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)

DOMAIN DECOMPOSITION PARALLELIZATION OF NEUTRON TRANSPORT CALCULATIONS

S. Van Criekingen*
Forschungszentrum Karlsruhe, Institute for Neutron Physics and Reactor Technology
Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
criekingen@irs.fzk.de

F. Nataf
J.-L. Lions Laboratory, Paris VI University, CNRS UMR 7598, France

P. Havé
French Petroleum Institute, Rueil-Malmaison, France

ABSTRACT

A Domain Decomposition method is presented in view of the parallelization of large neutron transport calculations. The second-order even-parity form of the time-independent Boltzmann transport equation is considered, with spatio-angular discretization performed using finite elements and spherical harmonic expansions (P_N method). Enabling one processor to handle more than one domain yields proper evaluations of the speed-up, and allows the use of a better-quality decomposition (better load balancing) without requiring a higher number of processors.

Key Words: domain decomposition, parallel computing, neutron transport, spherical harmonics

1. INTRODUCTION

A Domain Decomposition (DD) method is here presented for the parallelization of neutron transport calculations. Application of DD methods to neutron transport was already addressed by de Oliveira et al. [1] and by Guérin et al. [2]. We consider here a non-overlapping DD method, that is, the global domain is split up into several (sub)domains sharing not more than interfaces. Our DD method can be related to an idea of P.L. Lions [3], as described in section 2.2.

We use the even-parity formulation of the Boltzmann transport equation. The spatial discretization is performed using finite elements (FEs), and the angular discretization is performed using spherical harmonic expansions (P_N method).

Section 2 presents the theoretical algebraic formulation of the proposed DD method. Application to neutron transport is considered in section 3, which demonstrates the potential of the method on the 3-D Takeda 1 benchmark.

*Corresponding author

2. ALGEBRAIC FORMULATION

2.1. A Two-domain Example in Two Dimensions

In this section, we consider the general problem

$$\mathcal{A}\Psi = Q \tag{1}$$

where $\mathcal{A}$ is a linear operator (e.g., the Boltzmann transport operator), Ψ is the unknown, and Q is a source term. In view of simplification, we consider here that problem (1) is defined on only two non-overlapping (sub)domains E^1 and E^2, sharing one common edge $\Gamma = E^1 \cap E^2$. Moreover, we also consider that, as depicted in Fig. 1, both domains E^1 and E^2 are made out of one single FE, namely at this point a lowest-order (i.e., linear) conforming Lagrangian FE with 4 nodes located at the vertices. The nodal point values of Ψ in E^i ($i = 1, 2$) are denoted by φ^i_j ($j = 1..4$), as illustrated in Fig. 1.

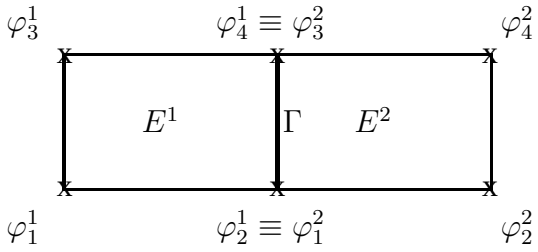

Figure 1. A two-domain example, where each domain is made out of one lowest-order conforming Lagrangian FE. The two domains are coupled by enforcing that φ^1_2 be equal to φ^2_1, as well as φ^1_4 be equal to φ^2_3.

Applying the FE method separately to E^1 and E^2 results in two symmetric (assuming $\mathcal{A}$ is self-adjoint) linear systems of the form

$$\begin{pmatrix} a^i_{11} & a^i_{21} & a^i_{31} & a^i_{41} \\ a^i_{21} & a^i_{22} & a^i_{32} & a^i_{42} \\ a^i_{31} & a^i_{32} & a^i_{33} & a^i_{43} \\ a^i_{41} & a^i_{42} & a^i_{43} & a^i_{44} \end{pmatrix} \begin{pmatrix} \varphi^i_1 \\ \varphi^i_2 \\ \varphi^i_3 \\ \varphi^i_4 \end{pmatrix} = \begin{pmatrix} q^i_1 \\ q^i_2 \\ q^i_3 \\ q^i_4 \end{pmatrix}, \qquad i = 1, 2.$$

To couple these two linear systems together, we use the fact that, since they represent the unknown Ψ at the same physical point, the values φ^1_2 and φ^2_1 must be identical. (More precisely, these two values will asymptotically tend to the same value in an iterative process involving parallel calculations on the two domains.) Similarly, the values φ^1_4 and φ^2_3 must be identical. We therefore

write

$$\begin{pmatrix} a_{11}^1 & a_{21}^1 & a_{31}^1 & a_{41}^1 & 0 & 0 & 0 & 0 \\ a_{21}^1 & a_{22}^1 & a_{32}^1 & a_{42}^1 & a_{11}^2 & a_{21}^2 & a_{31}^2 & a_{41}^2 \\ a_{31}^1 & a_{32}^1 & a_{33}^1 & a_{43}^1 & 0 & 0 & 0 & 0 \\ a_{41}^1 & a_{42}^1 & a_{43}^1 & a_{44}^1 & a_{31}^2 & a_{32}^2 & a_{33}^2 & a_{43}^2 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{21}^2 & a_{22}^2 & a_{32}^2 & a_{42}^2 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & a_{41}^2 & a_{42}^2 & a_{43}^2 & a_{44}^2 \end{pmatrix} \begin{pmatrix} \varphi_1^1 \\ \varphi_2^1 \\ \varphi_3^1 \\ \varphi_4^1 \\ \varphi_1^2 \\ \varphi_2^2 \\ \varphi_3^2 \\ \varphi_4^2 \end{pmatrix} = \begin{pmatrix} q_1^1 \\ q_2^1 + q_1^2 \\ q_3^1 \\ q_4^1 + q_3^2 \\ 0 \\ q_2^2 \\ 0 \\ q_4^2 \end{pmatrix}. \tag{2}$$

In the matrix on the left-hand side of Eq. (2), we moreover replace the ± 1 coefficients by some tunable α coefficient, namely α_a for $\varphi_2^1 \equiv \varphi_1^2$ and α_b for $\varphi_4^1 \equiv \varphi_3^2$. (This yields an algebraically - but not numerically - equivalent system.) Moreover, the diagonal blocks can be made both symmetric by adding and subtracting the lines corresponding to a same duplicated unknown. After re-organization, the system (2) can be written in a general two-domain formulation:

$$\begin{pmatrix} A^{11} & A^{1\,\Gamma} & 0 & 0 \\ A^{\Gamma\,1} & A_1^{\Gamma\,\Gamma} + I^\alpha & A^{\Gamma\,2} & A_2^{\Gamma\,\Gamma} - I^\alpha \\ 0 & 0 & A^{22} & A^{2\,\Gamma} \\ A^{\Gamma\,1} & A_1^{\Gamma\,\Gamma} - I^\alpha & A^{\Gamma\,2} & A_2^{\Gamma\,\Gamma} + I^\alpha \end{pmatrix} \begin{pmatrix} \varphi^1 \\ \varphi^{\Gamma,1} \\ \varphi^2 \\ \varphi^{\Gamma,2} \end{pmatrix} = \begin{pmatrix} Q^1 \\ Q_1^{\Gamma\,\Gamma} + Q_2^{\Gamma\,\Gamma} \\ Q^2 \\ Q_1^{\Gamma\,\Gamma} + Q_2^{\Gamma\,\Gamma} \end{pmatrix} \tag{3}$$

where, if as above each domain contains only one FE, one has

$$\varphi^1 = \begin{pmatrix} \varphi_1^1 \\ \varphi_3^1 \end{pmatrix}, \varphi^{\Gamma,1} = \begin{pmatrix} \varphi_2^1 \\ \varphi_4^1 \end{pmatrix}, \varphi^2 = \begin{pmatrix} \varphi_2^2 \\ \varphi_4^2 \end{pmatrix}, \varphi^{\Gamma,2} = \begin{pmatrix} \varphi_1^2 \\ \varphi_3^2 \end{pmatrix},$$

$$A^{11} = \begin{pmatrix} a_{11}^1 & a_{31}^1 \\ a_{31}^1 & a_{33}^1 \end{pmatrix}, A^{1\Gamma} = A^{\Gamma 1 T} = \begin{pmatrix} a_{21}^1 & a_{41}^1 \\ a_{32}^1 & a_{43}^1 \end{pmatrix},$$

$$A_1^{\Gamma\Gamma} = \begin{pmatrix} a_{22}^1 & a_{42}^1 \\ a_{42}^1 & a_{44}^1 \end{pmatrix}, I^\alpha = \begin{pmatrix} \alpha_a & 0 \\ 0 & \alpha_b \end{pmatrix}, \ldots$$

Note that the global system matrix in (3) is not symmetric, but that the diagonal blocks

$$\mathbf{A_1} = \begin{pmatrix} A^{11} & A^{1\,\Gamma} \\ A^{\Gamma\,1} & A_1^{\Gamma\,\Gamma} + I^\alpha \end{pmatrix} \quad \text{and} \quad \mathbf{A_2} = \begin{pmatrix} A^{22} & A^{2\,\Gamma} \\ A^{\Gamma\,2} & A_2^{\Gamma\,\Gamma} + I^\alpha \end{pmatrix} \tag{4}$$

are both symmetric. Our DD method then consists in solving the global system (3) using an iterative method. Since the system matrix in (3) is not symmetric, we use the Krylov-type method known as the BiCGStab method [4] at this global level. We moreover implement a block-diagonal preconditioning, which implies that the blocks $\mathbf{A_1}$ and $\mathbf{A_2}$ have to be "inverted" (in fact, linear systems have to be solved) at each BiCGStab iteration. These "inversions" are performed in parallel on different processors. They will be therefore referred to as "local solves" in the sequel. The local

solves can be performed using a direct or an iterative method, and can make use of the symmetry property of $\mathbf{A_1}$ and $\mathbf{A_2}$.

Note that, compared to a method in which the whole domain ($E_1 \bigcup E_2$ in Fig. 1) is not split into subdomains, the DD method presented here has a higher total number of unknowns since the interface unknowns are duplicated among the subdomains. However, this duplication enables the local solves to be performed in parallel on different processors.

The choice of α is an open question. Some theoretical results exist for the case of the Helmholtz equation [5], but not for our field. In this study, we do not attempt at a theoretical determination of α, but refer to numerical tests for its empirical tuning. Note that α can be given a different value at different interfaces. However, we do not investigate this option in this study. In the P_N case with $N > 1$, more than one unknown (in fact as many as spherical harmonics used for the angular expansion) are to be determined at each FE node. Therefore, a different value of α can be taken for the different harmonics. However, in the numerical tests presented below, the best results were achieved by taking for all the harmonics the same value of α (namely $\alpha = 1$ for the considered benchmark).

2.2. Interpretation

As stated above, we solve Eq. (3) iteratively using the BiCGStab method. However, an interpretation of our DD method requires to consider a more basic iterative approach. The most basic method to solve (3) iteratively consists in doing

$$\begin{pmatrix} A^{11} & A^{1\,\Gamma} & 0 & 0 \\ A^{\Gamma\,1} & A_1^{\Gamma\,\Gamma} + I^\alpha & 0 & 0 \\ 0 & 0 & A^{22} & A^{2\,\Gamma} \\ 0 & 0 & A^{\Gamma\,2} & A_2^{\Gamma\,\Gamma} + I^\alpha \end{pmatrix} \begin{pmatrix} \varphi^1 \\ \varphi^{\Gamma,\,1} \\ \varphi^2 \\ \varphi^{\Gamma,\,2} \end{pmatrix}^{\mathbf{n+1}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -A^{\Gamma\,2} & -A_2^{\Gamma\,\Gamma} + I^\alpha \\ 0 & 0 & 0 & 0 \\ -A^{\Gamma\,1} & -A_1^{\Gamma\,\Gamma} + I^\alpha & 0 & 0 \end{pmatrix} \begin{pmatrix} \varphi^1 \\ \varphi^{\Gamma,\,1} \\ \varphi^2 \\ \varphi^{\Gamma,\,2} \end{pmatrix}^{\mathbf{n}} + \begin{pmatrix} Q^1 \\ Q_1^{\Gamma\,\Gamma} + Q_2^{\Gamma\,\Gamma} \\ Q^2 \\ Q_1^{\Gamma\,\Gamma} + Q_2^{\Gamma\,\Gamma} \end{pmatrix}. \quad (5)$$

Following [6], Eq. (5) can be interpreted as the matrix formulation of an algorithm whose continuous version is

$$\mathcal{A}\Psi^{\mathbf{n+1}} = Q \quad \text{in } E^1 \text{ and } E^2, \quad (6)$$

$$(\mathcal{A}^\Gamma + \mathcal{I}^\alpha)\Psi_{E^1}^{\mathbf{n+1}} = (\mathcal{A}^\Gamma + \mathcal{I}^\alpha)\Psi_{E^2}^{\mathbf{n}} \quad \text{on } \Gamma, \quad (7)$$

$$(\mathcal{A}^\Gamma + \mathcal{I}^\alpha)\Psi_{E^2}^{\mathbf{n+1}} = (\mathcal{A}^\Gamma + \mathcal{I}^\alpha)\Psi_{E^1}^{\mathbf{n}} \quad \text{on } \Gamma, \quad (8)$$

where $\mathcal{A}$ and Q are as in Eq. (1), the operator $\mathcal{A}^\Gamma$ is the part of $\mathcal{A}$ acting on Γ, and the operator $\mathcal{I}^\alpha$ multiplies by the appropriate α value. The interpretation of algorithm (6)-(8) is that, at each iteration,

- the considered problem (1) is solved in each domain (Eq. (6)),
- the boundary condition in one domain is given as a function of the unknown Ψ in the neighboring domain at the previous iteration (Eq. (7) and (8)).

The boundary conditions at domain interfaces is the way information is exchanged between domains in the iterative process. To illustrate these interface conditions, we further simplify our example by considering the Poisson equation, namely the case where $\mathcal{A} = \Delta$, the Laplacian operator. In this case, the integration by parts performed to obtain the weak form (standard in the FE theory) yields $\mathcal{A}^\Gamma = \partial_n$, that is, $\mathcal{A}^\Gamma$ is the normal derivative on the interface. If we consider a unique value of α along an interface, the interface condition enforces the continuity of

$$\partial_n \Psi + \alpha \Psi$$

through interfaces. This is a so-called Robin-type condition. At each iteration, point values as well as normal derivatives of the unknown Ψ are transferred between neighboring domains, with the parameter α giving the proportionality between "Dirichlet" (point values) and "Neumann" (normal derivatives) data. Such procedure was shown to be convergent by P.L. Lions [3].

When $\mathcal{A}$ is the Boltzmann transport operator, the situation is more complicated since $\mathcal{A}^\Gamma$ also involves the angular variable. However, the idea of the parameter α giving the proportionality between "Dirichlet" and "Neumann" data, remains valid.

2.3. Non-conforming Finite Elements

In two dimensions, we now consider the NC_4 non-conforming FE, defined on a rectangle with 4 nodes placed each in the middle of an edge. This element differs from the lowest-order conforming Lagrangian FE, whose 4 nodes are each placed on a vertex. (In the finite element literature, the lowest-order conforming Lagrangian FE is known as "Q_1", and the NC_4 non-conforming FE as "rotated Q_1" element.) Fig. 2 depicts the two-domain configuration when NC_4 elements are used.

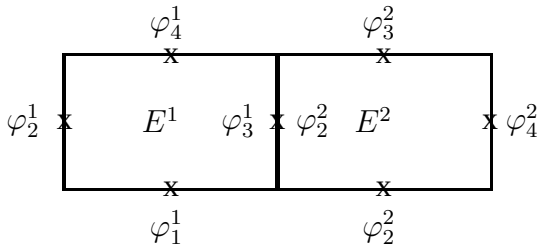

Figure 2. A two-domain example, where each domain is made out of one non-conforming FE. The two domains are coupled by enforcing that φ_3^1 be equal to φ_2^2.

In three dimensions, we equivalently consider the NC_6 non-conforming FE, defined on a rectangular parallelepiped (cube in the simplest case) with 6 nodes placed each in the middle of a

face.

Applying the FE method separately in E^1 and E^2, and following the procedure described in section 2.1, yields a linear system similar to the one in Eq. (3). Non-conforming FEs appear particularly appropriate for our DD method, because placing the nodes away from the vertices avoids having interface nodes belong to more than 2 elements at a time. In turn, each FE node belongs at most to only two domains. This simplifies the coding of data transfer between domains. Indeed, this must be compared to what happens when conforming FEs are used, that is when certain nodes can belong to as much as 4 (in 2-D) or 8 (in 3-D) domains. Moreover, the use of non-conforming FEs has been showed to be valuable in P_N neutronic calculations [7].

3. APPLICATION TO NEUTRON TRANSPORT

We now consider $\mathcal{A}$ to be the linear Boltzmann transport operator in its second-order even-parity formulation. This formulation has the advantage of being self-adjoint. Moreover, we consider criticality eigenvalue calculations, that we solve using the well-known power iteration method. At each "outer" iteration of the power method, we solve a linear system of the form (1) using the above described DD method. Thus, the BiCGStab iterations are here the "inner" iterations of the classical outer/inner iteration procedure.

As stated in section 2.1, our DD method consists in iteratively solving a linear system whose matrix has symmetric diagonal blocks of the form given by Eq. (4). The local solves, i.e., the "inversions" of these diagonal blocks, are performed in parallel on different processors. Numerous methods can be considered to carry out these local solves. Here, we use either the *direct* solver library SUPERLU [8], or the *iterative* solver library SPARSELIB [9]. The SUPERLU library first factorizes the system matrix into a LU form (i.e., the product of a lower triangular L by an upper triangular U matrix), and then solves the resulting two triangular systems. The SPARSELIB library performs a preconditioned conjugate gradient (PCG) calculation. That is, it first computes an approximate LU factorization known as ILU (i.e., incomplete LU), and then uses this ILU factorization as preconditioner in a Krylov-type iterative method, namely the conjugate gradient method in our case.

The parallelization is performed using Message Passing Interface (MPI) on the Forschungszentrum Karlsruhe "CampusGrid" Linux cluster.

3.1. The Takeda 1 benchmark

We here consider the 3-D Takeda 1 benchmark [10]. The 3-D Takeda 1 benchmark consists of a small LWR quarter-core of cubic shape, depicted in figure 3. Note that we consider the 'case 1' of this benchmark, namely when the zone 3 does not contain any control rod. The presence of a low-density zone in the voided guide tube makes the test more demanding - in fact, it makes the simplified transport method SP_N inappropriate to handle this benchmark. We here use the P_N method for the angular discretization, namely we use spherical harmonics to expand the angular dependence of the neutron flux.

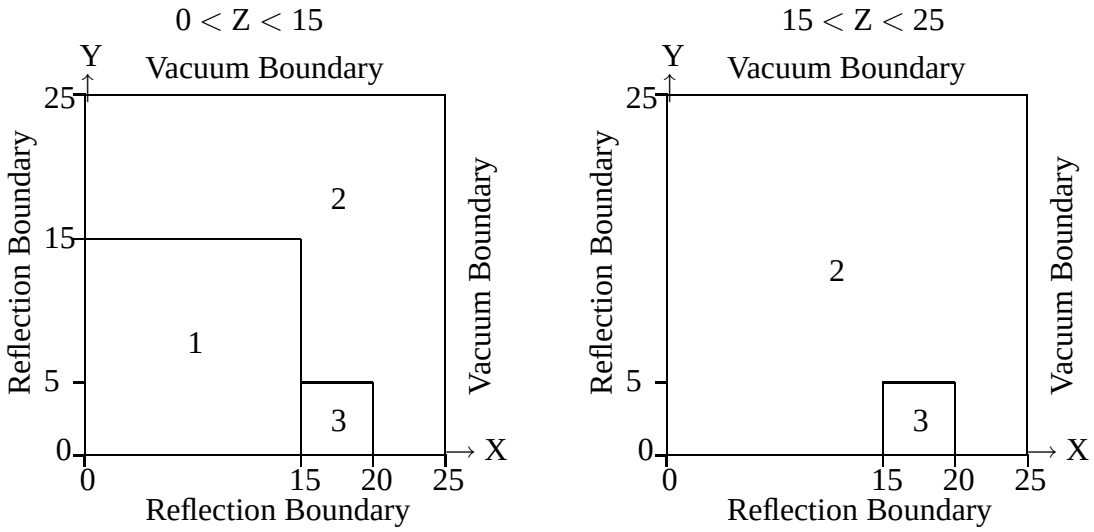

Figure 3. Geometry of the 3-D Takeda 1 benchmark. Zone 1 = core, zone 2 = reflector, zone 3 = low-density zone ('void'). The 2-group cross-section values can be found in [10]

For the spatial discretization, we use the above described NC_6 non-conforming finite elements, i.e., cubes having one node in the middle of each face. The capacity of these FEs to accurately solve the Takeda 1 benchmark has been assessed previously [7]. For all the numerical results presented below, a $25 \times 25 \times 25$ FE mesh was used, which represents 48750 nodes using the NC_6 FEs. These nodes are distributed (with interface nodes duplicated) among the different domains according to geometrical domain decompositions that we now describe.

We consider several geometrical domain decompositions of this benchmark, ranging from $2 \times 2 \times 2$ to $5 \times 5 \times 5$ domains. The domain interfaces follow planes parallel to the cube faces, trying to achieve a good compromise between load balancing and physical homogeneity of the domains. More precisely, here are the coordinates of the interface planes for the different decompositions that we consider:

- $2 \times 2 \times 2$: X = 15; Y = 15; Z = 15
- $3 \times 3 \times 3$: X = 15 and 20; Y = 5 and 15; Z = 5 and 15
- $4 \times 4 \times 4$: X = 8, 15 and 20; Y = 5, 15 and 20; Z = 8, 15 and 20
- $5 \times 5 \times 5$: X = 5, 10, 15 and 20 (same for Y and Z).

The $2 \times 2 \times 2$ yields domains of different sizes and in-homogeneous compositions. The decompositions into $3 \times 3 \times 3$ and $4 \times 4 \times 4$ give rise to domains with homogeneous compositions but different sizes. Only the $5 \times 5 \times 5$ decomposition yields domains of equal sizes and homogeneous compositions. Therefore, this last decomposition has the best "quality". However, since increasing the number of domains increases the number of duplicated interface FE nodes, the $5 \times 5 \times 5$ decomposition yields the biggest total number of unknowns.

3.2. Numerical Results

Sections 3.2.1 and 3.2.2 present results obtained using one processor per domain. They respectively address the tuning of the α parameter and the choice of direct or iterative method for the local solves.

Using one processor per domain does not enable proper speed-up evaluations, since using a different number of processor means also using a different decomposition, and therefore not solving the same linear system. We therefore enabled each processor to handle more than one domain. Section 3.2.3 presents the resulting speed-up evaluations, and discusses the advantages of handling several domains with one processor.

3.2.1. Tuning of α

Table 3.2.1 presents P_1 results obtained with local solves performed with the direct SUPERLU library [8]. The value of the α parameter was tuned. It appears that its influence is rather limited, and that taking $\alpha = 1$ gives the best results (except for an insignificant time improvement using a higher value in the 64 domain case).

Calculations (not detailed here) with the P_3 approximation were also performed, using different values of α for the P_1 and P_3 harmonics (one P_1 and five P_3 harmonics are present in the even-parity P_3 approximation). Again, the use of $\alpha = 1$ for all the harmonics proved the most efficient.

Table I. P_1 results showing the influence of the α parameter. The number of outer ("power") iterations, as well as the total number of inner (BiCGStab) iterations, is given. The local solves were performed using the SUPERLU library.

# domains = # processors	α_{P_1}	# inner (outer) iterations	wall-clock time (s)
1 (sequential SUPERLU)		26 (13)	1107
8 ($2 \times 2 \times 2$)	.1	250 (92)	58
	1	145 (28)	53
	10	322 (16)	66
27 ($3 \times 3 \times 3$)	.1	($>$ 100 outers)	
	1	211 (22)	16
	10	339 (28)	17
64 ($4 \times 4 \times 4$)	.1	($>$ 100 outers)	
	1	170 (30)	13
	10	347 (12)	12
125 ($5 \times 5 \times 5$)	.1	($>$ 100 outers)	
	1	160 (25)	9
	10	372 (17)	13

3.2.2. Direct vs. iterative method for the local solves

Beside the direct SUPERLU method, we also considered to perform the local solves using the SPARSELIB [9] implementation of the preconditioned conjugate gradient (PCG) with incomplete lower-upper (ILU) preconditioning. Our numerical tests show that this PCG-ILU method does not perform well (in fact, diverges) when applied within the low-density zone (zone 3) of the considered Takeda 1 benchmark. This is due to the well-known fact that the matrices arising from such zones are ill-conditioned (and might actually be overcome by using a more robust implementation of the preconditioner, or by using another preconditioner). The direct SUPERLU solver appears more robust (but slower as we will see), and manages to handle these matrices. Therefore, two sets of numerical tests were performed: one with the direct SUPERLU solver applied to all diagonal blocks, and one mixing both the direct (for domains containing at least part of the low-density zone) and the iterative PCG-ILU solvers (for all other domains). Moreover, since the local solves are wrapped-up in the BiCGStab iterations, which themselves are wrapped-up in the outer power iterations, the local solves, when performed iteratively, do not need to be done with high precision. A convergence criteria of 10^{-2} on the PCG iterations proved to yield the best results, presented in table 3.2.2.

Table II. P_3 results ($\alpha = 1$ for all harmonics) with two different methods for the local solves .

	direct method		direct + PCG method	
# domains = # processors	# inner (outer) iterations	wall-clock time (s)	# inner (outer) iterations	wall-clock time (s)
8 ($2 \times 2 \times 2$)	159 (33)	6465	166 (34)	2713
27 ($3 \times 3 \times 3$)	192 (31)	1194	259 (48)	474
64 ($4 \times 4 \times 4$)	205 (40)	214	216 (41)	124
125 ($5 \times 5 \times 5$)	205 (40)	36	216 (41)	24

Table 3.2.2 shows that the use of the PCG iterative method where applicable (i.e., outside of the low-density zone) increases the number of iterations, but significantly decreases the resolution time. This time gain is particularly significant when a small amount of processors is used, that is, when the diagonal blocks to factorize are relatively large. This is due to the well-known fact that the iterative methods perform better than the direct ones on large systems.

Notes on the results in Tables 3.2.1 and 3.2.2. It is important to note that no proper speed-up estimations can be done from the results in Tables 3.2.1 and 3.2.2, since changing the decomposition means solving another linear system. The fact that the decrease in the wall-clock time is often larger than the increase in the number of processors (the "pseudo speed-up" is bigger than one) is therefore not surprising.

Note also that these results where obtained without taking advantage of the diagonal block symmetry, because neither the SUPERLU nor the SPARSELIB library offer a symmetric storage option. The comparison results presented here would however most probably remain qualitatively valid if such option were available, since both techniques would then be improved.

3.2.3. Speed-up evaluations, and the gain in handling several domains with one processor

Tables 3.2.3 and 3.2.3 present results for the $2 \times 2 \times 2$ and $5 \times 5 \times 5$ decompositions (with the P_3 angular approximation), obtained after enabling a single processor to handle more than one domain. As already stated, the $5 \times 5 \times 5$ decomposition is of much better quality than the other ones, in that it yields domains of equal sizes and homogeneous compositions, which in turn optimizes load balancing and local solve handling (as described above, the direct solver is used in domains containing at least part of the low-density zone, and the iterative solver in the other domains). With respect to this, the $2 \times 2 \times 2$ decomposition has the worst quality.

Table III. Speed-up estimations for the $2 \times 2 \times 2$ decomposition.

# processors	wall-clock time (s)	speed-up	efficiency
1	7611	1	1
2	5375	1.42	0.71
4	4174	1.82	0.46
8	2713	2.81	0.35

Table IV. Speed-up estimations for the $5 \times 5 \times 5$ decomposition.

# processors	wall-clock time (s)	speed-up	efficiency
1	1222	1	1
5	267	4.58	0.92
25	65	18.8	0.75
50	39	31.33	0.63
75	35	34.91	0.47
125	24	50.92	0.41

Due to its better quality, the $5 \times 5 \times 5$ decomposition yields better speed-up evaluations than the ones achieved with the $2 \times 2 \times 2$ decomposition. The most important observation must however be done by comparing the wall-clock times. For the same quality reason, the $5 \times 5 \times 5$ decomposition yields much better wall-clock times than the other decompositions. One can for instance note that the solution time using 8 processors with the $2 \times 2 \times 2$ decomposition (2713 s. - see Table 3.2.2

and 3.2.3), as well as the solution time using 27 processors with the $3 \times 3 \times 3$ decomposition (474 s. - see Table 3.2.2), can be greatly improved by using the $5 \times 5 \times 5$ decomposition with only 5 processors (267 s. - see Table 3.2.2) . Similarly, the solution time using 64 processors with the $4 \times 4 \times 4$ decomposition (124 s. - see Table 3.2.2), can be improved by using the $5 \times 5 \times 5$ decomposition with only 25 processors (65 s. - see Table 3.2.3). This stresses the importance of having a code that enables the handling of several domains by one single processor: a better-quality decomposition can then be used without requiring a higher number of processors. Moreover, the gain of using a finer (and better) decomposition is, at least for this benchmark, much higher than the additional cost induced by the increased number of duplicated nodes.

4. CONCLUSIONS

The encouraging numerical results presented here show the potentialities of the proposed DD method. The quality of the decomposition, in terms of domain homogeneity and size similarity, appears of great importance for the success of the method. Enabling one processor to handle more than one domain yields proper speed-up evaluations, and allows the use of a better-quality decomposition without requiring a higher number of processors. This ability is believed to be a novelty compared to previous investigations in the neutronic field [1, 2, 11].

The tuning the α parameter showed that, for the considered benchmark, taking the same value of α for all the spherical harmonics was the most appropriate.

Performing the local solves iteratively with a loose convergence criteria proved to significantly decrease the computing time, especially on large domains. However, the PCG-ILU iterative solver did not prove robust enough to handle the ill-conditioned matrices arising from a low-density zone. More robust implementations or other preconditioning techniques at the local level should therefore be investigated to remedy this fact.

As other future prospect, a systematic load-balancing strategy [11] should be implemented. Moreover, the quasi-reflected interface conditions developed by Lewis [12] for the spherical harmonic discretization, could prove valuable in the present framework in that it could reduce the exchanges between processors.

Finally, while non-conforming FEs appear particularly appropriate for our DD method, it could also be proved of interest for conforming FEs, even though the presence of FE nodes at vertices implies more complicated data exchanges between processors.

ACKNOWLEDGEMENTS

This work originates from post-doctoral work supported by the French Petroleum Institute.

REFERENCES

[1] C.R.E. de Oliveira, C.C. Pain, and J.H. Goddard. “Parallel domain decomposition methods for large-scale finite element transport modelling.” In *Proc. Int. Conf. Math. Comp. Reactor Physics and Environmental Analysis of Nuclear System*, Portland, Oregon, April 30 - May 4, 1995.

[2] P. Guérin, A.M. Baudron, and J.J. Lautard. "Domain decomposition methods for core calculations using the MINOS solver." In *Proc. Joint International Topical Meeting on Mathematics & Computation, Supercomputing in Nuclear Applications (M & C + SNA 2007)*, Monterey, CA, U.S.A., April 15-19 2007.

[3] P.L. Lions. "On the Schwarz alternating method III: a variant for nonoverlapping subdomains." In *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 202–223, Philadelphia, 1990. SIAM.

[4] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.

[5] Martin J. Gander, Frédéric Magoulès, and Frédéric Nataf. "Optimized Schwarz methods without overlap for the Helmholtz equation." *SIAM J. Sci. Comput.*, 24(1):38–60, 2002.

[6] F. Nataf. "Domain decomposition methods for non-symmetric problems." In *Computational Mechanics using High Performance Computing - Proceedings of the Third Euro-Conference on Parallel and Distributed Computing for Computational Mechanics*, pages 185–198, Weimar, Germany, March 20-25 1999. Saxe-Coburg Publications.

[7] S. Van Criekingen. "A 2D/3D cartesian geometry non-conforming spherical harmonic neutron transport solver." *Annals of nuclear energy*, 34(3):177–187, 2007.

[8] James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. "A supernodal approach to sparse partial pivoting." *SIAM J. Matrix Analysis and Applications*, 20(3):720–755, 1999.

[9] J. Dongarra, A. Lumsdaine, R. Pozo, and K. Remington. "A sparse matrix library in C++ for high performance architectures." pages 214–218, 1994.

[10] T. Takeda, M. Tamitani, and H. Unesaki. "Proposal of 3D neutron transport benchmark." NEACRPA-953 rev.1, Department of nuclear engineering, Osaka University, Japan, 1989.

[11] A. Pattnaik and C.R.E. de Oliveira. "A load balancing strategy for the radiation transport code EVENT." In *Proc. Joint International Topical Meeting on Mathematics & Computation, Supercomputing in Nuclear Applications (M & C + SNA 2007)*, Monterey, CA, U.S.A., April 15-19 2007.

[12] E.E.Lewis. "Quasi-reflected interface conditions for variational nodal lattice calculations." In *Proceedings of the PHYSOR 2006 ANS Topical Meeting on Reactor Physics*, Vancouver, BC, Canada, 2006.