The P_N form of the Neutron Transport Problem Achieves Linear Scalability Through Domain Decomposition

K. Assogba and L. Bourhrara
Université Paris-Saclay, CEA, Service d'Études des Réacteurs et de Mathématiques Appliquées
91191 Gif-sur-Yvette, France

kenneth.assogba@cea.fr, lahbib.bourhrara@cea.fr

ABSTRACT

Due to strong coupling between the angular moments, the spherical harmonics (P_N) formulation of the neutron transport equation has been neglected in favor of the discrete ordinates (S_N) form.

In this work, we target large-scale neutron transport simulation using a combined discontinuous Galerkin (DG) – spherical harmonics approximation. We leverage the benefits of DG discretization to wrap the previous developed solver, called `NYMO`, in a canonical ghost-mesh based domain decomposition framework. The developed solver handles unstructured, curved and non-conforming meshes with vacuum and reflection as boundary conditions. Robustness, strong and weak scalability experiments have been conducted on the CEA's pre-exascale system Topaze. We reach and maintain a strong scaling efficiency of 100 % up to 4096 cores and 80 % up to 8192 cores. In particular, a calculation with 913 million degrees of freedom is performed in 101 seconds. Thus outperforming previously published results for P_N transport as well as many S_N and SP_N solvers.

KEYWORDS: domain decomposition, discontinuous Galerkin, spherical harmonics, MPI, OpenMP

1. INTRODUCTION

The development of new computing architectures, particularly at the exascale era (systems capable of at least 10^{18} floating point operations per second) is reshaping the landscape of nuclear reactors physics. Such systems enables to model the entire lifecycle of a nuclear reactor at pin-cell scale, while engineers will be able to iterate rapidly on the design of advanced reactor concepts. To fully exploit the capabilities of these new machines, it is necessary to develop scalable approximation methods and software code.

1.1. Neutron transport at scale

Research on high-performance numerical methods in neutronics has been focused on the S_N method. Proposed in the 90s at Los Alamos National Laboratory, the Koch-Baker-Alcouffe (KBA) algorithm [1] quickly established as a way to perform S_N sweeps efficiently. In 2017, using this algorithm, the `Denovo` S_N code [2] shows a strong scalability efficiency around 90% at 144 cores and $\sim$70% at 3600 cores using S_{16} approximation. The weak scalability efficiency provided is greater than 100% up to 40k cores compared to the computation time on 4096 cores. More recently and in another register, Vermaak et al. [3] propose a parallel sweeping method able to handle meshes with cyclic dependencies. The resulting solver named `Chi-Tech`, achieves a weak scaling efficiency $>$80% on more than 100k processes on a problem with 87.7 trillion unknowns. For strong scaling on a mesh with cyclic dependencies, the efficiency is 58% or 34% at 2048 process depending on the partitioning strategy used.

Inspired by parallel methods for elliptic problems, a concomitant approach has been developed, using domain decomposition à la Schwarz. In [4], Jamelot and Ciarlet propose a Schwarz iterative algorithm with Robin interface conditions for the SP_N diffusion model. On a 3D 900 MWe pressurized water reactor core, the authors show that the method can achieve good parallel performance on up to 128 cores, with an efficiency of around 80%. In [5], the authors propose a domain decomposition method based on a block-Jacobi algorithm for `Minaret` the APOLLO3 S_N transport core solver. However, the proposed method suffers from convergence penalty in terms of both computing time and number of iterations [6]. For a 2D core with 675k mesh cells the efficiency is between 32 and 37% for the calculations performed with MPI.

On the P_N side, in the mid 90s, de Oliveira et al. [7] describe a multi-block-explicit domain decomposition method. More recently, `PARAFISH` [8], a P_N–finite-element neutron code employ an algebraic domain-decomposition to solve the second-order even-parity form of the Boltzmann equation. On a benchmark with P_7 approximation with a total of 3,150,000 unknowns the efficiency is around 28% at 125 cores.

Efficiency being a good common metric, Table 1 summarizes strong scaling results for `Minos`, `Minaret` and `PARAFISH`. These results are compared to those obtained by `NYMO` on a problem with 913 million unknowns. `NYMO` maintains near perfect efficiency with ease.

Table 1: Comparison of strong scalability of parallel neutronics finite element solvers. The data reported are from the best published results for each solver.

	Efficiency % (#core)			Model
`PARAFISH` [8, §3.3]	53 (5)	37 (25)	28 (125)	Transport P_N
`Minaret` [5, §3.3]	64 (13)	32 (27)	37 (39)	Transport S_N
`Minos` [4, §7.4]	88 (32)	88 (64)	85 (128)	Diffusion SP_N
`NYMO`	110 (2048)	113 (4096)	82 (8192)	Transport P_N

1.2. Target and outline

This work reports on the development of a simple domain decomposition method for a spherical harmonics – discontinuous Galerkin transport solver. A core component of this approach is the block structure of the matrices arising from the discretization. The off-diagonal blocks then describe the interactions between adjacent mesh elements and in fact allow the coupling of adjacent subdomains through the transmission of the upwind flux. This coupling is easily carried out at the matrix-vector product level. The MPI communications are made in a non-blocking way, thus enabling the overlap of communications and calculations. Due to the choice of the finite element basis functions, the elementary matrices of identical elements, given a translation, are the same, thus allowing to implement the solver in a matrix assembly-free fashion.

By positioning at the algebraic scale, non-conforming and unstructured meshes with curved faces can be easily handled. We overcome the problem of partitioning these types of meshes by implementing a partitioning strategy based on Hilbert space filing curves [9], along with a naive partitioning, using the mesh numbering. Finally, the implementation is minimally intrusive and the parallel execution requires no additional effort from the end-users.

These developments have been conducted within `NYMO` [10–12], the multipurpose P_N-transport solver of the reactor physics platform `APOLLO3`® [13].

Numerical experiment have been conducted on the pre-exascale cluster Topaze. These tests point out that the method achieves almost perfect linear scaling. The asynchronous communications are very effective, thus the overlap between communications and computations is near perfect. The solver is moreover robust, as the variations on the obtained solutions compared to the sequential solver are close to the machine precision.

This article is organized as follows: in section 2 we present the multigroup transport model and the variational formulation used. Then in section 3 we describe the discretization, in particular the elements of interest for the parallelism. In section 4 we describe the important steps of the algorithm. Finally in section 5 we evaluate its robustness, weak and strong scalability behavior.

2. MODEL

The neutron transport equation describes the neutron distribution in a medium, taking into account scattering, fission and external sources. One wishes to determine the multigroup neutron flux $u^g(x, \omega)$ at any point

(x, ω) of the phase space $X = D \times \mathbf{S}^2$. The space variable is denoted x and D is the spatial domain. The angular direction is denoted ω and $\mathbf{S}^2$ is the angular domain. The subscript g is the energy group, and unless necessary is omitted in the following. In the phase space X with an incoming flux f through the inflow boundary Γ_-, the problem is written,

$$\omega \cdot \boldsymbol{\nabla} u + \sigma u = q \quad \text{in} \quad X \tag{1a}$$
$$u = f \quad \text{on} \quad \Gamma_- \tag{1b}$$

where σ denotes the macroscopic total cross section and q the neutron source taking into account scattering (Hu) and possibly fission (Fu) and external neutron sources (q_{ext}). The eigenvalue problem consists in finding the eigenvalue λ associated with u when there is no external source.

Let V denote the variational space, and $v \in V$ a test function. The outgoing boundary is denoted Γ_+. We multiply (1a) by $(v + \frac{1}{\sigma}\omega \cdot \boldsymbol{\nabla} v)$ and then integrate over the phase space. After using Green's formula (see [14]) and boundary condition (1b), one obtain the variational problem:

$$\text{find } u \in V \ \text{ such that } \ a(u, v) = L(v), \ \ \forall v \in V. \tag{2}$$

With

$$a(u, v) = \int_X \left(\frac{1}{\sigma} (\omega \cdot \boldsymbol{\nabla} u)(\omega \cdot \boldsymbol{\nabla} v) + \sigma u v \right) + \int_{\Gamma_+} uv(\omega \cdot n), \tag{3}$$

$$L(v) = \int_X q \left(v + \frac{1}{\sigma} (\omega \cdot \boldsymbol{\nabla} v) \right) - \int_{\Gamma_-} fv(\omega \cdot n). \tag{4}$$

Under assumptions about data f, q and σ, this problem is well posed in the sense of Hadamard [14]. That is to say that it admits a unique solution and this solution depends continuously on the data of the problem. In the following, let us discretize the problem (2).

3. DISCRETIZATION

The discontinuous Galerkin – spherical harmonics numerical scheme for the linear Boltzmann equation has been introduced in [10], [12] and [11]. Let us recall here the key elements we need for the parallel implementation of the solver. The spherical harmonics method or $\mathrm{P_N}$ consists in developing with truncation to the order N, the angular flux on the real spherical harmonics basis $y_n^m(\omega)$ [10],

$$u_N(x, \omega) = \sum_{n=0}^{N} \sum_{m=-n}^{n} u_n^m(x) y_n^m(\omega). \tag{5}$$

The unknown resulting from this development u_n^m, called the flux moment, is approached by piecewise polynomials of degree at most k, which we note in the following $\mathbb{P}^k = \text{span}\,\{\varphi_1, \ldots, \varphi_J\}$. In the end,

$$u_{N,k}(x, \omega) = \sum_{n=0}^{N} \sum_{m=-n}^{n} \sum_{j=1}^{J} u_{n,j}^m \varphi_j(x) y_n^m(\omega). \tag{6}$$

The approximation space thus constructed is $V_h = \text{span}(\varphi_j y_n^m)$. The discrete problem is:

$$\text{find } u \in V_h \ \text{ such that } \ a(u, v) = L(v), \ \ \forall v \in V_h. \tag{7}$$

The incoming flux and the sources are developed in the same way. The test function is replaced by $v = \varphi_i(x) y_l^k(\omega)$. In each mesh cell D_r, the result is a local linear system of unknowns $\mathbf{u_r} = \left(u_{n,j}^m \right) \Big|_{D_r}$

$$A_r \mathbf{u_r} = \mathbf{q_r} \tag{8}$$

with

$$A_r \mathbf{u_r} = \left(\frac{1}{\sigma}A_r^0 + \sigma A_r^1 + A_r^+\right)\mathbf{u_r} + \sum_{F\in\partial D_r} A_F^- u_F^\uparrow, \tag{9}$$

$$\mathbf{q_r} = \left(A_r^1 + \frac{1}{\sigma}A_r^2\right)q, \tag{10}$$

and $u_F^\uparrow$ is either the boundary flux f or the flux coming from the neighboring regions of D_r through the internal face F. The elementary matrices are defined as

$$A_r^0(i,n,m,j,l,k) = \sum_{p=1}^{3}\sum_{q=1}^{3}\left(\int_{D_r}\partial_p\varphi_i\partial_q\varphi_j\,\mathrm{d}x\right)\left(\int_{\mathbf{S}^2}\omega_p\omega_q y_n^m y_l^k\,\mathrm{d}\omega\right), \tag{11}$$

$$A_r^1(i,n,m,j,l,k) = \delta_{n,l}\delta_{m,k}\int_{D_r}\varphi_i\varphi_j\,\mathrm{d}x\,, \tag{12}$$

$$A_r^2(i,n,m,j,l,k) = \sum_{p=1}^{3}\left(\int_{D_r}(\partial_p\varphi_i)\varphi_j\,\mathrm{d}x\right)\left(\int_{\mathbf{S}^2}\omega_p y_n^m y_l^k\,\mathrm{d}\omega\right), \tag{13}$$

$$A_r^+(i,n,m,j,l,k) = \sum_{F\in\partial D_r}\int_F\varphi_i\varphi_j\int_{(\omega\cdot n)>0} y_n^m y_l^k(\omega\cdot n)\,\mathrm{d}\omega\,\mathrm{d}s\,, \tag{14}$$

$$A_F^-(i,n,m,j,l,k) = \int_F\varphi_i\varphi_j\int_{(\omega\cdot n)<0} y_n^m y_l^k(\omega\cdot n)\,\mathrm{d}\omega\,\mathrm{d}s\,. \tag{15}$$

The coefficients of these matrices are calculated in an exact manner according to the method described in [10].

3.1. Mesh equivalence class

The base of polynomials $\{\varphi_j\}$ depends on the region of calculation. With a judicious choice of this base [10] the elementary matrices 11-15 are the same for equivalent regions, we mean by equivalent regions that are identical up a translation. Thus by grouping all the regions by packets where each packet contains only equivalent regions, the calculation of the elementary matrices is done only once per packet of equivalent regions and not for each calculation region. This results in substantial gains in storage and simulation time due to efficient cache fetching. The equivalence classes of a small mesh are shown on Figure 1.

In the domain decomposition framework, the solution on each subdomain is coupled to the solution on its neighboring subdomains using ghost cell communication. Three basic components are needed: a mesh partitioner, distributed sparse matrix-vector multiplication (SpMV) and dot-products. In the next, we present each ingredient and the recipe to combine them.

4. IMPLEMENTATION

Modern computing clusters are made up of independent computing nodes linked together by a fast interconnection network. Programmatic access to these – so-called distributed – systems can be achieved through MPI (Message Passing Interface). The MPI standard defines a library of functions that allow parallel programming by exchanging messages. Within each host device having continuous memory addressing (also referred to as NUMA node) it is possible to spread a task across threads through OpenMP compiler directives.

The move to distributed systems requires effective design of data distribution and communication between processes. A straightforward advantage is that it allows extremely large problems to be solved, since each process holds only a small part of the original data. The implementation is simplified in the context of discontinuous Galerkin approximation. In fact, each mesh element is only coupled with its neighboring cells.

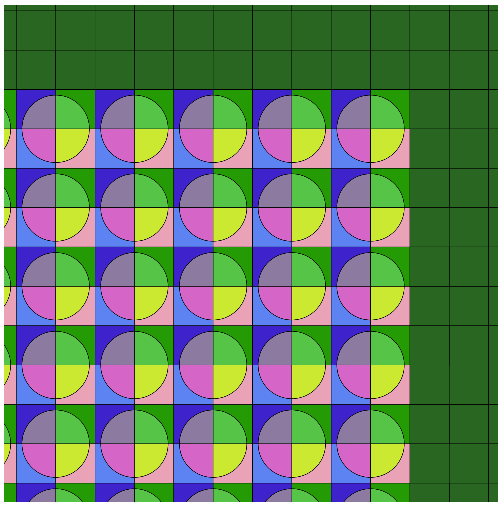

Figure 1: Equivalence classes of a portion of a mesh.

Thus the matrices resulting from this approximation are essentially block diagonal, with off-diagonal blocks coupling neighboring cells. As a corollary, after data distribution, each subdomain only needs to fetch the flux from the cells that share a face with its local cells.

4.1. Mesh Partitioning

Let us denote by n_d the number of MPI process in the global communicator. The parallel resolution start by partitioning the initial mesh into n_d subdomains. One need to guarantee a load balancing between the processes while minimizing the communication between adjacent sub-domains. The load balancing criterion is ensured by imposing that the number of elements inside each subdomain is the same. Two widely used mesh partitioners are Metis [15] and Scotch [16], but they do not provide the capabilities to handle non-conforming meshes with curved elements. We have therefore implemented two partitioning strategies. The first one is based on mesh numbering and is called *simple* partitioning in the following. The second one is based on Hilbert space-filling curves [9] and is referred to as *geometric* partitioning. The advantage of space filling curves is that locality of mesh cells is fairly well preserved. Figure 2 shows the partitioning of a set of 4 assemblies. Let us note here that the numbering is contiguous within an assembly. Thus the naive partitioning also produces harmonious results.

After the partitioning, each process uses the macroscopic cross-sections corresponding to its subdomain. The cells having a part of their stencil straddling two domains, are referred to as ghost or halo cells and are also collected. The rest of the solver proceeds by substituting the global data (mesh and cross-sections) by the local data. These global data can then be freed to save memory space. Finally, each process computes its sub-matrices A_r^0, A_r^1, A_r^2, A_r^+, A_F^- (11) - (15) on the local mesh and a communication matrix A^- is generated from the ghost mesh. All matrices are built in an assembly-free fashion.

4.2. Asynchronous Sparse Matrix Vector Multiplication

When solving the linear system (8) inside the power iteration algorithm, all process work concurrently. The neighboring domains are coupled via the ghost cells, and exchange upwind flux on their common interfaces. At the algebraic level, this exchange is done at each sparse matrix-vector product (SpMV). The use of non-blocking communications allows communication time to be covered by computing time.

The asynchronous matrix-vector product attached to the Krylov solver takes place in three main stages and is described on Algorithm 1. We start by posting an `MPI_Irecv` request to start receiving the incoming flux $x^\uparrow$ from neighboring domains. Then for each neighboring domain, the outgoing portion of the flux is

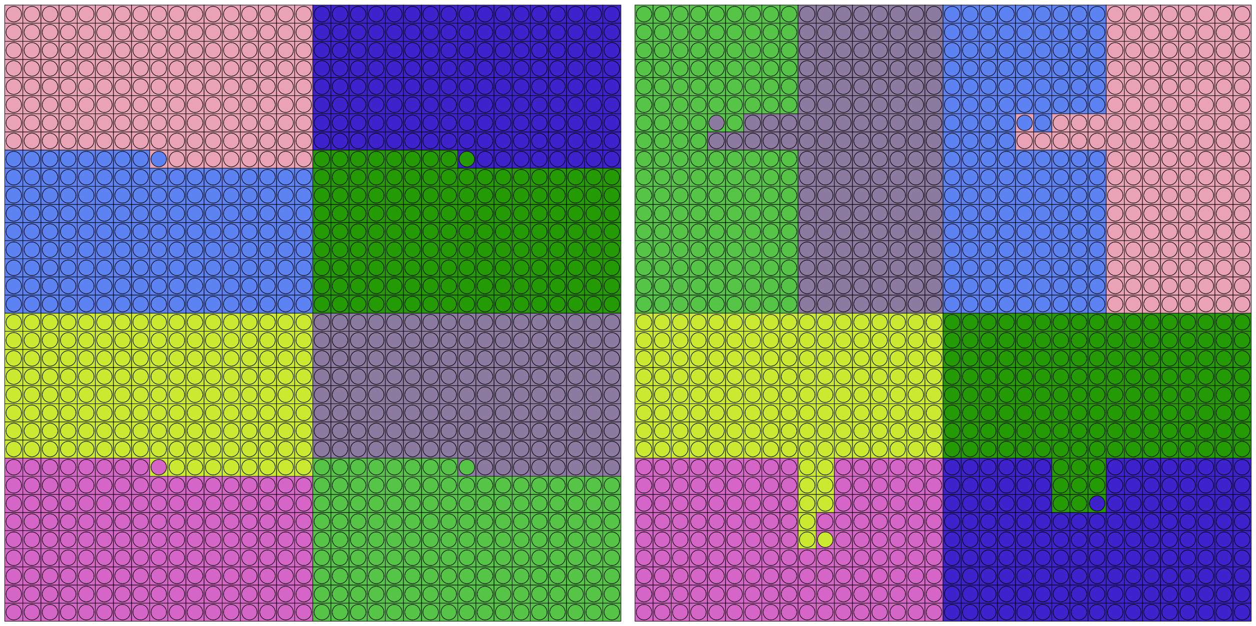
Figure 2: Simple (left) and geometric (right) partitioning of 4 17-by-17 assemblies into 8 domains.

copied into a buffer x_{out} and a `MPI_Isend` request is posted. The calculation starts and the part of the product involving only local degrees of freedom is calculated. Then, one waits for the incoming and outgoing communications to finish. The overlap works very well and the waiting time is usually negligible. Finally, one use the received flux to compute the off-diagonal part of the product.

The copy step is needed because the ghost cells data are not necessary contiguous inside the flux vector. This operation is accelerated using OpenMP and the overhead is negligible. Two other possibilities are (i) to send the data element by elements or (ii) to define a MPI datatype that will handle the stride inside the flux vector. These two options have been tested and are slower than the one chosen. Waiting for outgoing communications to stop is necessary to deallocate requests handler and avoid memory leaks.

Algorithm 1: The general Matrix-Vector product from the point of view of a subdomain.

```
Input: $A_{loc}, x_{loc}, y_{loc}$
1 foreach $i_d \in domain_{neighbors}$ do
2     $reqs_{in} \leftarrow$ MPI_Irecv($x^{\uparrow}$)                         // async
3     Copy outgoing part of $x_{loc}$ in the $x_{out}$ buffer
4     $reqs_{out} \leftarrow$ MPI_Isend($x_{out}$)                        // async
5 $y \leftarrow y + A_{|\mathrm{diag}} x$                // $A_{|\mathrm{diag}} = A^0 + A^1 + A^+ + A^-_{\mathrm{loc}}$
6 MPI_Waitall($reqs_{in}, reqs_{out}$)
7 $y \leftarrow y + A_{|\mathrm{offd}} x^{\uparrow}$                         // $A_{|\mathrm{offd}} = A^-_{\mathrm{comm}}$
```

To close this part, it is useful to note that, scalar products involve a collective reduction operation between all domains.

5. EXPERIMENTS

The numerical method described in Section 3 have been applied to a wide variety of reactor cores and lattice calculation [10–12]. The C5G7 benchmark [17] is used here to assess the performance of the implementation of the approach described in Section 4.

C5G7 is a reduced Pressurized Water Reactor designed to evaluate the ability of transport solvers to perform core calculations without spatial homogenization. Presented on the Figure 3, the quarter-core is made up of 4 assemblies surrounded by a reflector. Each assembly is made of a grid of 17 by 17 pins. There are 4 axial slices, including 3 fuel slices and in the case presented here, Rodded B, the control rods are inserted ⅔ in the UOx and ⅓ in the MOx. The results of shared memory calculations are presented in [12], here we focus on the distributed solver.

For all the experiments carried out, the discretization chosen is of order P_6 in angle and linear $\mathbb{P}^1$ in space. The Krylov solver used is BiCGSTAB [18] with a tolerance (if not specified) of 10^{-4} and the point Jacobi preconditioner. The tolerance on the fission source and on the k_{eff} during outer iterations is 10^{-5}. The flux is calculated and stored in single precision, and the k_{eff} in double precision.

Three experiments are conducted: robustness, strong scalability and weak scalability. All calculations are performed on the Milan partition of the CEA's petaflopic cluster Topaze. Milan has 864 nodes, each housing two AMD EPYC 7763 2.45 GHz sockets, equipped with 64 cores each. The nodes are linked together by a high-performance InfiniBand HDR-100 network. The shared-memory only calculation are performed on one socket. For hybrid MPI + OpenMP jobs, each MPI process is binded to one socket.

The software tool chain consists of Intel C++ Compiler and Intel MPI in their versions 2019.5.281. All developments are available in `APOLLO3®`.

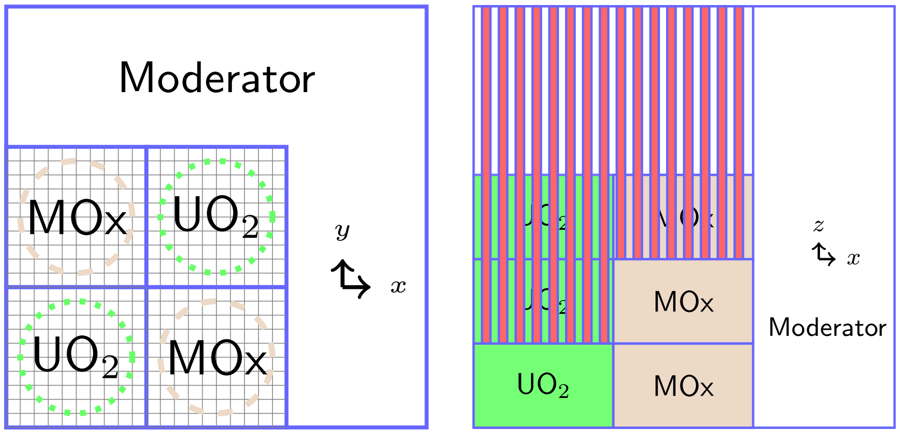

Figure 3: Radial (left) and axial (right) sections of the C5G7 core.

5.1. Robustness

The robustness analysis is done in two steps. First, we check that the `NYMO` calculation without decomposition is consistent with the Monte Carlo reference solution provided in [17]. These results are provided in the table 2. Here, each of the 3 fuel axial slices is divided into 8, the top layer (moderator) is divided into 16, bringing the number of cells to 832,320. All the tolerances (on outer iterations, k_{eff}, thermals, and matrix solver) are fixed at 10^{-5}. The average (avg), root mean square (rms) and mean relative (mre) pin power errors are defined in [12, §IV]. The maximum error on the power is 1.7%, it is consistent with what is expected when comparing a deterministic code to a Monte Carlo code.

In a second stage, we evaluate the variation on the k_{eff}, the pin power and the number of outer iterations when increasing the number of subdomains. The objective is to verify that the error on the eigenvalue and on the flux in parallel is negligible compared to the sequential solution. The simple partitioning is used. For the eigenvalue, the calculated error is the relative error with respect to the solution without domain decomposition $\delta = {}^{(\lambda_{n_d} - \lambda_{n_0})}/_{\lambda_{n_0}}$ in pcm (10^{-5}) .

The table 3 shows the results of the robustness experiment. We observe that there is no variation on the k_{eff} and on the number of iterations. The maximum bias on the fission rate is 4 pcm. Since the flux is computed and stored in simple precision, these spreads are negligible. However, they can be justified by the fact that in parallel, the same operations are performed as in sequential, but in a different order. As machine arithmetic is not associative, these small variations are observed. The method is robust, in the sense that the variations on the solution obtained when the number of subdomains is increased are of the order of machine precision. This offers strong guarantees on the behavior of the parallel solver, which is essential for an industrial use.

Table 2: Eigenvalue and pin power distribution error for Rodded B case of C5G7 problem.

	k_{eff}	error (pcm)	avg (%)	rms (%)	mre (%)	max (%)	#outer
MCNP	1.07777	-	-	-	-	-	-
NYMO	1.07864	80	0.351	0.447	0.340	1.752	31

Table 3: Robustness experiment. The eigenvalue and fission rates obtained with domain decomposition are compared with a reference calculation without decomposition.

n_d	k_{eff}	error (pcm)	avg (%)	rms (%)	mre (%)	max (%)	#outer
0[*]	1.07864	-	-	-	-	-	31
2	1.07864	0	0.0006	0.0008	0.0005	0.0038	31
4	1.07864	0	0.0008	0.0009	0.0007	0.0025	31
8	1.07864	0	0.0010	0.0013	0.0009	0.0040	31

5.2. Strong scalability

The objective of strong scalability is to evaluate the potential reduction in time when more computing resources are available. Thus, the number of computing units is increased while the problem size remains constant. We measure speedup $s(d) = {}^{t(0)}/_{t(d)}$ and efficiency $e(d) = {}^{s(d)}/_{n_d}$, $t(d)$ being the elapsed time for n_d subdomains. We use this test to compare the *simple* and *geometric* partitioning. Each axial slice is divided into 4, bringing the number of cells in the mesh to 665,856 and the total number of degrees of freedom to 913,554,432.

Table 4 shows the results of this experiment. We go up to 128 domains, which corresponds to 8192 cores. We first notice that whatever the chosen partitioning, the measured time decreases linearly. In addition, there is no real difference in performance between simple and geometric partitioning here. The efficiency is almost perfect up to 64 domains, then decreases a little at 128 domains. At 128 domains, each domain own 5 216 mesh cells, it can be reasonably assumed that the workload is low in this case. In addition, communication time represents around 50% of the total time elapsed. The bulk of the communication time is dominated by global reduction operations.

5.3. Weak scalability

Here, the number of subdomains is increased from 16 to 32 then 64, while keeping the problem size per process constant. To do this, the mesh is refined axially. Starting from the unrefined mesh, each plane is divided into 2 then into 4. The performance metric used is the weak scaling efficiency, which is defined as the ratio of the execution time on 16 process to the execution time on n_d process. The goal is to achieve an efficiency close to one, which indicates that the algorithm scales effectively as more process are added.

Table 5 presents the results of this experiment. The efficiency obtained being close to 100%, these results are excellent. Added to those of the strong scalability, they demonstrate that our parallel implementation is very efficient and versatile.

[*]0 means shared-memory calculation without domain decomposition.

Table 4: Strong scaling experiment on up to 128 domains using a total of 8192 CPU-cores.

		Simple partitioning			Geometric partitioning		
n_d	#core	time (s)	speedup	efficiency (%)	time (s)	speedup	efficiency (%)
0*	64	10597	-	-	10597	-	-
4	256	2557	4.1	104	2658	4	100
8	512	1242	8.5	107	1245	8.5	106
16	1024	667	15.9	99	628	16.9	106
32	2048	302	35.1	110	305	34.7	109
64	4096	146	72.6	113	147	72.1	113
128	8192	101	104.9	82	108	98.1	77

Table 5: Weak scaling experiment.

			Simple partitioning		Geometric partitioning	
n_d	#core	#dof	time (s)	efficiency (%)	time (s)	efficiency (%)
16	1024	228,388,608	145	-	142	-
32	2048	456,777,216	154	94	157	90
64	4096	913,554,432	146	99	147	96

6. CONCLUSIONS

In this paper, we have described a domain decomposition method suitable for solving the transport equation in distributed memory. We also present a mesh partitioning strategy based on Hilbert space-filling curves. The method is well adapted for DG based transport discretizations and is minimally intrusive in terms of implementation.

Numerical experiments show that the method is very robust, as we observe little variation in the solutions obtained when increasing the number of computational processes. Moreover, it enjoys an excellent weak and strong scalability. These advances are also valuable from the point of view of the end-users, as the use of this tool requires no additional effort compared to the sequential version.

Future work involves the evaluation of this approach on larger test cases, in particular the direct calculation of heterogeneous whole core. In addition, it would be valuable to integrate distributed and scalable preconditioners and acceleration strategies.

ACKNOWLEDGEMENTS

K. Assogba's PhD research work is supported by the CEA NUMERICS program, which has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No 800945.

The authors wishes to thank Grégoire Allaire, Anne-Marie Baudron, Ansar Calloo and Igor Zmijarevic for fruitful discussions.

REFERENCES

[1] K. Koch, R. Baker, and R. Alcouffe. "Solution of the First-Order Form of the Three-Dimensional Discrete Ordinates Equation on a Massively Parallel Machine." *Trans Am Nucl Soc*, **volume 65**(198) (1992).

[2] T. M. Evans, A. S. Stafford, R. N. Slaybaugh, and K. T. Clarno. "Denovo: A New Three-Dimensional Parallel Discrete Ordinates Code in SCALE." *Nuclear Technology*, **volume 171**(2), pp. 171–200 (2010).

[3] J. I. C. Vermaak, J. C. Ragusa, M. L. Adams, and J. E. Morel. “Massively Parallel Transport Sweeps on Meshes with Cyclic Dependencies.” *Journal of Computational Physics*, **volume 425**, p. 109892 (2021).

[4] E. Jamelot and P. Ciarlet. “Fast Non-Overlapping Schwarz Domain Decomposition Methods for Solving the Neutron Diffusion Equation.” *Journal of Computational Physics*, **volume 241**, pp. 445–463 (2013).

[5] N. Odry, J.-F. Vidal, G. Rimpault, J.-J. Lautard, and A.-M. Baudron. “Performance Study of a Parallel Domain Decomposition Method.” In *PHYSOR 2016 - International Conference on the Physics of Reactors: Unifying Theory and Experiments in the 21st Century* (2016).

[6] N. Odry, J.-J. Lautard, J.-F. Vidal, and G. Rimpault. “Coarse Mesh Rebalance Acceleration Applied to an Iterative Domain Decomposition Method on Unstructured Mesh.” *Nuclear Science and Engineering*, **volume 187**(3), pp. 240–253 (2017).

[7] C. R. E. de Oliveira, C. C. Pain, and A. J. H. Goddard. “Parallel Domain Decomposition Methods for Large-Scale Finite Element Transport Modelling.” In *Int. Conf. Math. Comp. Reactor Physics and Environmental Analysis of Nuclear System*. American Nuclear Society, Portland, Oregon (1995).

[8] S. Van Criekingen, F. Nataf, and P. Havé. “Parafish: A Parallel FE–PN Neutron Transport Solver Based on Domain Decomposition.” *Annals of Nuclear Energy*, **volume 38**(1), pp. 145–150 (2011).

[9] D. Hilbert. “Über die stetige Abbildung einer Linie auf ein Flächenstück.” In D. Hilbert, editor, *Dritter Band: Analysis - Grundlagen der Mathematik - Physik Verschiedenes: Nebst Einer Lebensgeschichte*, pp. 1–2. Springer, Berlin, Heidelberg (1935).

[10] L. Bourhrara. “A New Numerical Method for Solving the Boltzmann Transport Equation Using the PN Method and the Discontinuous Finite Elements on Unstructured and Curved Meshes.” *Journal of Computational Physics*, **volume 397** (2019).

[11] K. Assogba, L. Bourhrara, I. Zmijarevic, and G. Allaire. “Precise 3D Reactor Core Calculation Using Spherical Harmonics and Discontinuous Galerkin Finite Element Methods.” In *Proceedings of International Conference on Physics of Reactors 2022 (PHYSOR 2022)*, pp. 1224–1233. American Nuclear Society, Pittsburgh, PA, United States (2022).

[12] K. Assogba, L. Bourhrara, I. Zmijarevic, G. Allaire, and A. Galia. “Spherical Harmonics and Discontinuous Galerkin Finite Element Methods for the Three-Dimensional Neutron Transport Equation: Application to Core and Lattice Calculation.” *Nuclear Science and Engineering* (2023).

[13] P. Mosca, L. Bourhrara, A. Calloo, A. Gammicchia, F. Goubioud, L. Mao, F. Madiot, F. Malouch, E. Masiello, F. Moreau, S. Santandrea, D. Sciannandrone, I. Zmijarevic, E. Y. Garcias-Cervantes, G. Valocchi, J. Vidal, F. Damian, P. Laurent, A. Willien, A. Brighenti, L. Graziano, and B. Vezzoni. “APOLLO3®: Overview of the New Code Capabilities for Reactor Physics Analysis.” In *Proceedings of This Conference* (2023).

[14] L. Bourhrara. “New Variational Formulations for the Neutron Transport Equation.” *Transport Theory and Statistical Physics*, **volume 33**(2), pp. 93–124 (2004).

[15] G. Karypis and V. Kumar. “A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs.” *SIAM Journal of Scientific Computing*, **volume 20**(1), pp. 359–392 (1998).

[16] C. Chevalier and F. Pellegrini. “PT-Scotch: A Tool for Efficient Parallel Graph Ordering.” *Parallel Computing*, **volume 34**(6), pp. 318–331 (2008).

[17] E. E. Lewis, G. Palmiotti, T. A. Taiwo, R. N. Blomquist, M. A. Smith, and N. Tsoulfanidis. “Benchmark Specifications for Deterministic MOX Fuel Assembly Transport Calculations without Spatial Homogenization (3-D Extension C5G7 MOX).” *OECD’s Nuclear Energy Agency*, **volume 6** (2003).

[18] H. A. van der Vorst. “Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems.” *SIAM Journal on Scientific and Statistical Computing*, **volume 13**(2), pp. 631–644 (1992).