A note on the strong parallel scalability of numerically scalable Poisson linear solvers

Emmanuel Agullo, Luc Giraud, Valentin Joncquieres, Gilles Marait, Louis Poirel, Olivier Vermorel, Wilca Villafana

To cite this version:

Emmanuel Agullo, Luc Giraud, Valentin Joncquieres, Gilles Marait, Louis Poirel, et al.. A note on the strong parallel scalability of numerically scalable Poisson linear solvers. [Research Report] RR-9423, Inria Bordeaux - Sud Ouest. 2021, pp.31. hal-03352049v2

HAL Id: hal-03352049
https://hal.inria.fr/hal-03352049v2

Submitted on 24 Sep 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A note on the strong parallel scalability of numerically scalable Poisson linear solvers

E. Agullo, L. Giraud, V. Joncquières , G. Marait, L. Poirel ,
O. Vermorel, W. Villafana

ISSN 0249-6399 ISRN INRIA/RR--9423--FR+ENG

RESEARCH REPORT
N° 9423
September 2021
Project-Teams HiePACS

A note on the strong parallel scalability of numerically scalable Poisson linear solvers

E. Agullo*, L. Giraud*, V. Joncquières† ‡, G. Marait*,
L. Poirel* §, O. Vermorel†, W. Villafana†

Project-Teams HiePACS

Research Report n° 9423 — September 2021 — 31 pages

Abstract: In the context of a parallel plasma physics simulation code, we perform a qualitative performance study between two natural candidates for the parallel solution of 3D Poisson problems that are multigrid and domain decomposition. We selected one representative of each of these numerical techniques implemented in state of the art parallel packages and show that depending on the regime used in terms of number of unknowns per computing cores the best alternative in terms of time to solution varies. Those results show the interest of having both types of numerical solvers integrated in a simulation code that can be used in very different configurations in terms of selected modelisations, problem sizes and parallel computing platforms.

Key-words: Strong parallel scalability, numerical scalability, Poisson solvers, PIC model, fluid model, plasma simulation

---

[‡] Current address: Capgemini, Toulouse, France
[§] Current address: Michelin, Clermont-Ferrand, France
[*] Inria, Bordeaux, France
[†] Cerfacs, Toulouse, France

RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

Une note sur le passage à l'échelle parallèle des solveurs linéaires de Poisson numériquement scalables

Résumé : Dans le contexte d'un code de simulation parallèle de la physique des plasmas, nous effectuons une étude qualitative des performances entre deux candidats naturels pour la résolution parallèle de problèmes de Poisson en 3D que sontles méthodes multigrilles et les méthodes de décomposition de domaine. Nous avons sélectionné un représentant de chacune de ces techniques numériques implémentées dans des progiciels parallèles de pointe. Nous montrons que selon le régime utilisé en termes de nombre d'inconnues par cœurs de calcul la meilleure alternative en termes de temps de résolution varie. Ces résultats montrent l'intérêt de disposer de ces deux types de solveurs numériques intégrés pour de la simulation intensive qui peut être utilisée dans des configurations très différentes en termes de modélisations choisies, taille des problèmes et plates-formes de calcul haute performance.

Mots-clés : Scalabilité parallèle forte, scalabilité numérique, solveur de Poisson, modème PIC, modèle fluide, simulation de plasma

Contents

1 Introduction 4

2 The need of a scalable Poisson solver for space propulsion numerical simulation 4

3 Numerical linear solvers 7

4 Numerical experiments 9
- 4.1 Experimental framework 9
- 4.2 Parallel performance and discussion 12
  - 4.2.1 The fluid solver 12
  - 4.2.2 The PIC solver 13

5 Concluding remarks 15

A Useful parameters for reproducibility 18

B Benefits of the coarse grid correction 19

C Fluid 21

D Various performance results 23
- D.1 Elapsed times 23
- D.2 Memory consumption 25

1 Introduction

Numerical simulation has become a major tool in the understanding of complex phenomena as it allows for detailed studies at fine time and spatial scales once a suited model has been designed. In many engineering applications in industry or academia, such simulations involve at some stage the solution of sparse linear systems which size increases when the discretization step decreases or the bounded domain of calculation increases. When the problem is well-posed, the matrix associated with the discretization is non-singular and the solution of the linear system is not a theoretical issue. However, it often turns out to be a challenge in practice when the model has to be implemented in a parallel numerical software where the time to solution becomes a key criterion of performance. In the literature, the parallel scalability of the linear algebra solvers is often studied in details by the designers of the numerical schemes in a stand-alone setting. However, in practice, the solvers are only one component of a more complex software environment for large simulation codes; consequently they are used in various regimes that depend on the context of the simulations.

In this work, we investigate the parallel performance of parallel iterative schemes for the solution of a 3D Poisson problem in a plasma physics simulation. In such a simulation, most of the parallel computation kernels do scale very well and the linear solver, that is not the most computing intensive part, possibly becomes the bottleneck of the parallel performance. In that context, we perform parallel experiments with two natural candidates for solving 3D Poisson problems efficiently in parallel, namely a multigrid solver and a 2-level domain decomposition solver. On a realistic test example, we show that depending on the regime the solvers are used, in terms of number of degrees of freedom per computing cores, the time to solution ranking of the two solvers varies while the domain decomposition solver is always the most memory consuming.

The paper is organized as follows. In Section 2 the plasma simulation code used in this study is presented. The set of equations used is briefly described in order to highlight the challenges raised by their numerical resolution. Section 3 reviews the different numerical techniques available for the resolution of linear systems arising from the discretization of 3D elliptic Partial Differential Equations (PDE). The multigrid solver and the 2-level domain decomposition solver used for this parallel performance study are introduced in Section 4. The main results obtained with the two solvers on a typical plasma thruster configuration are discussed. Finally, some concluding remarks are provided in Section 5.

2 The need of a scalable Poisson solver for space propulsion numerical simulation

Electric propulsion is an increasingly popular technology to power a satellite for station-keeping or orbit raising missions. For electric propulsion companies, the design and qualification phases of electric thrusters are long (several years) and expensive and some physical phenomena occurring there are still not understood (loss of efficiency, appearance of spurious instabilities, anomalous erosion of the discharge chamber wall, ...). The AVIP code has been developed in order to help the understanding of plasma physics phenomena inside space plasma thrusters, in particular Hall effect thrusters [10]. This solver uses unstructured meshes in a high performance computing framework based on the AVBP code [6], state-of-the-art CFD code for reactive flows. AVIP solves the equations of charged particles (ions and electrons) and neutrals inside a plasma thruster chamber using two formalisms:

- an Eulerian fluid formalism describing the evolution of particle macroscopic quantities with a fluid equation system [10, 11], which can be written as :

$$\frac{\partial \vec{W}_\alpha}{\partial t} = \vec{F}_{convection}(\vec{W}_\alpha) + \vec{F}_{Lorentz,\alpha}(\vec{W}_\alpha) + \vec{S}_{collisions}(\vec{W}_\alpha) \tag{1}$$

with $\vec{W}_\alpha$ the vector of conservative variables for each species α (n for neutral, i for ion and e for electron) inside the plasma :

$$\vec{W}_\alpha = \begin{pmatrix} \rho_\alpha = m_\alpha n_\alpha \\ \rho \vec{u}_\alpha \\ \rho E_\alpha \end{pmatrix} \tag{2}$$

m_α, ρ_α, n_α, $\vec{u_\alpha}$ and E_α represent respectively the mass, mass density, number density, velocity vector and total energy of species α.

In Eq. 1, convections forces are denoted by $\vec{F}_{convection}$ and collision processes between different populations of particles are represented through the term $\vec{S}_{collisions}$. The Lorentz force $\vec{F}_{Lorentz,\alpha}$ represents the dynamics of charged particles (ions or electrons) in the electromagnetic field and is expressed by:

$$\vec{F}_{Lorentz,\alpha} = q(\vec{E} + \vec{u_\alpha} \times \vec{B}) \tag{3}$$

with q the charge of the considered species, $\vec{E}$ and $\vec{B}$ the electric and magnetic fields. In practice q is 0 for neutrals, $-e$ the elementary charge for electrons and a positive multiple of the elementary charge e for ions (singly ionized or more).

- a Lagrangian PIC (Particle-In-Cell) formalism in which the velocity space is discretized by considering a large number of particles belonging to species α. Each individual particle is tracked down by applying and integrating the Second Newton's Law:

$$\frac{d\vec{u_{p,\alpha}}}{dt} = \vec{F}_{Lorentz,p,\alpha} \tag{4}$$

$$\frac{d\vec{x_{p,\alpha}}}{dt} = \vec{u_{p,\alpha}} \tag{5}$$

  with $\vec{x_{p,\alpha}}$ and $\vec{u_{p,\alpha}}$ the position and velocity of each individual particle. Here the Lorentz force $\vec{F}_{Lorentz,p,\alpha}$ has the same expression has in Eq. 3 but the macroscopic speed $\vec{u_\alpha}$ is replaced by the individual particle speed $\vec{u_{p,\alpha}}$.

The system of fluid equations Eq. 1 is recognized as more complex to formulate than the system of PIC equations Eqs. 4 and 5 because it usually requires the introduction of many models to close the problem. For example, assumptions about the particle velocity distribution must be used to derive the Euler-type fluid system, which can potentially bias the simulation results. In this sense, the PIC approach can be considered more precise than the fluid approach and is generally the preferred approach for reproducing physics with a higher level of detail. However the number of particles required in a PIC approach to describe the overall plasma physics can become very large, which makes a PIC formulation more computationally challenging than a fluid formulation. Indeed the average number density of charged particles inside a thruster chamber is around $n_{e,i} = 10^{18}\ m^{-3}$ particles. Considering a thruster chamber of volume $V_c \approx 60\ cm^{-3}$, the number of particles to track is therefore around $10^{13} - 10^{14}$ which greatly exceeds the current calculation capabilities. Physical particles are then grouped in numerical macro-particles to follow their trajectories all at once. Typical 3D PIC simulations use $\sim 10^9$ numerical macro-particles. Despite this numerical trick, the computational time to treat all these particles is very large and high performances on large clusters are required to run realistic simulations of Hall effect thrusters. In practice, the fluid and PIC formalisms complement each other since PIC simulations can be used to assess the fluid model which aims to compute greater time ranges.

The Lorentz force $\vec{F}_{Lorentz}$ appearing in both formalisms (Eqs. 4 and 1) is a function of the electric field $\vec{E}$ and magnetic field $\vec{B}$. These electromagnetic fields $\vec{E}$ and $\vec{B}$ are related to each other via the well-known Maxwell equations. However in several space thrusters, the magnetic field $\vec{B}$ can be considered constant and the set of Maxwell equations reduces to a Poisson equation for the electric field $\vec{E}$. The Poisson equation describes the behavior of the electric potential ϕ generated by the difference of electric charges inside the domain (i.e., the difference between ion and electron densities) :

$$\Delta\phi = -\frac{e}{\epsilon_0}(n_i - n_e) \tag{6}$$

with ϵ_0 the dielectric permittivity.

Once the Poisson equation has been solved, the electric field can then be calculated with the relation:

$$\vec{E} = -\vec{\nabla}\phi.$$

The electric field $\vec{E}$ and the density of charged particles n_e and n_i are strongly coupled in this set of equations. Any error on the electric field can lead to a misprediction of the electron velocity that will directly affect the difference of charges. A high precision on the Lorentz force prediction and the electric field is therefore required to correctly reproduce the behavior of the plasma. Concerning the Poisson equation, great precision on the gradient of the unknown is required.

In general the meshes used in a 3D configuration of a thruster are relatively homogeneous. Boundary layers, commonly known as electric sheaths, exist nears walls and are characterized by high electric fields and large gradients of particle densities. Their discretization requires a local refinement of the mesh [11] but this difference in mesh size induces only a moderate heterogeneity in the entries of the discretization matrix.

In the present case, both AVIP-fluid and AVIP-PIC use a node-centered formalism based on a finite volume method. Therefore, the Poisson equation in Eq. 6 is discretized at the nodes as shown in Fig. 1. The details of the computation over the nodal volume is left to the reader [10].

Eventually the discretization of Eq. 6 leads to a linear system with a symmetric positive matrix A which solution ϕ_h is the potential ϕ at the discretization nodes:

$$\phi_h = -\frac{e}{\epsilon_0}(n_i - n_e) = f_h. \tag{7}$$

A short time to solution is always a crucial feature for a numerical solver, but it is even more true for AVIP which aims to deal with real-size realistic configurations. In practice, the computational time of an AVIP simulation (regardless of the formalism used, fluid or PIC) can be divided into three main physical operators: particle convection, collisions between particles (Monte-Carlo algorithm for AVIP-PIC, implicit solver for AVIP-Fluid) and Poisson equation.

The accuracy and the cost of a fluid approach is directly governed by the size of the mesh. The Lagrangian approach is based on a different data structure and its cost responds to a different logic. Accuracy is mainly driven by the number of numerical particles tracked and not by the

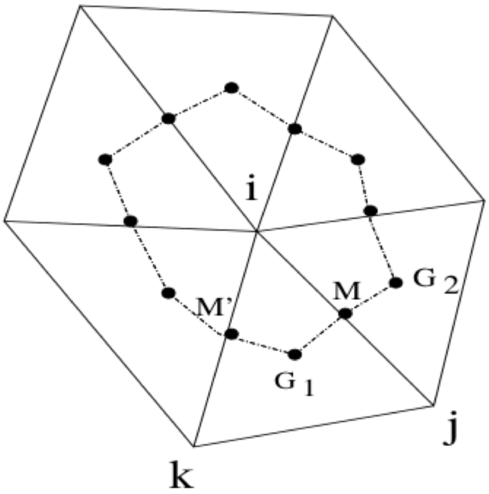


Figure 1: Discretization of the Poisson equation in a 2D triangular mesh with a node-centered strategy.

eulerian mesh resolution. In other words, it means that increased precision is usually achieved by increasing the number of particles while keeping the mesh unchanged, in order to improve the statistical representation of the dispersed phase. For all Lagrangian operators, increasing the number of cores is an obvious solution to reduce the time to solution considering their very good scalability. The challenge then is to solve the Poisson equation on an even smaller number of degrees of freedom (dof) per core. In practice, the objective is to solve the Poisson equation for numbers of dof per core of the order of a few hundred, numbers for which standard linear solvers are not necessarily designed and efficient. Thus, a Poisson solver able to work under different regimes in terms of number of unknowns per computing cores is absolutely essential for anyone who wishes to use both fluid and PIC approaches in a same code.

3 Numerical linear solvers

The solution of sparse linear systems of equations has been studied for a long time and still remains an active field of research. Many numerical packages have been developed during the last decades to efficiently cope with the features of the ever evolving computer architectures. Among the oldest and general purpose approaches are the ones referred to as direct methods. They are based on Gaussian elimination and rely on either multifrontal or supernodal schemes to address sparsity and parallelism. State of the art parallel packages that implement these techniques are: MUMPS [2], PARDISO [19], PaStIX [9] or SuperLU [13] to name a few. The direct methods are numerically robust, but their robustness comes at the price of large memory consumption and floating point operation (flop) count for the solution of linear systems arising from 3D PDE. For instance, the solution of a PDE defined on a 3D Cartesian grid of size $n = N^3$ requires a memory consumption of $\mathcal{O}(n^{2/3})$ and a flop count of $\mathcal{O}(n^2)$. In order to reduce these costs, recent developments attempt to exploit data sparsity as introduced in the $\mathcal{H}$-matrix community [7]. Alternatively, iterative methods have a low memory footprint but need to be combined with acceleration techniques to control the number of iterations. Modern iterative approaches are based on Krylov subspace and the acceleration techniques are referred to as preconditioning. For the solution of linear systems arising from the discretization of elliptic PDE, numerically scalable preconditionners have been designed, that can be split into two main families.

The first ones are multigrid techniques initially introduced on regular grids and later extended to unstructured matrices with the naming Algebraic Multi-Grid (AMG). In a two-level setting, the general idea is to smooth the high frequencies of the error via simple relaxation techniques and solve for the low frequency modes of the error on a coarser grid where those modes can be captured in a lower dimensional space. Applying recursively the two-grid idea to solve for the low frequencies leads to the multigrid method. In that context a sequence of linear systems of decreasing size are considered, on each of them a few relaxation steps, called pre-smoothing, are performed before restricting the residual on the next coarser space, where the error equation is handled. Once the coarsest level is reached, the error equation is solved and the solution is prolongated to the preceding finer level to correct the current iterate on this level; few more relaxation steps, called post-smoothing, might be applied before prolonging again the resulting approximated error to the next finer level. This procedure is applied until the finest level is reached and a new iterate for the solution of the linear system is produced. For elliptic problems, the multigrid algorithms have unique computational and numerical features such as a convergence that is independent of the mesh size and an asymptotic computational cost (memory, flop) that is linear with respect to the problem size. Their parallel implementation is based on a matrix partitioning, all the steps (smoothing, restriction, prolongation) can be implemented with a few neighbor-to-neighbor communications as long as the coarse problems are fine enough to keep all the computing cores busy. Intrinsically, when going down the multigrid hierarchy the computational granularity reduces. This is a very brief and incomplete overview of multigrid principles, we refer the reader to the rich bibliography and textbooks such as [4, 8, 23].

Another class of techniques for the solution of PDEs is known as domain decomposition methods. The main motivation is to naturally express some parallelism while overwhelming the unaffordable calculation costs of sparse direct methods if they were applied to the whole problem. In these approaches, the robustness of the direct methods is still exploited locally for the solution of the sub-problems resulting from the decomposition and an iterative process is designed to "glue" the local solutions so that each of them eventually converges towards the global solution restricted to the individual partition. The decomposition of the domain can have overlapping sub-domains, leading to a class of techniques originally called Schwarz methods. This refers to the alternating method introduced by H. A. Schwarz as a constructive procedure to prove the existence of the solution to a Poisson problem in a complex domain [20]; he built the solution as the limit of a series of solutions defined on a set of overlapping domains where each has a canonical geometry and whose union forms the complex domain. This pioneering idea has inspired and still inspires research activities and his name is associated with a large class of techniques. Alternatively, sub-domains without overlap can be considered leading to techniques often referred to as sub-structuring approaches where local solutions are composed to form a reduced linear system, only defined on the interface between the sub-domains. Within this approach, the reduced system, also called the Schur complement system, is solved iteratively. The natural parallelism introduced by the splitting of the domain has a defect due to the associated induced decoupling, the number of iterations increases with the number of sub-domains. A numerical remedy to design numerically scalable preconditioners, i.e., with a convergence that does not depend on the number of sub-domains, consists in introducing a coarse space to re-couple contributions from all the sub-domains represented in a low dimensional space. In the parallel design of the solver a special attention should be paid in the implementation of the coarse space correction that might become a bottleneck for the parallel scalability. This illustrates the fragile trade-off to be found between numerical and parallel scalability of the linear solvers. For a more detailed presentation of the domain decomposition techniques and associated theory we refer to the following list of books and reference therein [5, 14, 18].

4 Numerical experiments

4.1 Experimental framework

The parallel performance study has been conducted with one representative of each of the two scalable numerical linear algebra solvers. Namely, for the representative of the multigrid we chose the GAMG algebraic multigrid solver available in the PETSc library [3] named AMG in the paper; it implements a smoothed aggregation algebraic multigrid method. As representative of domain decomposition solver, we selected MaPHyS [1, 17] that implements an additive Schwarz preconditioner for the Schur complement with a coarse space correction scheme that follows GENEO [22, 21] ideas. This specific version of Maphys [17] will be referred to as DDM in the sequel. For both solvers, we tuned the control parameters following the advice of the package developers through a trial and correct iterative loop. The selected parameters that depart from the default are listed in Appendix A, while we refer the reader to the user's guide of the two packages for more details.

The same stopping criterion is used in both linear solvers, which relies on a standard scaled residual norm:

$$||f_h - A\phi_h||_{L_2} \leq \epsilon \|f_h\|_{L_2}. \tag{8}$$

Solver performances are evaluated on a standard AVIP configuration of a 3D Hall effect thruster cavity. The geometry represents a 5 degree azimuthal sector of the ionization chamber and the plume, with dimensions of a standard SPT-100 Hall thruster (15mm high and 34 mm long) [15]. It corresponds to an azimuthal extrusion of the 2D case simulated in [12]. The geometry and the boundary conditions applied for ions, electrons and the electric potential are shown in Fig. 2. Only periodic and Dirichlet boundary conditions are used for the Poisson equation. Both fluid and PIC simulations are performed on the same mesh made of 27 million tetrahedral elements (around 5 million dof). The cell size is relatively homogeneous with a minimum of $50\mu m$ near the ceramic walls (blue hatches in Fig. 2) and around $80\mu m$ elsewhere.

Figure 3 shows some typical plasma quantities inside the computational domain for the fluid simulation at steady state. In this configuration, the propellant (neutral xenon atoms) is injected at the anode (see Fig. 2) while electrons are injected at the cathode. As the neutral xenon atoms diffuse into the channel of the thruster (Fig. 3a), they are ionized by collisions with circulating electrons (Fig. 3b). The difference of electric charges creates an electric field in the axial (z) direction that accelerates the ions in the thruster exit plane in order to provide the thrust (Fig. 3d).

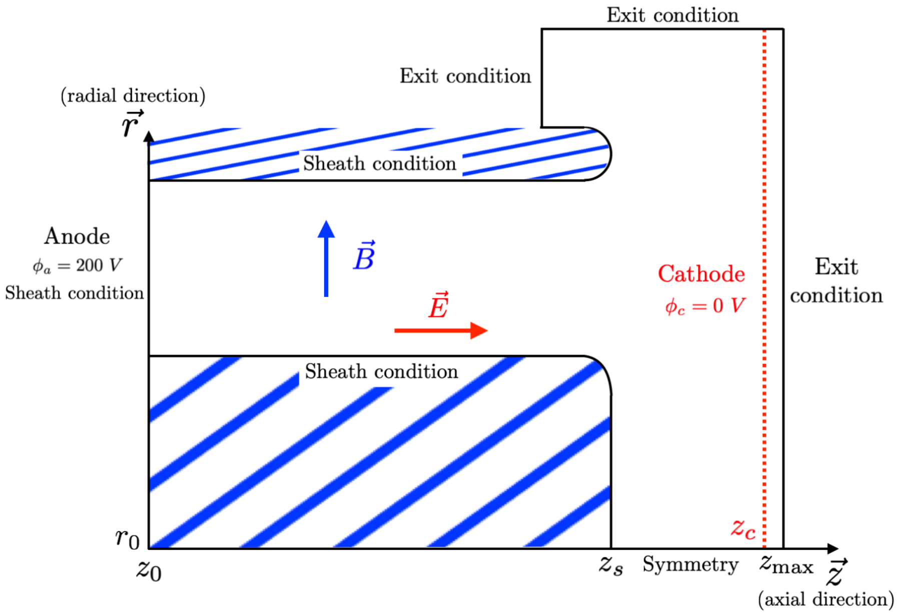
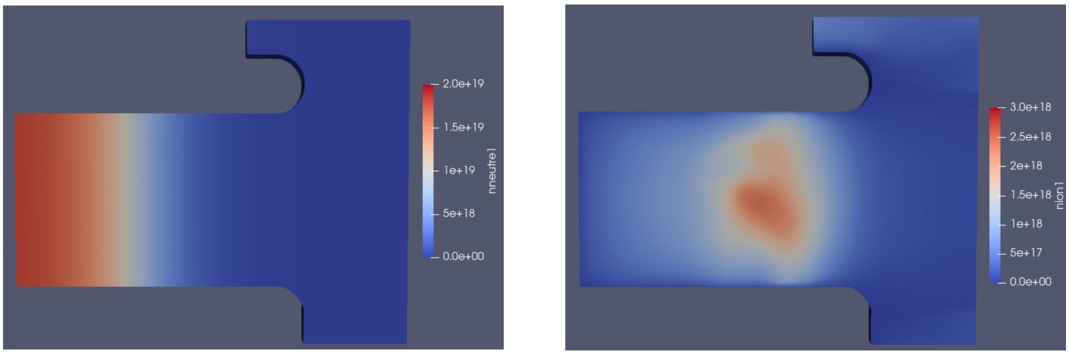
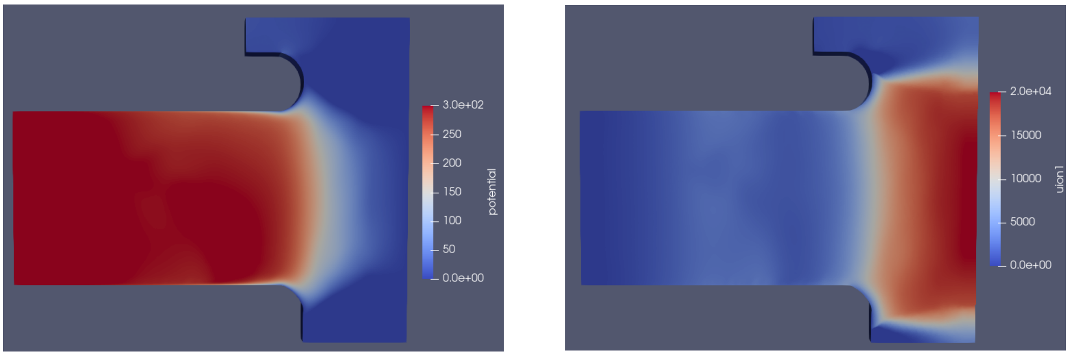

Exit condition
Exit condition
(radial direction)
$\vec{r}$
Sheath condition
Anode
$\phi_a = 200\ V$
Sheath condition
$\vec{B}$
$\vec{E}$
Cathode
$\phi_c = 0\ V$
Exit
condition
Sheath condition
r_0
z_c
z_0
z_s
Symmetry
$z_{\max}$
$\vec{z}$
(axial direction)

Figure 2: Geometry and boundary conditions for the simulation of a SPT-100 (the blue hatches represent ceramic walls) [12]. The figure does not represent the azimuthal direction $\vec{\theta}$ (5 degree sector) for which periodic conditions are imposed.



(a) Neutral density $[m^{-3}]$ diffusing towards the plume.

(b) Ion density $[m{-}3]$ generated by neutral-electron collisions.



(c) Electric potential $[V]$: drop at the thruster exit plane.

(d) Axial ion velocity $[m.s^{-1}]$: acceleration at the thruster exit plane.

Figure 3: Working principle of a Hall thruster at steady state from a $r - z$ plane view.

4.2 Parallel performance and discussion

All simulations have been performed on the Occigen supercomputer hosted by CINES in Montpellier (France) [16]. In order to get reliable and avoid possible outliers, each run was repeated three times and subsequent performances were averaged out. All the parallel elapsed times reported in this section have been measured on the first 50 time steps of the AVIP solver using either the fluid or the PIC model. A typical simulation requires around two million time steps, but 50 are enough to get representative values of the parallel performance of all the numerical kernels. For all the simulations the number of unknowns involved in the Poisson problem is about 5 millions.

4.2.1 The fluid solver

The computational complexities of the computational kernels involved in the AVIP-Fluid solver are fully governed by the mesh size. In Fig. 4, we display the time spent in each of these numerical kernels. Regarding the performance of the two parallel linear solvers, it can be observed that when the number of unknowns per core is larger than 5 000 (i.e., using less than 1000 cores on our examples), the multigrid solver is faster than the domain decomposition one; after this crossover point the tendency is reversed and multigrid can be up-to 2.5 slower than the domain decomposition solver while it was more than 4 times faster on 360 cores. The time spent in the linear system solution with the multigrid solver reaches a minimum value for a computing core number around 800, while the minimum for the domain decomposition approach is slightly lower and observed for a number of cores around 1500. Eventually, the time for the two solvers increases with the same rate, when the solution time starts to be dominated by communications.

Regarding the other numerical kernels, the cost of the collision operator is always marginal, while the convection operator consumes most of the time when the number of computing cores is small. These two operators scale in excellent proportions to about 2000 dof per core (i.e., 2500 computing cores) but still gain speed beyond this limit.

In terms of total time to solution, one can observe that it reduces with the two solvers up-to 2500 computing cores. Beyond, the time spent in the linear solvers starts to dominate so that the solution time even increases with AMG beyond 2500 computing cores and 3500 cores for the DDM solver. Consequently, it does not make sens to use more than 2500 computing cores for AVIP-Fluid with AMG and more than 3500 with DDM for this specific case. Yet, because AVIP spends more than 80% of the CPU time in the convection operator in the high dof per core range, the initial advantage of the AMG solver is unfortunately not significative.

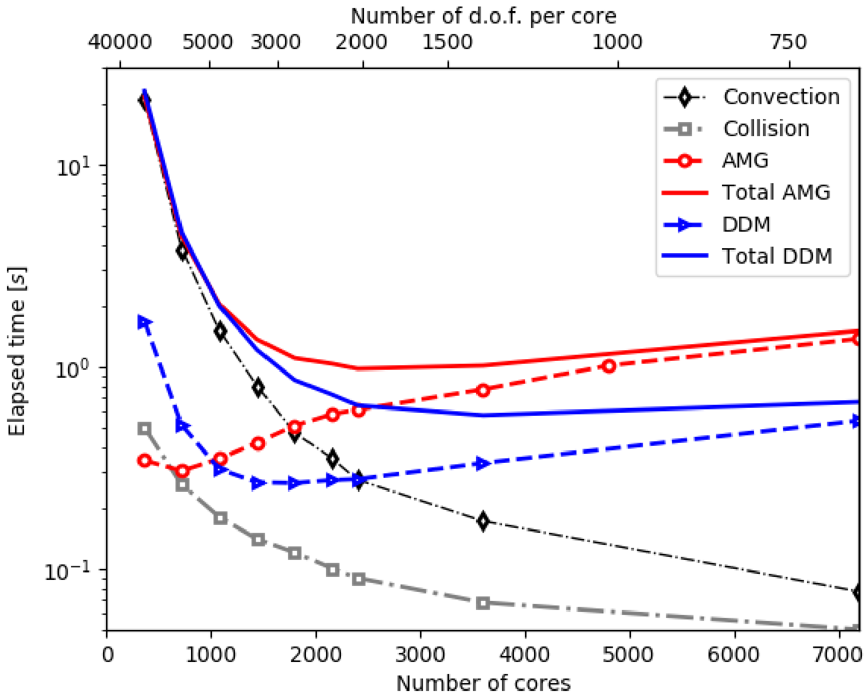


Figure 4: Elapsed time for experiments using fluid model.

4.2.2 The PIC solver

Three PIC simulations were performed by varying the number of macro-particles per cell at initialization: 30, 60 and 120. The total number of macro-particles tracked during the computation therefore varies between approximately 2.43 10^9 and 9.72 10^9. The computational complexity for the PIC simulations depends on the number of macro-particules involved in the simulation. In order to illustrate this effect on the parallel performance we consider two configurations with 30 and 120 macro-particules per cell at the initialisation. We report in Figs. 5 and 6, the parallel strong scalability performance when the number of cores varies from 360 up-to 7200. For these simulations the mesh remains unchanged and the Poisson problem still has around 5 Mdof. In that respect, the comments made above on the parallel performance of the linear solvers remain the same.

In Fig. 5, it can be seen that the kernels related to the convection, collison and interpolation calculations scale relatively well up-to 7200 cores. The total simulation time scales reasonable well up-to 2500 cores, beyond the times to solution do still reduce with the DDM solver and stagnates with AMG. This relatively poor parallel performance is due to the lack of scalability of the two parallel linear solvers when the number of unknowns per computing core decreases.

Similar observations can be made for results presented in Fig. 6 on the parallel performance where 120 macro-particules per cell are used to get a better statistical analysis on the simulated physics. On that example, due to memory constraint, the first measurement point corresponds to a run on 1000 cores, that is the cross-point between the AMG and DDM parallel performance. Because the relative weigth of the linear solver is lower than for the fluid model, the total times to solution decrease up-to 7200 cores when either of the two solvers is used.

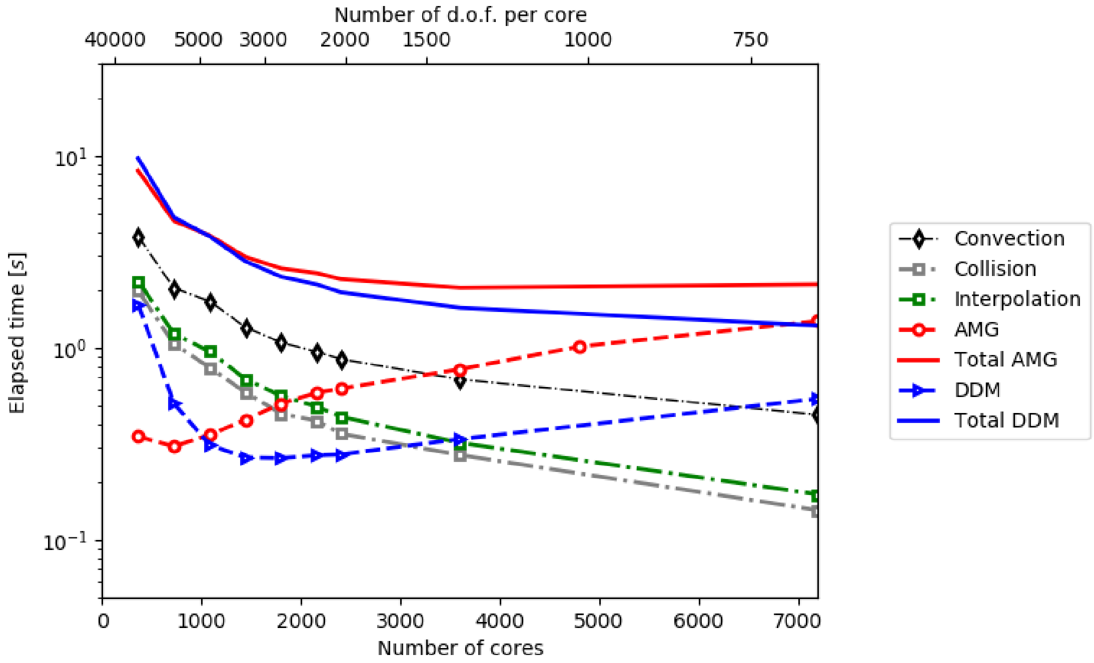


Figure 5: Elapsed time for experiments with 30 macro-particles per cell.

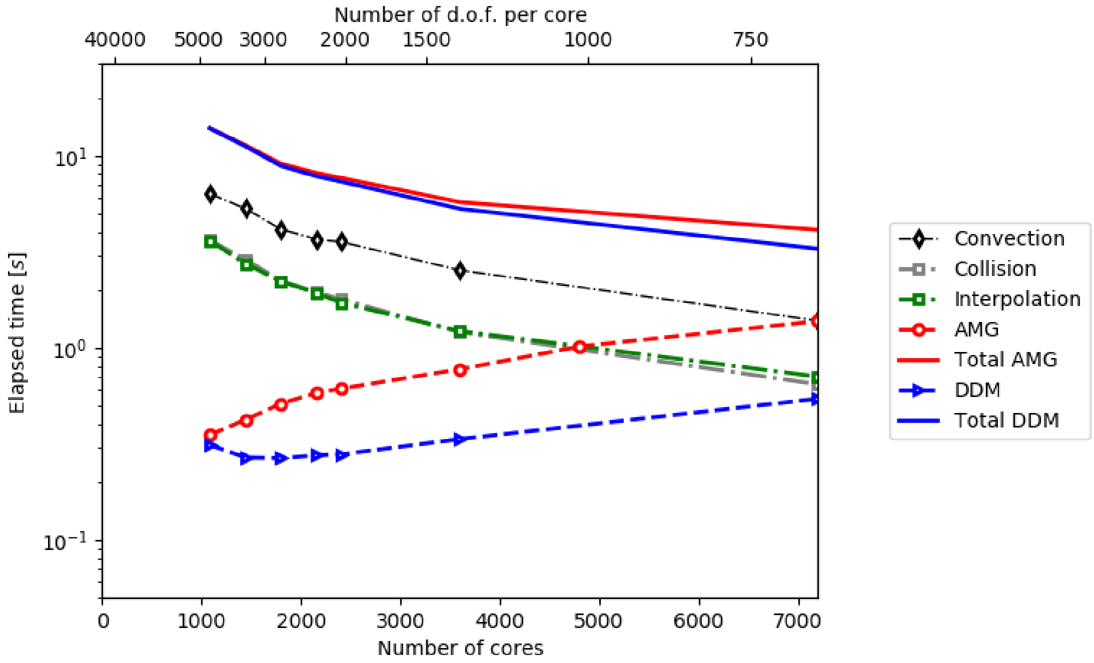


Figure 6: Elapsed time for experiments with 120 macro-particles per cell.

We report in Fig. 7 the memory consumption (mean and max value) for both linear solvers when the number of computing cores varies. The AMG solver is significantly less memory demanding with a very good scalability and a nice load balancing; futhermore the memory footprint becomes negligible when using more and more processors. The DDM approach is much more memory consuming, the mean memory scales also rather well but a significant unbalance appears. This is due to the fact that the coarse space correction that involved a linear system which size grows linearly with the number of sub-domains is solved on a fixed size MPI sub-communicator. Consequently the memory used on thus sub-communicator increases as the number of sub-domains/cores increases. Therefore, the DDM strategy can represent beyond 80 % of total allocated memory for the sub-communicator. While the parallel implementation could be improved it remains an intrinsic parallel bottleneck for a two-level numerical scheme that should be extended to multi-level techniques to possibly exhibit memory performance similar to AMG. For more detailed results on the memory consumption scalability of the two linear solvers we refer the reader to Figs. 19-23 in Appendix D.2.

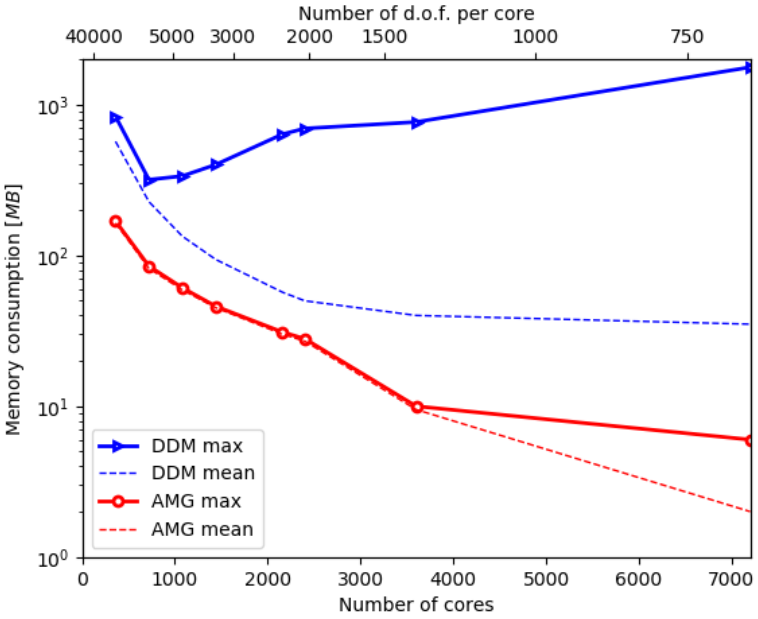


Figure 7: Memory consumption in function of the number of cores for two linear solvers. The maximal memory consumption is represented with dashed lines, the mean consumption with continuous lines.

5 Concluding remarks

As shown by the results of this paper, neither of the two numerically scalable linear solvers that are multigrid or domain decomposition are systematically superior to the other on parallel platforms even for the widely encountered Poisson problem. The parallel ranking between the two classes of numerically scalable linear solvers depends on the range of problem and platform sizes. In our example in plasma physics, multigrid is the most efficient with large number of unknowns per core for a moderate number of cores; it quickly suffers from a lost of computational granularity that leads to an increasing time to solution. Scalable domain decomposition methods are more expensive for large sub-domain size due to the weight of its direct solver component but significant gains can quickly be obtained thanks to the nonlinear computational cost of this direct solver ingredient.

While it might sound obvious, this study highlights that the performance agility of a complex simulation code relies on its capability to use a variety of parallel linear solvers and select then on their best performance ranges. The rule for AVIP is to use multigrid when the number of unknowns is larger than 5000 and switch to domain decomposition for lower values. One option for a possibly better scalability on a large number of computing cores would be to depart from the classical MPI SPMD programming model and would consist in buiding a sub-communicator for the solution of the Poisson problem which size will be adjusted to get the best performance of the chosen linear solver. Such an implementation in the AVIP code will be the subject of a possible future work.

Acknowledgements

This work is financially supported by Safran Aircraft Engines as a part of a CIFRE convention and supervized by Stephan Zurbach. This work is also financially supported by the ANR chaire industrielle POSEIDON (ANR-16-CHIN-0003-01) (LPP/CERFACS/SAFRAN). It was performed using HPC resources from GENCI (Grant A0032B10157). We are extremely thankful to the PETSc team for the great and reactive support all along this study, which allowed us to significantly improve the usage of the solver.

References

[1] Emmanuel Agullo, Luc Giraud, and Louis Poirel. Robust preconditioners via generalized eigenproblems for hybrid sparse linear solvers. *SIAM Journal on Matrix Analysis and Applications*, 20(2):23, 2019.

[2] Patrick R. Amestoy, Iain S. Duff, Jean-Yves L'Excellent, and Jacko Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[3] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc Web page. 2018.

[4] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A Multigrid Tutorial (2Nd Ed.).* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[5] Victorita Dolean, Pierre Jolivet, and Frédéric Nataf. *An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation*, volume 144. SIAM, 2015.

[6] N. Gourdain, L. Gicquel, M. Montagnac, O. Vermorel, M. Gazaix, G. Staffelbach, M. Garcia, J-F Boussuge, and T. Poinsot. High performance parallel computing of flows in complex geometries: I. Methods. *Computational Science and Discovery*, 2(015003):1–26, 2009.

[7] Lars Grasedyck and Wolfgang Hackbusch. Construction and Arithmetics of H-Matrices. *Computing*, 70(4):295–334, August 2003.

[8] Wolfgang Hackbusch. *Multi-Grid Methods and Applications.* Springer, 1985.

[9] Pascal Hénon, Pierre Ramet, and Jean Roman. PASTIX: A high-performance parallel direct solver for sparse symmetric positive definite systems. *Parallel Computing*, 28(2):301–321, 2002.

[10] V. Joncquières. *Modélisation et simulation numérique des moteurs à effet Hall.* PhD thesis, Institut National Polytechnique de Toulouse, 2019.

[11] V. Joncquières, F. Pechereau, A. Alvarez Laguna, A. Bourdon, O. Vermorel, and B. Cuenot. A 10-moment fluid numerical solver of plasma with sheaths in a Hall effect thruster. In *AIAA Joint Propulsion Conference, Cincinnati, USA*, 4905, 2018.

[12] V. Joncquières, O. Vermorel, and B. Cuenot. A fluid formalism for low-temperature plasma flows inside hall effect thrusters in an HPC unstructured solver. *Plasma Sources Sci. Technol.*, 2019.

[13] Xiaoye S. Li and James W. Demmel. SuperLU_DIST: A Scalable Distributed-memory Sparse Direct Solver for Unsymmetric Linear Systems. *ACM Trans. Math. Softw.*, 29(2):110–140, June 2003.

[14] Tarek P. A. Mathew. *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations*, volume 61. Springer Science & Business Media, 2008.

[15] A. Morozov and V.V. Savelyev. Fundamentals of stationary plasma thruster theory. In *Reviews of plasma physics*, pages 203–391. Springer, 2000.

[16] Occigen. https://www.cines.fr/calcul/materiels/occigen/.

[17] Louis Poirel. *Algebraic domain decomposition methods for hybrid (iterative/direct) solvers.* Theses, Université de Bordeaux, November 2018.

[18] Alfio Quarteroni and Alberto Valli. *Domain Decomposition Methods for Partial Differential Equations.* Oxford University Press, 1999.

[19] Olaf Schenk, Klaus Gärtner, Wolfgang Fichtner, and Andreas Stricker. PARDISO: A High-Performance Serial and Parallel Sparse Linear Solver in Semiconductor Device Simulation. 2000.

[20] Hermann Amandus Schwarz. *Über Einen Grenzübergang Durch Alternirendes Verfahren.* Zürcher u. Furrer, 1870.

[21] Nicole Spillane. *Méthodes de décomposition de domaine robustes pour les problèmes symétriques définis positifs.* PhD Thesis, Paris 6, 2014.

[22] Nicole Spillane, Victorita Dolean, Patrice Hauret, Frédéric Nataf, Clemens Pechstein, and Robert Scheichl. Achieving robustness through coarse space enrichment in the two level Schwarz framework. In *Domain Decomposition Methods in Science and Engineering XXI*, pages 447–455. Springer, 2014.

[23] Ulrich Trottenberg, Cornelius W. Oosterlee, and Anton Schuller. *Multigrid.* Elsevier, 2001.

A Useful parameters for reproducibility

PETSc 3.11 GAMG	
Options	Definition
-gamg_est_ksp_type cg	iterative solver for the computation
-gamg_est_ksp_max_it 10	max number of iterations
-pc_gamg_process_eq_limit 500	max unknowns on each process
-pc_gamg_coarse_eq_limit 1000	max unknowns on coarsest grid
-mg_levels_esteig_ksp_type cg	iterative solver for the coarse grid
-mg_levels_esteig_ksp_max_it 10	max unknowns for the coarse grid
-mg_levels_pc_type sor	sor smoothing for the coarse grid
-mg_levels_ksp_type richardson	solver used for the smoothing
-ksp_norm_type unpreconditioned	stopping criterion based on the unpreconditioned residual norm

MaPHyS 0.9.8	
Options	Definition
ICNTL(21) = 10	
ICNTL(51) = 4	
ICNTL(52) = 192	
ICNTL(33) = 5	

Table 1: Parameters that depart from the default for each package.

B Benefits of the coarse grid correction

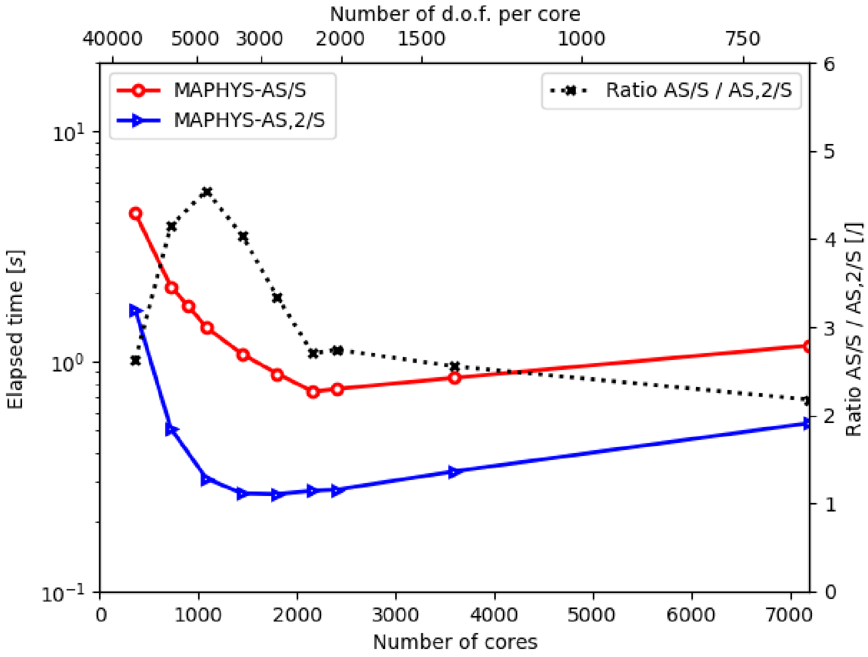


Figure 8: Elapsed time in function of the number of cores for MaPHyS with and without the coarse grid correction.

C Fluid

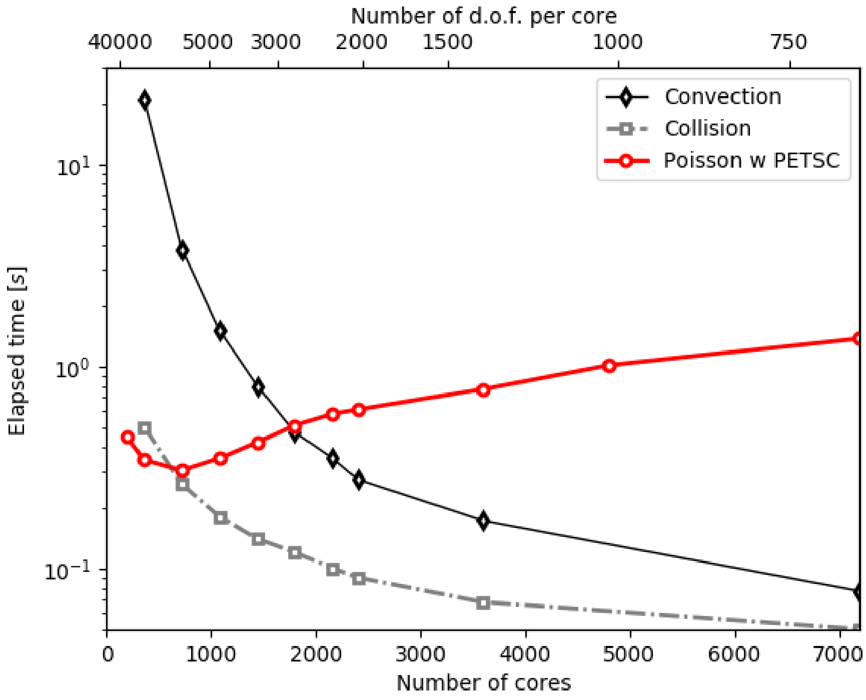


Figure 9: Elapsed time spent in each operator depending on the number of cores for a AVIP-Fluid simulation.

D Various performance results

D.1 Elapsed times

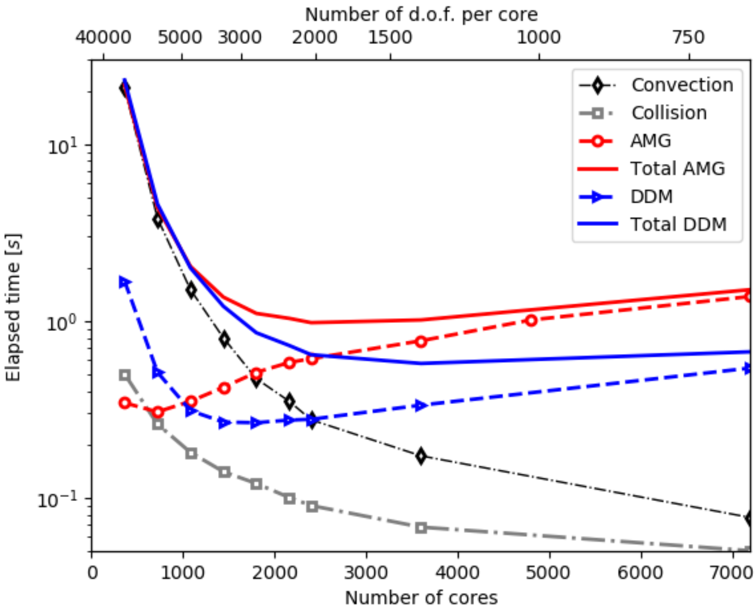


Figure 10: Elapsed time for experiments using fluid model.

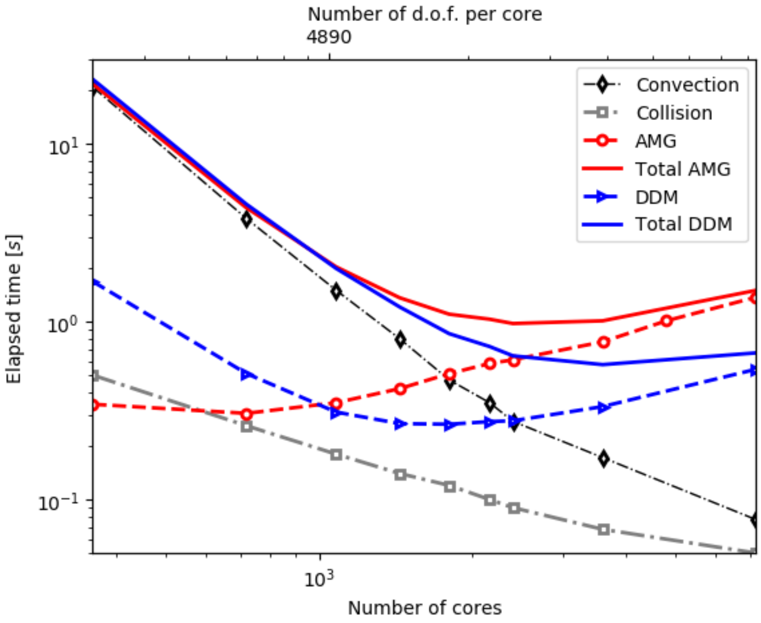


Figure 11: Elapsed time for experiments using fluid model with log scale on core axis.

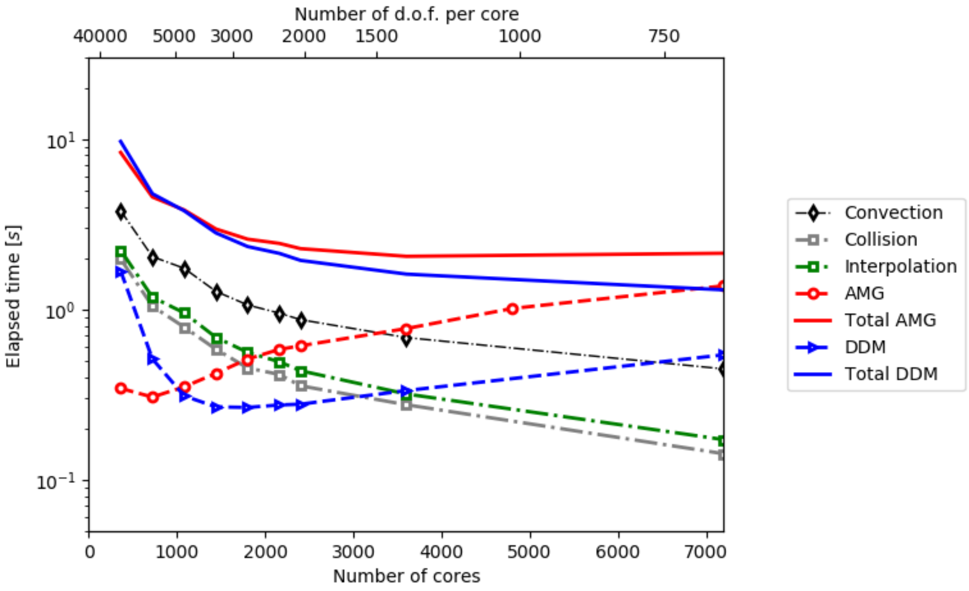


Figure 12: Elapsed time for experiments with 30 particles per cell.

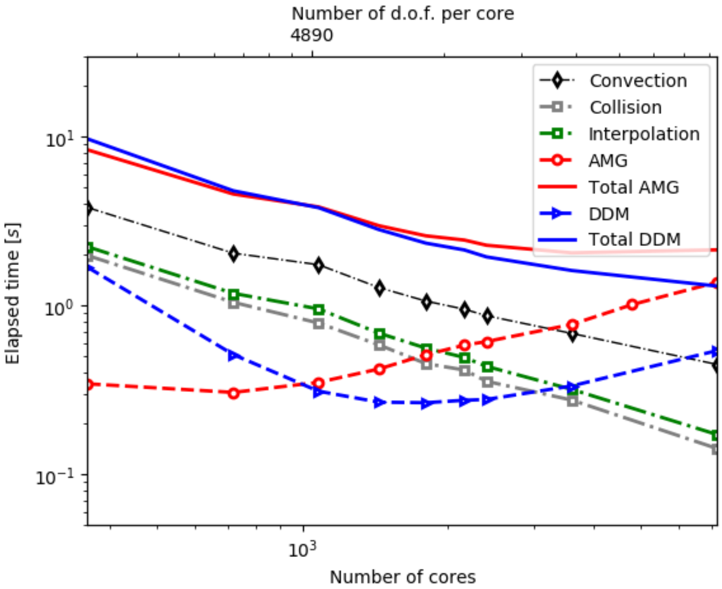


Figure 13: Elapsed time for experiments with 30 particles per cell with log scale on core axis.

D.2 Memory consumption

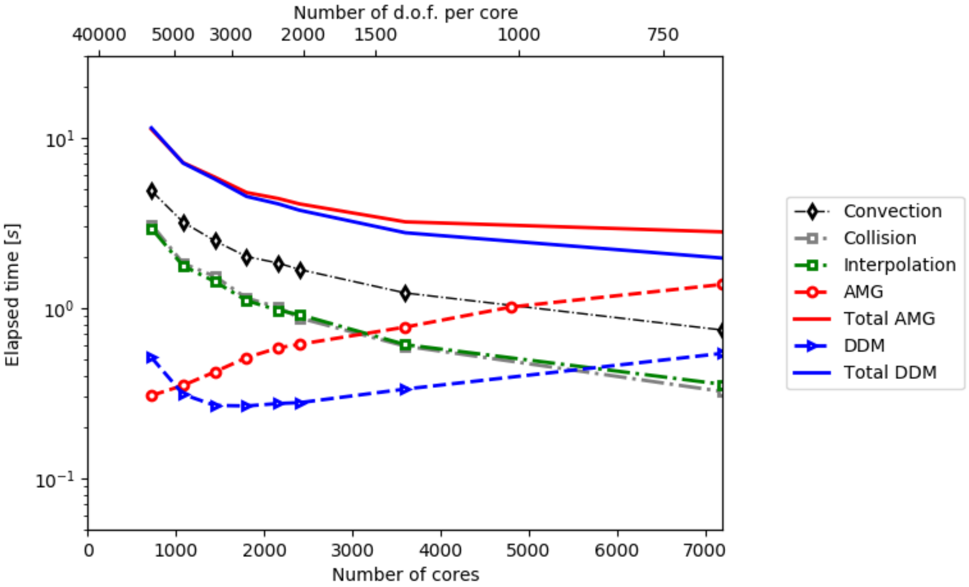


Figure 14: Elapsed time for experiments with 60 particles per cell.

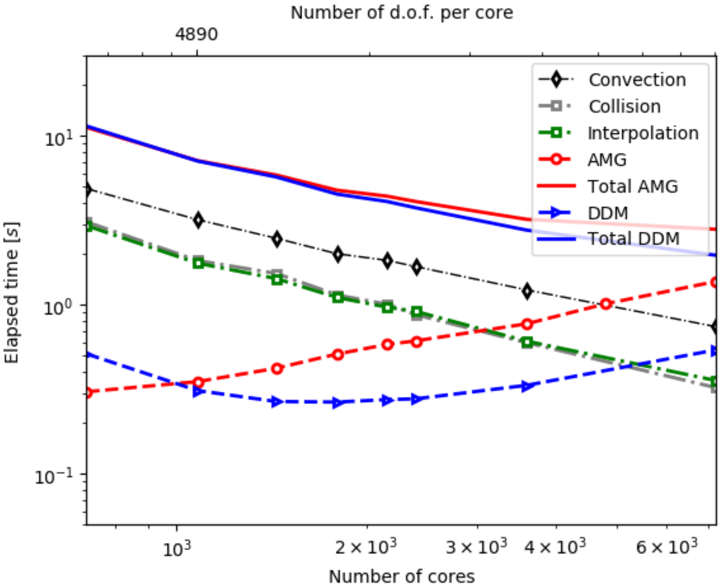


Figure 15: Elapsed time for experiments with 60 particles per cell with log scale on core axis.

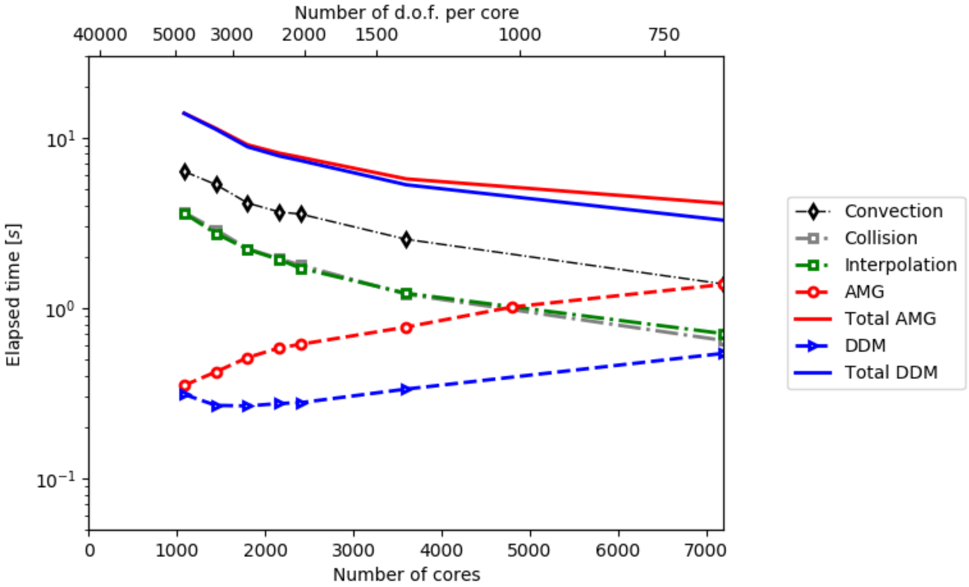


Figure 16: Elapsed time for experiments with 120 particles per cell.

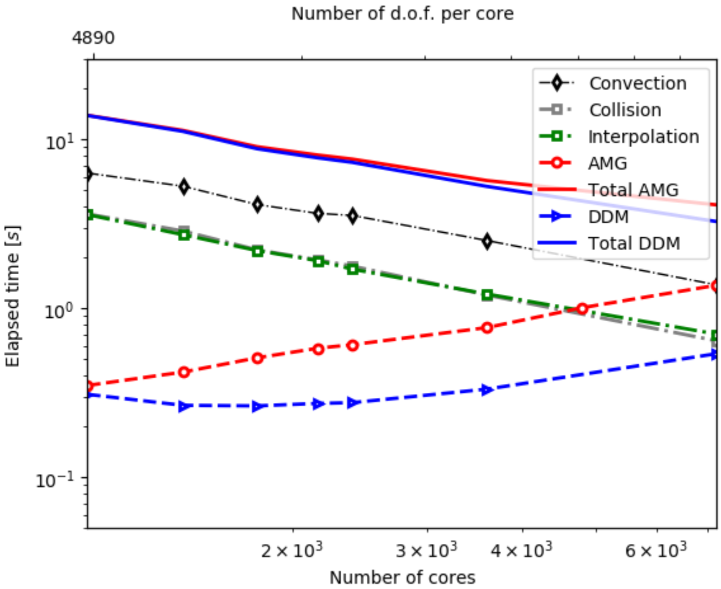


Figure 17: Elapsed time for experiments with 120 particles per cell with log scale on core axis.

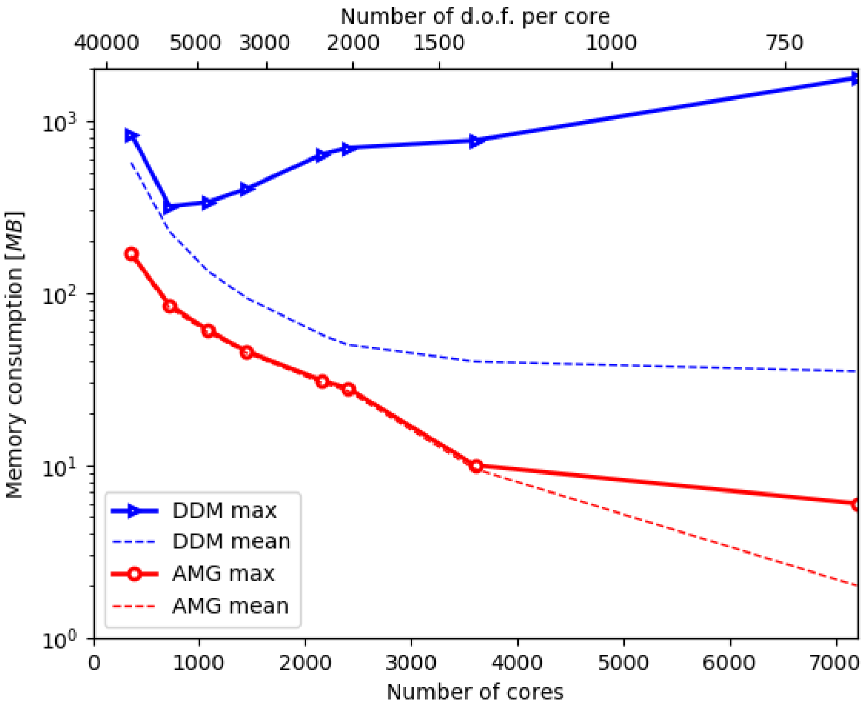


Figure 18: Comparison of the memory consumption of the AMG and DDM solvers.

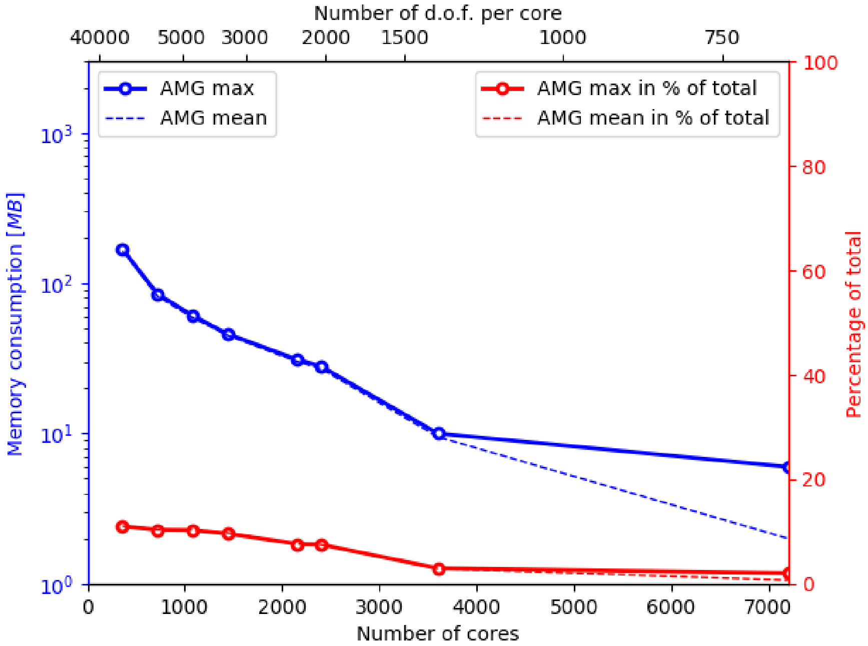


Figure 19: Memory consumption with AMG and ratio of the total memory for PIC simulation with 30 particles per cell.

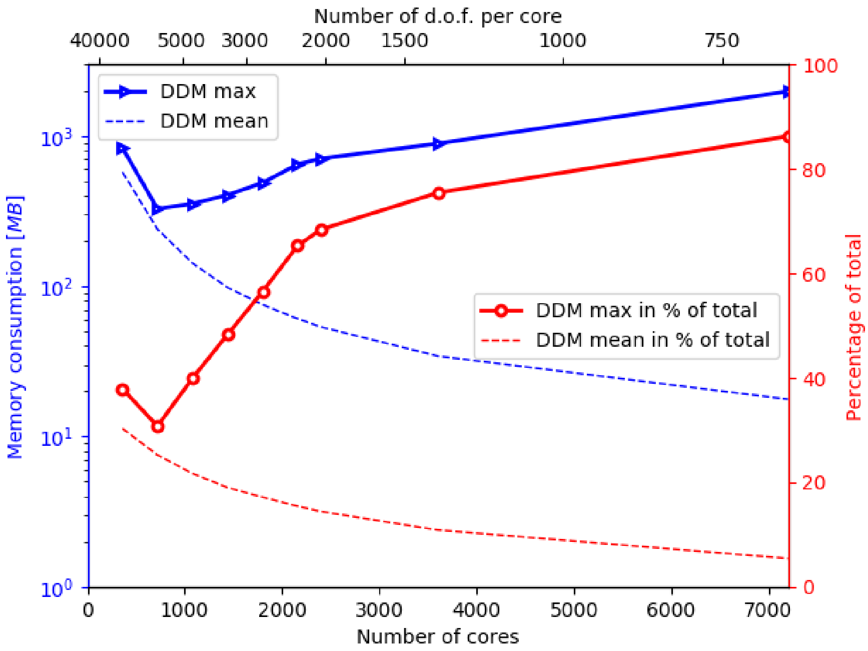


Figure 20: Memory consumption with DDM and ratio of the total memory for PIC simulation with 30 particles per cell.

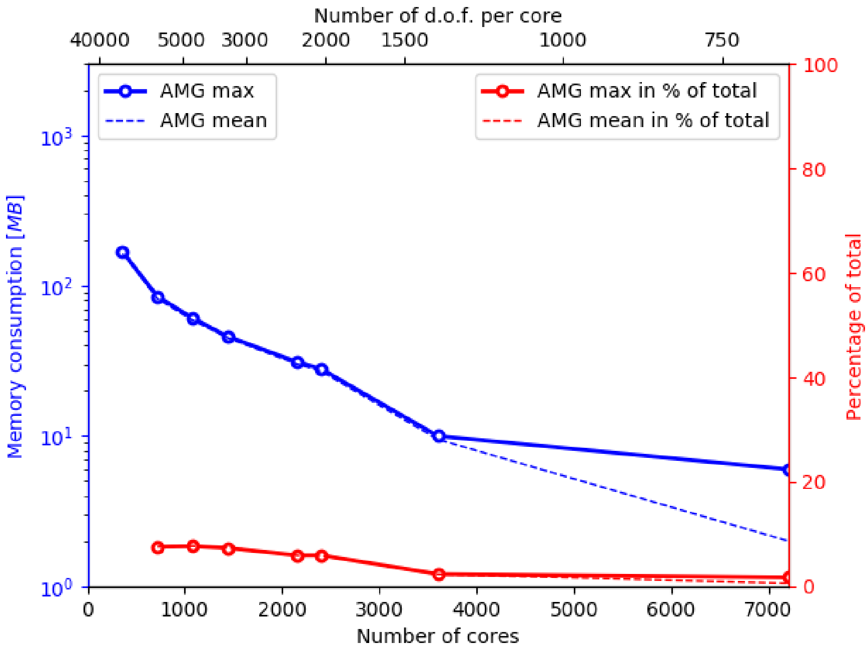


Figure 21: Memory consumption with AMG and ratio of the total memory for PIC simulation with 60 particles per cell.

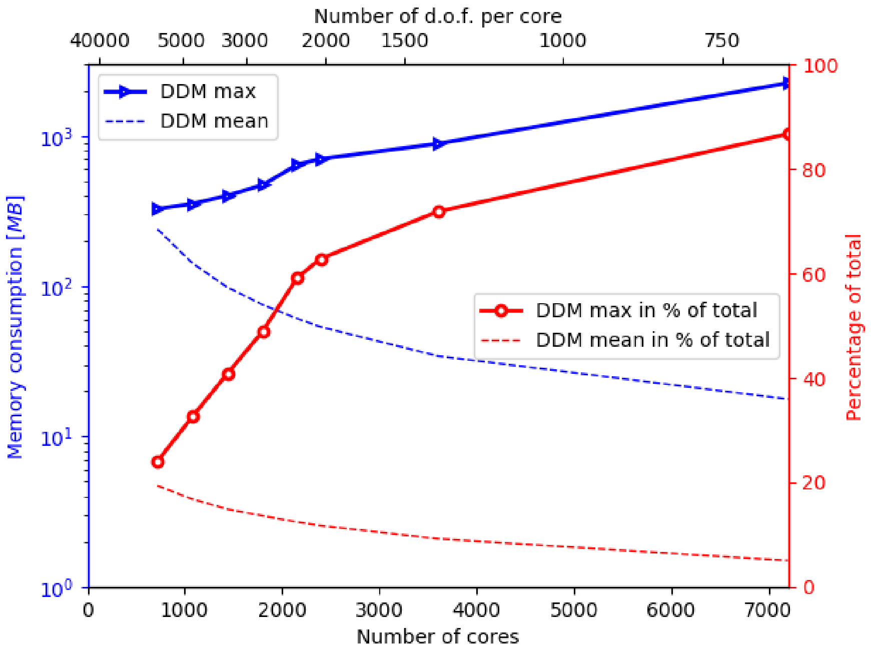


Figure 22: Memory consumption with DDM and ratio of the total memory for PIC simulation with 60 particles per cell.

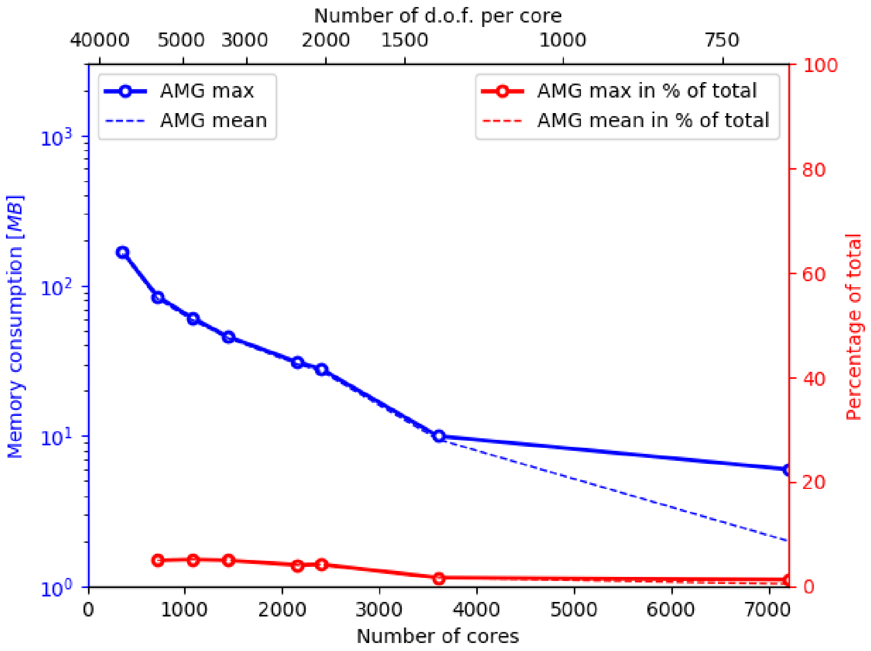


Figure 23: Memory consumption with AMG and ratio of the total memory for PIC simulation with 120 particles per cell.

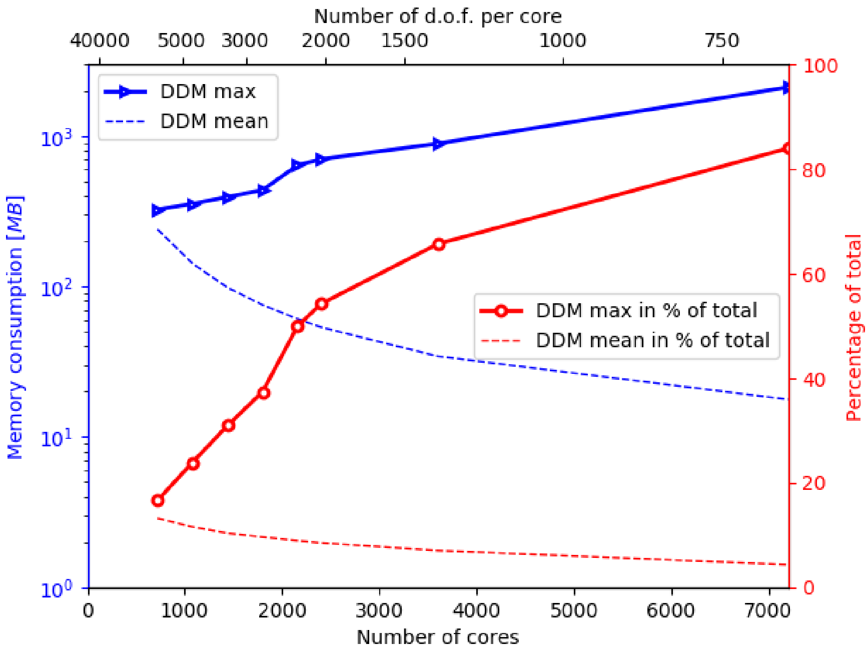


Figure 24: Memory consumption with DDM and ratio of the total memory for PIC simulation with 120 particles per cell.

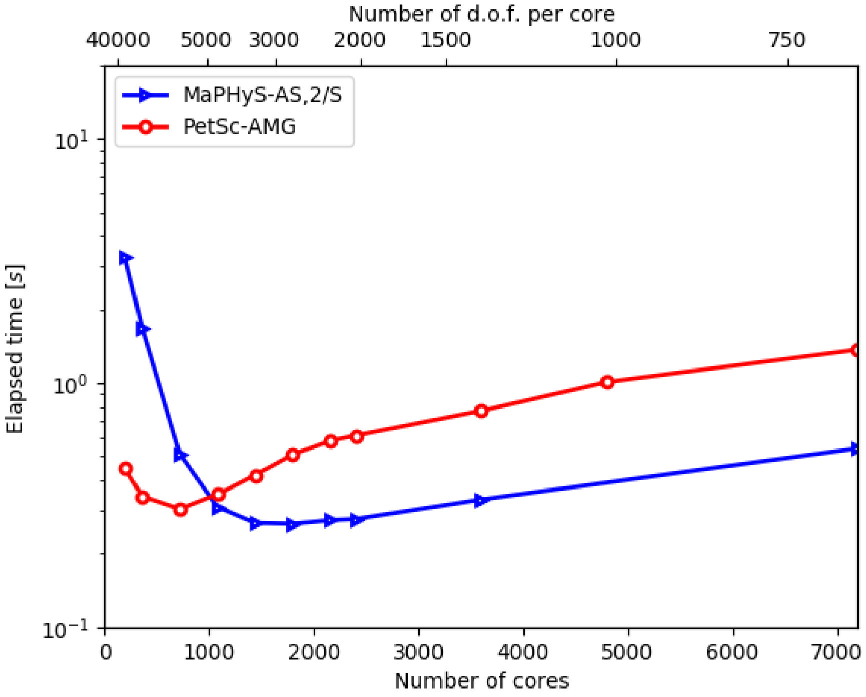


Figure 25: Elapsed time in Poisson solution using AMG or DDM.

RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399