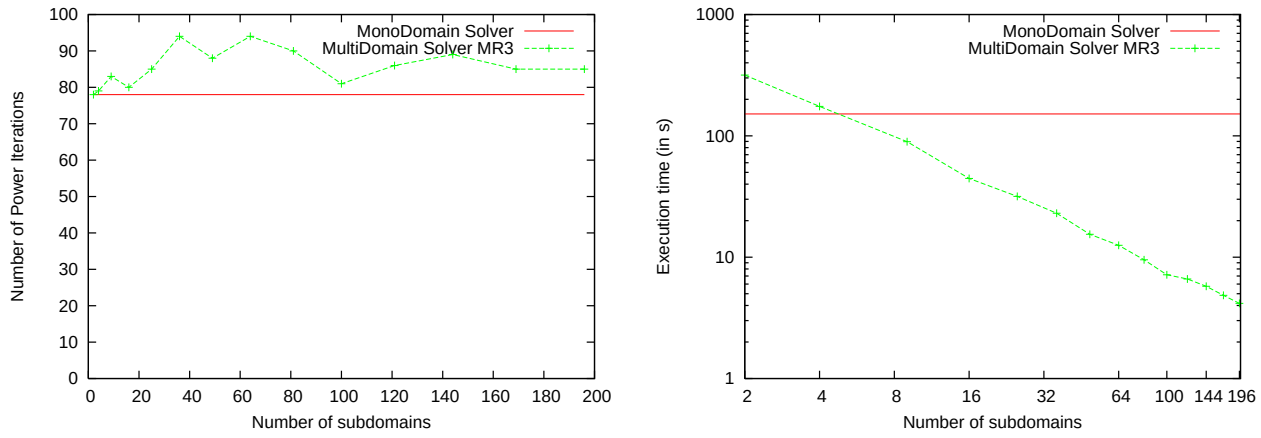
[**] Generalized terminology comes from the non-symmetry of A_{DD}.
[††] AXPY : $Y \leftarrow Y + a * X$, where X and Y are two vectors and a is a scalar.

called `MRi`. In the same way as the sequential monodomain solver, the outer iteration is used to get convergence. The overall algorithm is now:

1. Inverse Power Iteration algorithm using a Chebyshev acceleration;
2. Uzawa-MR algorithm over subdomains;
3. Block Gauss-Seidel algorithm over energy groups;
4. Block Gauss-Seidel algorithm over harmonics;
5. Alternate direction algorithm based on block-Gauss Seidel Algorithm;
6. Exact resolution in each current direction using Cholesky factorization.

4. Experimental results

The tests have been performed on a cluster with 208 nodes (each consists in two Intel Xeon processors, 3.40GHz, 2MB Cache and with 4 GB PC3200 DDR2). The used MPI implementation is MPICH 1.2.7 with an Infiniband (openib-2.0.5) network. During batch submissions, one node is assigned to each subdomain to avoid concurrent memory access.



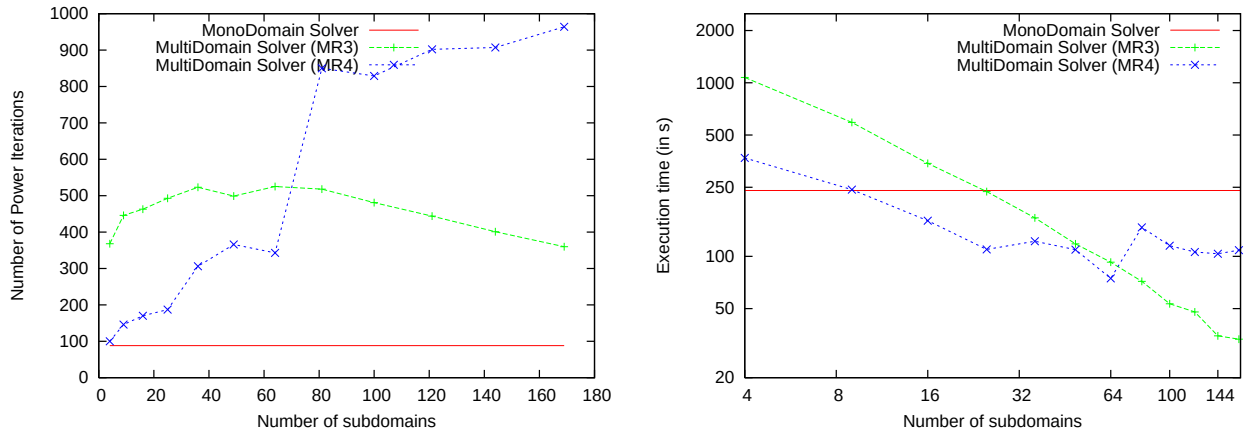
(a) (b)

Figure 2. BenchMark IAEA: behavior of `MR3` algorithm

To validate our method, we consider first the two group homogeneous IAEA-3D benchmark [3]. For a SP1-RT0 pin by pin computation, the computational mesh used is made of $289 \times 289 \times 38$ cells. The inverse power algorithm has the following convergence criteria:

$$\frac{\|S_n - S_{n-1}\|}{\|S_n\|} < 10^{-6} \quad \text{and} \quad \frac{|\lambda_n - \lambda_{n-1}|}{|\lambda_n|} < 10^{-6}$$

where S_n and λ_n are the source of fission and the estimated eigenvalue at iteration n.



(a) (b)

Figure 3. Real Core: behavior of MR3 and MR4 algorithms

The MR3 algorithm is used. So, our method brings to four more amount of computations than the sequential monodomain solver at each power iteration. Consequently, the best efficiency we can expect is 25%. We have performed the parallel domain decomposition method from 2 to 192 subdomains. Except for the two subdomain case, the domain is cut into the square root of the number of nodes, in each radial direction (x and y). We have checked that our method brings to the same solution as the sequential solver for the requested accuracy.

We report on **Figure 2**, for each domain partition, the number of power iterations and the execution time. The execution time of the sequential monodomain solver is reported also. On the one hand, the introduction of the domain decomposition leads to a small increase ($< 20\%$) in the number of power iterations (see **Figure 2(a)**). On the other hand, the efficiency of MR3 algorithm is constant and nearly equal to 21%, which is rather good compared to the best we can expect ($\approx 25\%$). It ensures the good quality of the parallel implementation of the method. In other words, five processors are required to get a parallel domain decomposition solver faster than the sequential monodomain one. Once this price is accepted, we get a good parallel algorithm: the execution time decreases proportionally with the number of processors (see **Figure 2(b)**). We obtain similar results while using the RT1 finite element approximation.

Secondly, we consider a SP1 homogeneous pin by pin computation [‡‡] on data coming from an industrial application. The results obtained by MR3 and MR4 algorithms are reported on **Figure 3**. We observe that the domain decomposition loop leads to an important increase in the iterations number of the outer loop. With MR4, nine processors are required to get a parallel solver faster than the sequential monodomain one, and a speed-up between 6 and 7 is reached with a large number of nodes. In this case, the main interest of the method is to tackle memory problem.

[‡‡] Similar results are obtained on heterogeneous pin by pin computation.

5. Conclusions

The proposed domain decomposition method in the difficult context of an approximate resolution of the linear system at each inverse power iteration reveals very reliable and efficient on SP1 homogeneous pin by pin computation coming from the IAEA benchmark. On data coming from an industrial case, the method answers very well to the memory problem but it is limited in terms of parallel efficiency.

Hardware accelerators such as Graphic Processor Units suffer of insufficient memory size but can be really relevant in terms of computational time. In [15] the authors obtained a speed-up of 30 for the monodomain solver. So, it seems natural to use the domain decomposition method on a GPU cluster in order to take advantage of GPU acceleration without memory limitation.

We are also working on a more code intrusive domain decomposition where the loop over the subdomains is emplaced under the loop over current directions. The new method is much less generic than the method developed in this paper and parallelism has a finer grain. But at this level, the method has interesting properties:

- The interface system is symmetric positive definite;
- The local solvers are exact. On the one hand the use of Krylov algorithm (like Conjugate Gradient) becomes possible without the difficulties coming from inner iterations. On the other hand, it allows us to build explicitly the interface system and to solve it exactly. The domain decomposition method does not introduce difficulties in the inner/outer process.

Preliminary results obtained on this method are very promising. We focus now our work on optimizing and testing it.

References

[1] L. Plagne and A. Ponçot. "Generic programming for deterministic neutron transport codes." *In Proceedings of Mathematical and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications, Palais des Papes, Avignon, France*, p. 10 (2005)

[2] C. Pomraning. "Asymptotic and variational derivation of the simplified Pn equations." *American Nuclear Energy*, **vol. 20**, pp. 623–637 (1993)

[3] B. Micheelsen. "3D IAEA Benchmark Problem." Tech. rep., IAEA (1977)

[4] D. Schneider. *Éléments finis mixtes duaux pour la résolution numérique de l'équation de la diffusion neutronique en géométrie hexagonale*. Ph.D. thesis, Université Paris VI (2001)

[5] H. A. van der Vorst. "Computational methods for large eigenvalue problems." J. L. RG. Ciarlet, editor, *Handbook of Numerical Analysis*, vol. 8, pp. 3–179. North-Holland (Elsevier) (2002)

[6] A. Baudron, J. Lautard, and D. Schneider. "Mixed dual methods for neutronic reactor core calculations in the CRONOS system." *Reactor Physics and Environmental Analysis of Nuclear Systems* (1999)

[7] P. Raviart and J. Thomas. "A mixed finite element method for second order elliptic problems." *Lecture Notes in Mathematics* (1977)

[8] J. C. Nédélec. "A new family of mixed finite elements in $\mathbb{R}^3$." *Numerische Mathematik*, **vol. 50**, pp. 57–81 (1986)

[9] K. Pinchedez. *Calcul parallèle pour les équations de diffusion et de transport homogènes en neutronique*. Ph.D. thesis, Université Paris XI (1999)

[10] P. Guérin. *Méthodes de décomposition de domaine pour la formulation mixte duale du problème critique de la diffusion des neutrons*. Ph.D. thesis, Université Paris VI (2007)

[11] C. Bernardi, Y. Maday, and A. T. Patera. "A new non conforming approach to domain decomposition: The mortar element method." H. Brezis and J.-L. Lions, editors, *Collège de France Seminar*. Pitman. This paper appeared as a technical report about five years earlier. (1994)

[12] F. B. Belgacem. *Discrétisations 3D non conformes pour la méthode de décomposition de domaine des éléments avec joints : analyse mathématique et mise en oeuvre pour le problème de Poisson*. Ph.D. thesis, EDF (1993)

[13] M. Barrault, B. Lathuilière, P. Ramet, and J. Roman. "A domain decomposition method applied to the simplified transport equations." *Proceedings IEEE CSE'08, 11th IEEE International Conference on Computational Science and Engineering, São Paulo, SP, Brazil*, pp. 91–97 (2008)

[14] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2003)

[15] W. Kirschenmann, L. Plagne, S. Ploix, A. Ponçot, and S. Vialle. "Massively parallel solving of 3D simplified P_N equations on graphic processing units." *Proceedings of Mathematics, Computational Methods & Reactor Physics* (2009)