Solution of univalued and multivalued pseudo-linear problems using parallel asynchronous multisplitting methods combined with Krylov methods

Thierry Garcia, Pierre Spitéri, Lilia Ziane Khodja, Raphael Couturier

Version of Record: https://www.sciencedirect.com/science/article/pii/S0965997820309753
Manuscript_84be1b7f3efd3b08cd7e2267f24794ad

Solution of univalued and multivalued pseudo-linear problems using parallel asynchronous multisplitting methods combined with Krylov methods

T. Garcia[a,*], P. Spiteri[b], L. Ziane-Khodja[c], R. Couturier[d]

[a] *University of Toulouse, IRIT-INPT , 2 rue Charles Camichel, B.P. 7122, 31071 Toulouse - Cedex 7, France*
[b] *University of Toulouse, IRIT-INPT-ENSEEIHT, 2 rue Charles Camichel, B.P. 7122, 31071 Toulouse - Cedex 7, France*
[c] *ANEO, 122 Avenue du Général Leclerc, 92100 Boulogne-Billancourt, France*
[d] *University of Bourgogne Franche Comté, CNRS, FEMTO-ST Institute, 19 Av. du Marchal Juin, BP 527, 90016 Belfort cedex, France*

Abstract

The paper improves a preliminary experimental study on a cluster by adding both theoretical results and experimental tests on a grid platform. These algorithms solve univalued and multivalued pseudo-linear problems using parallel asynchronous multisplitting methods combined with Krylov's methods. This paper also analyses these algorithms using contraction techniques. Two distinct applications, with discretized boundary value problems, are analyzed and simulated. First, a univalued convection-diffusion problem perturbed by an increasing diagonal operator is presented. Then, follows the description of a diffusion problem whose solution is constrained. This situation classically leads to the solution of a multivalued pseudo-linear problem in which the linear part is perturbed by an increasing diagonal multivalued operator. Parallel asynchronous and synchronous algorithms were implemented and tested on a grid platform composed of physically adjacent or geographically distant machines. In addition, the simulation results are detailed and show that the elapsed times obtained for the asynchronous algorithms are significantly less than those obtained for the synchronous algorithms.

Keywords: multisplitting methods, synchronous parallel algorithms, asynchronous parallel algorithms, sparse non-linear systems, Krylov methods, pseudo - linear problem, boundary value problem.

1. Introduction and motivation.

The present study is related to the analysis and application of mixed multisplitting methods for solving pseudo - linear stationary problems. These problems are stationary either intrinsically or as a result of the discretization of time evolution problems by implicit or semi-implicit time marching schemes. The considered problems are defined as an affine application $AU - F$ perturbed by an increasing diagonal operator Φ as follows

$$AU - G + \Phi(U) = 0, \tag{1}$$

where in the sequel A has the property of being a large scale M-matrix, G a vector, U is the unknown vector. Note that this type of problem occurs when solving elliptic, parabolic or hyperbolic second order boundaries values problems and that the M-matrix property is well verified after discretization by classical finite differences schemes, finite volumes schemes or finite elements methods provided that, in this last case, the angle condition is verified. In fact two distinct cases are then considered; in the first case corresponding to problem (1) $U \rightarrow \Phi(U)$ is a strongly non-linear univalued operator; while in the second case Φ is a multivalued application, for example $\Phi \equiv \partial\Psi$ sub-differential of the indicator function Ψ of the convex $\mathcal{K}$ taking into account constraints on the unknown U to be determined; so in this latter case, the problem to solve is defined by

$$AU - G + \partial\Psi(U) \ni 0, \tag{2}$$

*Corresponding author
Email addresses: thierry.garcia@irit.fr (T. Garcia), pierre.spiteri@enseeiht.fr (P. Spiteri), lzianekhodja@aneo.fr (L. Ziane-Khodja), raphael.couturier@univ-fcomte.fr (R. Couturier)

where for example the convex set is defined by

$$\mathcal{K} = \{U \text{ such that } U \geq \varrho\}$$

where ϱ is a given application. Two different kinds of sets can be considered: a convex set with a one sided inequality constraint like the previous one or a set where the solution U must be between two extreme values. In both previous cases each problem will be solved by a specific method; for the univalued problem (1) a local linearization corresponding to the implementation of the iterative Newton method will be carried out; while for the multivalued problem (2), one operates in an iterative alternative way by solving the linear equation without taking into account the constraints we obtain an intermediate value and then projecting this intermediate value on the convex set $\mathcal{K}$ until convergence. In each case, the computation method consists in solving a large sparse linear system. Each system is then associated with a fixed point problem and we intend to solve it by asynchronous parallel iterations [1]-[5]. Taking into account the properties of the matrix A and the operator's monotony property of the perturbed diagonal operator, it is proven in the following that the fixed point application is contractive with respect to a uniform weighted norm [6], which ensures on the one hand the existence and uniqueness of the solution of the algebraic system to be solved and on the other hand the convergence of parallel asynchronous iterations towards the solution of the target problem.

Using the fixed point equation, the parallel iterative asynchronous algorithms are then classically defined in [1] for the solution of large linear algebraic systems and [2] for large algebraic systems. In the first work on asynchronous parallel iterations, delays modeling the asynchronism between the processors were bounded (see [1] and [2]); in fact in [3] G. Baudet has extended the framework of the study to cover cases where delays are no longer bounded, allowing for cases of failure of one or more processors to be taken into account.

In addition, in order to unify the presentation and analysis of algorithm behavior, we consider multisplitting methods that unify the presentation of subdomain methods, either to model subdomain methods without overlap, or to model subdomain methods with overlap such as Schwarz's alternating method. The multisplitting method was introduced by O'Learry and White and also White (see [7]-[8]) in order to give a unified presentation of subdomain methods. Several contributions have been developed by many authors such as J. Arnal, Z.Z. Bai, R. Bru, A. Frommer, V. Migalon, J. Penades. D. Szyld, ... etc ... in collaboration with several co-authors (see [9]-[20]) for the solution of linear and nonlinear problems. Nevertheless, note that these previous works do not concern problem solution of multivalued problem except for the work of J. Bahi et all [21] developed in an Hilbertian context. These multisplitting methods are then applied to determine the solution to the two previous target problems as well as to the univalued problem (1) and to the multivalued problem (2). The convergence analysis is carried out by contraction techniques with respect to a weighted uniform norm. To solve the model problems, efficient methods are used to find the solution of each of the subproblems handled by each processor. More precisely, a coupling between asynchronous parallel methods and Krylov methods [22] is considered, since each diagonal subproblem obtained by the decomposition of problem (1) or (2) is solved by this last type of algorithm.

As an application, a convection-diffusion problem, perturbed by an increasing diagonal operator [23], is considered, the problem then being solved by a mixed Newton-multisplitting method. On the other hand, there is a diffusion problem whose solution is subjected to constraints of an inequality type, this problem being solved by a relaxation method with projection after each update of the iterative process, each linearized or linear subsystems being solved by the generalized minimal residual method (GMRES). Thus the results of parallel simulation achieved on a grid are presented and discussed in this paper.

The present study completes a preliminary study both theoretically and experimentally. Indeed, theoretically, in relation to [24] this paper is not limited to the single-valued problem but the resolution of multi-valued problems of the form (2) is also considered. On the experimental level, in [24] results on a cluster were presented while in the presented results come from grid, composed of distant and heterogeneous clusters.

In this paper there are two original contributions. The first contribution is in the theoretical part, especially with regard to the resolution of pseudo-linear univalued or multivalued problems by multisplitting methods. From an algorithmic point of view for the resolution of univalued problems the Newton's method is for example used. As previously said multivalued problems are formulated when the solution to be determined is subject to inequality constraints where, from an algorithmic point of view, one must update several components (additive type algorithm) or component by component (multiplicative type algorithm), in both cases without taking into account the

constraint and thus obtain an intermediate value which is projected onto the convex set to take into account the inequality constraints to which the solution is subject. The common point between the univalued and multivalued aspects is that the operator which perturbs the affine application $AU - G$ is in both univalued and multivalued cases (See books [25] and [26]) diagonal monotone, which facilitates the analysis of the behavior of the parallel asynchronous algorithms used to solve the target problems. In particular, the forthcoming Proposition 2 recalls a convergence result for any sub-problem decomposition due to the fact that matrix A is an M-matrix (see [27]). This has a consequence for the use of multisplitting methods where, usually, it is assumed that fixed point applications associated with model problem to solve are contractant. However, in the working framework considered in this paper, given the result of Proposition 2, this assumption of contraction of fixed point applications associated with problem to be solved is not necessary since this assumption is of course verified for every decomposition of the problem. In particular, this assumption is applicable for multi-valued problems which are not generally studied in paper concerning multisplitting methods and which are completely solved in this paper (see in particular the result of the forthcoming Corollary 1). Thus, in brief, the contribution at the theoretical level concerns the convergence of multisplitting methods applied to pseudo-linear problems (affine applications perturbed by monotone diagonal operators) in particular for problems whose solution is constrained and classically formulated under a multivalued formulation, since it is well known that the subdifferential of the indicator function is monotone (See book V. Barbu [25]). The second contribution is at the experimental level where, for the solution of pseudo-linear problems, on the one hand we combine a coordination iteration constituted by an asynchronous parallel iteration and on the other hand we solve local sub-problems by a very efficient conjugated gradient method.

The present paper is organized as follows. Section 2 presents the formulation of synchronous and, more generally, asynchronous parallel algorithms and some results allowing to analyze the convergence by contraction techniques are given. In the next section the parallel asynchronous multisplitting algorithms applied to pseudo linear problems are detailed. Section 4 is devoted to present boundary value problems, which, after appropriate discretization leads to solve pseudo linear algebraic systems. Thus the following section is devoted to present the principle of implementation of the studied parallel numerical methods. Section 6 presents the results of parallel experiments achieved on a grid. Finally a conclusion, some future studies and an appendix conclude the paper. This appendix presents the values of the maximum error of the correction obtained in both synchronous and asynchronous mode for the Newton method and these results are commented, both on an experimental and a theoretical level.

2. Parallel asynchronous algorithms

2.1. Discretized model problems

In the sequel, the set of natural integers will denoted by $\mathbb{N}$. Let $M \in \mathbb{N}$. Let us consider the following pseudo - linear problem, issued from the discretization of various boundary value problems

$$AU + \Phi(U) = G, \tag{3}$$

where $A \in \mathcal{L}(\mathbb{R}^M)$ is the discretisation matrix, $U \in \mathbb{R}^M$, $G \in \mathbb{R}^M$, and Φ and G result from the discretisation of a diagonal maximal monotone operator Φ, which means that $i-$th component of Φ is equal to $\Phi_i(u_i)$ where u_i is the $i-$th component of U and g the right-hand side of the problem, modelling the source term. Let us assume

$$A \text{ is an M-matrix} \tag{4}$$

$$\Phi \text{ is a diagonal increasing operator.} \tag{5}$$

Note that problem (3) is singlevalued.

Nevertheless when the solution U is submitted to some inequality constraints such that

$$U_{min} \leq U \text{ or } U \leq U_{Max} \text{ or } U_{min} \leq U \leq U_{Max}, \tag{6}$$

then, classically, the problem can be expressed by a multivalued formulation (see for example [25], [26]) as follows

$$AU + \partial\Psi(U) - G \ni 0, \tag{7}$$

where $\partial\Psi(U)$ results from the discretization of $\partial\psi(u)$, ψ being the indicator mapping of the convex set defining the constraints and $\partial\psi(u)$ denotes the subdifferential mapping of ψ; classically $\partial\Psi(U)$ satisfies the following property

$$\partial\Psi(U) \text{ is a diagonal increasing or more generally monotone operator.} \tag{8}$$

Let us denote by $\mathcal{E} = \mathbb{R}^M$; note that $\mathcal{E}$ is an Hilbert space. Consider also that the space $\mathcal{E} = \prod_{i=1}^{\alpha} \mathcal{E}_i$ is a finite product of $\alpha > 0$ subspaces denoted $\mathcal{E}_i = \mathbb{R}^{m_i}$, where $\sum_{i=1}^{\alpha} m_i = M$; note that $\mathcal{E}_i$ is also an Hilbert space where $\langle . , . \rangle_i$ denotes the scalar product and $|.|_i$ the associated norm, for all $i \in \{1, \ldots, \alpha\}$. Then for all $U, V \in \mathcal{E}$ let us denote by $\langle U , V \rangle = \sum_{i=1}^{\alpha} \langle U_i , V_i \rangle_i$ the scalar product on $\mathcal{E}$ and $\| . \|$ its associated norm.

We consider now a block-decomposition of problem (3) or (7) in α blocks, which leads for the first one to solve

$$\sum_{j=1}^{\alpha} A_{ij} U_j + \Phi_i(U_i) = G_i, \text{ for all } i \in \{1, \ldots, \alpha\}, \tag{9}$$

and for the second one (7)

$$\sum_{j=1}^{\alpha} A_{i,j}.U_j + \partial\Psi_i(U^i) - G_i \ni 0, \text{ for all } i \in \{1, \ldots, \alpha\}, \tag{10}$$

where $U_i \in \mathcal{E}_i$, $A = (A_{ij})$, and $\partial\Psi_i$ (or Φ_i) result from the associated block decomposition; the subdifferential mapping being multivalued, the previous relation (10) can also be written as follows

$$\sum_{j=1}^{\alpha} A_{i,j}.U_j + w_i(U_i) - G_i = 0, w_i(U_i) \in \partial\Psi_i(U_i), \forall i \in \{1, \ldots, \alpha\}. \tag{11}$$

Then, the following fixed point mapping, at its fixed point $U^\star$ if it exists, is associated to the problem (9) or (11) (in the following, this property is verified to be true):

$$\begin{cases} \text{Find } U^* \in \mathcal{E} \text{ such that} \\ U^* = F(U^*) \end{cases} \tag{12}$$

where $V \mapsto F(V)$ applies from $\mathcal{E}$ to $\mathcal{E}$. The mapping F is implicitly related to problem (3) to solve as follows

$$A_{i,i}.U_i + \Phi_i(U_i) = G_i - \sum_{j \neq i}^{\alpha} A_{ij} V_j, \text{ for all } i \in \{1, \ldots, \alpha\},$$

while for the multi-valued problem (10) the associated fixed point problem is

$$A_{i,i}.U_i + w_i(U_i) = G_i - \sum_{j \neq i}^{\alpha} A_{ij} V_j, w_i(U_i) \in \partial\Psi_i(U_i), \text{ for all } i \in \{1, \ldots, \alpha\},$$

2.2. *Parallel fixed point methods*

According to the decomposition of $\mathcal{E}$, let us consider the corresponding block decomposition of F and V

$$\begin{array}{rcl} F(V) & = & (F_1(V), \ldots, F_\alpha(V)). \\ V & = & (V_1, \ldots, V_\alpha) \end{array}$$

The parallel asynchronous fixed point iterations $\{U^p\}_{p \in \mathbb{N}}$, which are formally defined by using the following concepts, are then considered.

Definition 1. *A steering of block-components of the iterate vector is a sequence* $\{s(p)\}$, $p \in \mathbb{N}$, *such that* $s(p) \subset \{1, \ldots, \alpha\}$, $s(p) \neq \emptyset$, *for all* $p \in N$, *where in addition the steering satisfies*

$$\{p \in N \mid i \in s(p)\} \text{ is infinite, for all } i \in \{1, \ldots, \alpha\}. \tag{13}$$

A sequence of delayed iteration numbers $\{\rho(p)\}$ *of vectors* $\rho(p) = (\rho_1(p), \ldots, \rho_i(p), \ldots, \rho_\alpha(p)) \in \mathbb{N}^\alpha$ *is such that for all* $p \in \mathbb{N}$ *and* $i \in \{1, \ldots, \alpha\}$ *we have* $0 \leq \rho_i(p) \leq p$ *and* $\rho_i(p) = p$ *if* $i = s(p)$. *Moreover the delayed iteration numbers satisfy*

$$\lim_{p \to \infty} \rho_i(p) = +\infty, \text{ for all } i \in \{1, \ldots, \alpha\}. \tag{14}$$

Definition 2. *The general class of asynchronous iterative methods is defined recursively as follows.* U^0 *being given, for all* $p \in \mathbb{N}$ *and* $i \in \{1, \ldots, \alpha\}$:

$$U_i^{p+1} = \begin{cases} F_i(\tilde{U}^p) \text{ if } i \in s(p), \\ U_i^p \text{ if } i \notin s(p), \end{cases} \tag{15}$$

where

$$\tilde{U}^p = \{U_1^{\rho_1(p)}, \ldots, U_i^{\rho_i(p)}, \ldots, U_\alpha^{\rho_\alpha(p)}\} \in \mathcal{E}, \text{ if } p \geq 1 \tag{16}$$

$s(p)$ *and* $\rho(p)$ *being defined according to Definition 1.*

Remark 1. *Asynchronous iterations defined recursively by (15)-(16) are general iterative methods whereby iterations are carried out in parallel by up to* α *processors without any order nor synchronization. The main feature of this class of iterative methods is to allow flexible communications between the processors and to minimize the weight of synchronisations between the parallel processes. In such computational method, to make the most of computing power by eliminating idle times due to blocking expectations, synchronizations are not necessarily required which avoids having to wait for the communications of the values computed by the other processors; thus each process performs its own calculations using available data calculated by other processors. Then each processor advances its own calculations at its own pace, with communications taking place in no pre-established order. The choice of the relaxed components is performed using a selection of several components of the strategy* $s(p)$ *at each step* p *of the calculation; this strategy is in fact a non-empty subset of the set* $s(p) \subset \{1, 2, \ldots, \alpha\}$ *which well models parallelism between the processes, since each element of the strategy is not limited to a single element. In addition, theoretically, each block component of the iterate vector is continuously updated; but in practice the parallel iterative method is ended by a stopping criterion, which in the asynchronous context, is very hard to perform. For a fixed process, the asynchronism between updates is modelled by the introduction of delayed components calculated by other processors to take into account the necessary coupling between the various processes. Thus, in a typical update of the i-th block-component of the iterate vector at iteration* $p + 1$, *all the values* $\tilde{U}_j^p$ *of the block-components of the iterate vector are taken equal to* $\tilde{U}_j^p$ *corresponding to the value computed in the last update of the j-th block-component; so* $\tilde{U}_j^p \equiv U_j^{\rho_j(p)}$ *models the nondeterministic behavior of the iterative scheme. Practically, one will choose update corresponding to the last available value of each component. It should also be noted that in the asynchronous parallel iteration model the information exchanged between processors is no longer time-limited and can have unlimited communication delays, which makes it possible to take into account possible temporary failures of multiprocessors, clusters or grids. Note that parallel synchronous iterations correspond to the particular case where* $\rho_j(p) = p$ *for all* $p \in \mathbb{N}$; *moreover, in this case, if for all* $p \in \mathbb{N}$, $s(p) = \{1, \ldots, \alpha\}$ *(respectively* $s(p) = (1 + p \mod \alpha)$*) then algorithm (15)-(16) model the sequential block-Jacobi (respectively Gauss-Seidel) method.*

For simplicity's sake the study of the convergence is limited in this paper to the use of contraction techniques. The convergence of parallel asynchronous iterations for the solution of the multivalued problem (7) is considered; the case of singlevalued problem (3) being treated similarly in a straightforward way. First, a contraction result associated with the point decomposition is established; then it can be concluded that the fixed point F is contracting for any decomposition into α large blocks.

Proposition 1. *Consider the problem (7) (or (3)) and assume that the matrix A is an M-matrix and that the affine application* $AU - G$ *is perturbed by a diagonal monotone maximal operator eventually multivalued satisfying (8)*

(or (5) in the singlevalued case). Then, we can associate to problem (7) defined in the space $\mathcal{E}$ a contracting fixed point mapping denoted F_P associated to the point decomposition, and there exists one and only one fixed point $U^\star$ also unique solution of the discretized problem (7) (or (3)). Furthermore the parallel asynchronous method associated to the fixed point mapping F_P converges to $U^\star$ whatever be the initial guess U^0.

Proof. Indeed, for generality's sake, problem (7) should be considered; let us decompose this problem into M subproblems; let us write the i^{th} equation verified on the one hand for the exact solution and on the other hand for a current value of the $(p+1)^{th}$ relaxation component; subtracting member to member, multiplying each of the M relations by $(u_i^\star - u_i^{p+1})$ and taking the absolute value on both side leads to

$$\begin{aligned} 0 &\leq a_{i,i}(u_i^\star - u_i^{p+1})(u_i^\star - u_i^{p+1}) + (w_i^\star - w_i^{p+1})(u_i^\star - u_i^{p+1}) \\ &\leq \sum_{j\neq i} |a_{i,j}(u_j^\star - v_j)(u_i^\star - u_i^{p+1})|, \forall i \in \{1,\ldots,M\}, \end{aligned}$$

where $w_i^\star \in \partial\Psi_i(u_i^\star)$ and $w_i^{p+1} \in \partial\Psi_i(u_i^{p+1})$ and v_j are the available current values produced by the other processors according to the algorithm (15)-(16); then, since the subdifferential mapping is monotone the following inequality is true $(w_i^\star - w_i^{p+1}).(u_i^\star - u_i^{p+1}) \geq 0$ and the left hand-side of the previous inequality can be minored by $a_{i,i}|u_i^\star - u_i^{p+1}|^2$, which leads to

$$0 \leq a_{i,i}|u_i^\star - u_i^{p+1}|^2 \leq \sum_{j\neq i} |a_{i,j}(u_j^\star - v_j)(u_i^\star - u_i^{p+1})|, \forall i \in \{1,\ldots,M\};$$

as in the sum of the right member appears the absolute values the right-hand side of the previous inequality can be majored using a classical and straightforward inequality and since A is a M-matrix, its diagonal entries are negative or null and one can thus write $|a_{i,j}| = -a_{i,j}$ which leads finally to

$$|u_i^\star - u_i^{p+1}| \leq \sum_{j\neq i} -\frac{a_{i,j}}{a_{i,i}}|u_j^\star - v_j|, \forall i \in \{1,\ldots,M\}. \tag{17}$$

Note that the matrix J with diagonal entries that are null and off-diagonal entries equal to $-\frac{a_{i,j}}{a_{i,i}}$ is the Jacobi matrix of the matrix A. Since the matrix A is an M-matrix, then J is a nonnegative matrix and all eigenvalues of J have a modulus less than one. Let us denote by ν the spectral radius of J and by Θ the associated eigenvector; then $0 \leq \nu < 1$. Classically by the Perron-Frobenius theorem, all the components of Θ are strictly positive and the following inequality $J\Theta \leq \nu\Theta$ is valid (see [21]). Then, in a straightforward way, the result is

$$\|U^\star - U^{p+1}\|_{\nu,\Theta} \leq \nu\|U^\star - V\|_{\nu,\Theta}, \forall\ V \in \mathcal{E}, 0 \leq \nu < 1, \tag{18}$$

where $\|V\|_{\nu,\Theta}$, is a uniform weighted norm defined as follows

$$\|V\|_{\nu,\Theta} = \max_{i=1,\ldots,M} (\frac{|v_i|}{\Theta_i}); \tag{19}$$

thus, (18) shows that the fixed point mapping F_P associated to the point-decomposition is a contraction and we obtain a result of existence and uniqueness of both the fixed point $U^\star$ of F_P and of the solution of problem (7), which achieves the proof. In the case of problem (3), since Φ is also monotone, the proof follows accordingly. ■

In fact, in parallel computation, the user does not access as many processors as components. Practically, the algebraic system to solve is split into α contiguous blocks, $\alpha < M$, corresponding to a coarser decomposition, the communications consisting in exchanging only the necessary values of the components computed by the neighbouring processors. To solve the model problem (7) or (3) let us consider a parallel asynchronous block relaxation algorithm, associated to the $\alpha-$block decomposition of the discretized problem. Then several adjacent block components of the discretization matrix, and of the iterate vectors are processed accordingly by each processor. Such an implementation leads to a more multiplicative behavior of the considered subdomain methods. Furthermore, using a result of [28]-[27], if the subdomain decomposition is a point decomposition, the following result is obtained

Proposition 2. *Consider problem (7) or (3) and assume that the assumptions of Proposition 1 hold, particularly that the matrix A is an M-matrix and that the affine algebraic system is perturbed by a monotonous diagonal operator, singlevalued or multivalued. Consider any block decomposition in $\alpha < M$ blocks. Then, for any block decomposition, the associated fixed point mapping F is contractive; then there exists one and only one fixed point $U^\star$ also unique solution to the discretized problem (7) (or (3)). Furthermore, for every subdomain decomposition, the parallel asynchronous methods converge to $U^\star$ whatever the initial guess U^0 is.*

Proof. The proof is not very difficult but essentially technical. It consists of writing the equations considered in the proof of Proposition 1, then grouping them in α parts; in this context of block decomposition, using the concept of regular decomposition of M-matrix (see [29]), the result is established. So the fixed point mapping is contractive and the parallel asynchronous methods associated to each block decomposition converge for every decomposition. For more details the reader is referred to [27]. ■

Remark 2. *Moreover the considered pseudolinear problems can also be solved by subdomain methods with overlapping, like the Schwarz's alternating method; in this case, due to the augmentation process of the Schwarz's method, the pseudo-linear problems (1) are respectively written as follows*

$$\bar{A}\bar{U} - \bar{G} + \bar{\Phi}(\bar{U}) = 0. \tag{20}$$

and for problem (2) in the following way

$$\bar{A}\bar{U} - \bar{G} + \bar{\Psi}(\bar{U}) \ni 0. \tag{21}$$

Using a result of D.J. Evans and Van Deren (see [30]), if A is an M-matrix, then $\bar{A}$ is also an M-matrix. Moreover, by applying the augmentation process, the diagonal operator $\bar{\Phi}$ or $\bar{\Psi}$ are still diagonal monotone operator. So this is in the framework of the study of [28]-[27] and the results of this last papers concerning the convergence of parallel synchronous and asynchronous subdomain methods with or without overlapping can still be applied. In fact the multisplitting method presented below allows to give an unified presentation of subdomain methods with or without overlapping.

3. Parallel asynchronous multisplitting method

3.1. Analysis of the behavior of the method in a general context

Let us consider now the solution of problem (7) (or problem (3)) by the parallel asynchronous multisplitting method when A satisfies assumption (4). Let us also consider the n following regular splittings (see [29]) of matrix A

$$A = \mathcal{M}^l - \mathcal{N}^l, \quad l = 1, ..., n, n \in \mathbf{N}, \tag{22}$$

where in what follows $\mathcal{M}^l$ are block diagonal, i.e. $\mathcal{M}^l = diag(\mathcal{M}^l_i)$, $i = 1, \ldots, \alpha, l = 1, ..., n$ and moreover $(\mathcal{M}^l)^{-1} \geq 0$ and $\mathcal{N}^l \geq 0$; note that classically $\mathcal{M}^l$ are M-matrices for all l.

Let $F^l : \mathcal{D}(F^l) \to \mathcal{D}(F^l)$, $\mathcal{D}(F^l) \subset \mathcal{E}$, $l = 1, ..., n$, be n fixed point mappings associated with problem (7) (or problem (3)) and defined by:

$$F^l(V) = U, l = 1, \ldots, n, U, V \in \mathcal{D}(F^l) \subset \mathcal{E}, \tag{23}$$

such that

$$\mathcal{M}^l U = \mathcal{N}^l V + \bar{G};$$

then, for all $l = 1, ..., n$, the mapping F^l associated with problem (7) (or problem (3)) are implicitly defined by

$$G + \mathcal{N}^l V \in \mathcal{M}^l U + \partial\Psi(U) \Leftrightarrow U = F^l(V), \forall V \in \mathcal{E} = \mathcal{R}^M. \tag{24}$$

$$(G + \mathcal{N}^l V = \mathcal{M}^l U + \Phi(U) \Leftrightarrow U = F^l(V), \forall V \in \mathcal{E} = \mathcal{R}^M). \tag{25}$$

Assume that the space $E = \mathcal{E}^n$ is normed by the uniform weighted norm, defined in a similar way to (19) by

$$\|V\|_{\nu,\bar{\vartheta}} = \max_{l=1,...,n} (\max_{i=1,...,\alpha} (\frac{|V^l_i|}{(\bar{\vartheta}_l)_i})), \bar{\vartheta}_l > 0, \tag{26}$$

where here $|V_i|$ denotes any Hilbertian norm of V_i in $\mathcal{E}_i, i = 1, \ldots, \alpha$, subspace of $\mathcal{E}$; assume that for $l = 1, ..., n$, F^l is contractive with respect to U^*, its fixed point, with constant $0 \leq \nu_l < 1, l = 1, \ldots, n$ so that the following inequality is verified

$$\|F^l(V) - U^*\|_{\nu_l,\bar{\vartheta}_l} \leq \nu_l.\|V - U^*\|_{\nu_l,\bar{\vartheta}_l}, \text{ for } l = 1, ..., n. \tag{27}$$

Remark 3. *Note that for problems (7) and (3), according to the result of Proposition 2, due to the fact that A is an M-matrix and that in both cases the affine mapping $AU - G$ is perturbed by a diagonal monotone operator, then the fixed point mapping $F^l, l = 1, \ldots, n$ are contractive.*

A formal multisplitting associated with problem (7) (or problem (3)) is defined by the collection of fixed point problems (see [21])

$$U^* = F^l(U^*),\ l = 1, ..., n,\ U^* \in \mathcal{E}. \tag{28}$$

Let us now consider space $E = (\mathcal{E})^m$ like a finite product space of the n spaces $\mathcal{E}$ defined by

$$E = \prod_{l=1}^{n} \mathcal{E}_l,$$

where $\mathcal{E}_l = \mathcal{E}$ and consider the following $n-$block-decomposition of $\tilde{U} \in E$, defined by

$$\tilde{U} = \{U^1, \ldots, U^l, \ldots, U^n\} \in \prod_{l=1}^{n} \mathcal{E}_l,$$

where $U^1, \ldots, U^l, \ldots, U^n$ denote n vectors of $\mathcal{E}$.

Definition 3. *The extended fixed point mapping $\mathcal{F} : E \to E$ associated with the formal multisplitting is given as follows*

$$\mathcal{F}(\tilde{V}) = \tilde{U},\ \textit{such that } U^l = F^l(\sum_{k=1}^{n} W_{lk}V^k), l = 1, ..., n,$$

where W_{lk} are nonnegative diagonal weighting matrices satisfying for all $l \in \{1, \ldots, n\}$

$$\sum_{k=1}^{n} W_{lk} = I_l,$$

I_l being the identity matrix in $L(E_l)$.

Since $F^l(\mathcal{D}(F^l)) \subset \mathcal{D}(F^l)$, then obviously $\mathcal{F}(\mathcal{D}(\mathcal{F})) \subset \mathcal{D}(\mathcal{F})$ where $\mathcal{D}(\mathcal{F}) = \prod_{l=1}^{n} \mathcal{D}(F^l)$.

Note that considerable saving in computational work may be possible by using such a method, since a component of V^k needs not be computed if the corresponding diagonal entry of the weighting matrices is zero; then, in parallel computing, the role of such weighting matrices may be regarded as determining the distribution of the computational work of the individual processors.

Let the following block-decomposition of the mapping $\mathcal{F}$

$$\mathcal{F}(\tilde{V}) = \{\mathcal{F}^1(\tilde{V}), \ldots, \mathcal{F}^l(\tilde{V}), \ldots, \mathcal{F}^n(\tilde{V})\} \in \prod_{l=1}^{n} \mathcal{E}_l.$$

Note that for a particular choice of the weighting matrices W_{lk}, we can obtain various iterative methods and particularly on the one hand a subdomain method without overlapping and on the other hand the classical Schwarz alternating method (see [21]). According to [21] the block - Jacobi method corresponds to the following choice of $\mathcal{M}^l$

$$\mathcal{M}^l = diag(I_1, \ldots, I_{l-1}, A_{l,l}, I_{l+1}, \ldots, I_n)\ , \tag{29}$$

and to the choice of $W_{lk} \equiv \bar{W}_l$ given by

$$\bar{W}_l = diag(0, \ldots, 0, I_l, 0, \ldots, 0), \tag{30}$$

which in other words means that the entries of the weighting matrices are equal to one or to zero.

For the additive Schwarz alternating method more than one processor computes updated values of the same component, and the matrices W_l have positive entries smaller than one. The reader is referred to reference [7] and to other various references for other choices of weighting diagonal matrices and splittings for the definition of various multisplitting methods.

Let us recall now a result of [21]

Proposition 3. *Let us denote by $\tilde{U}^* = \{U^*, \ldots, U^*\}$ where U^* is the solution of (7) (or problem (3)); then if assumption (27) is verified, $\mathcal{F}$ is $\| \, . \, \|_{v,\bar{\vartheta}}$ contractive with respect to $\tilde{U}^*$, where $\| \, . \, \|_{v,\bar{\vartheta}}$ is defined by (26) the associated constant of contraction being*

$$\nu = \max_{1 \le l \le n}(\nu_l) < 1.$$

Then, using the result of Proposition 3, the following result is obtained

Corollary 1. *Consider the solution of problem (7) (or problem (3)); then if assumptions (4) - (5) (or (4) - (8)) are verified, in particular if matrix A is an M-matrix, then the formal multisplitting $\mathcal{F}$ is contractive with respect to $\tilde{U}^*$ and any sequential, parallel synchronous or asynchronous multisplitting method starting from $\tilde{U}^0 \in \mathcal{D}(\mathcal{F})$ converge to the solution of problem (7) (or problem (3)).*

Proof. Indeed, if assumption (4) is verified, and if the other assumptions of Proposition 2 concerning the monotony of the disturbing diagonal operator are satisfied, then each fixed point mapping $\hat{F}^l$ is contractive, for $l = 1, \ldots, n$ and then the proof is complete. ■

Then, for $l = 1, \ldots, n$, and for $i = 1, \ldots, \alpha$, for problem (7) such an asynchronous multisplitting method can be given by

$$\begin{cases} \mathcal{M}_i^l U_i^{l,p+1} + \omega_i(U_i^{l,p+1}) = (\mathcal{N}^l(\sum_{k=1}^{n} W_{lk} V_i^{k,\rho_k(p)}) + G)_i & , \text{if } i \in s(p) \\ U_i^{l,p+1} = U_i^{l,p} & \text{if } \ i \notin s(p) \end{cases}$$

where $\omega_i(U_i^{l,p+1}) \in \partial\Psi_i(U_i^{l,p+1})$ and accordingly for problem (3).

3.2. Application to pseudolinear problem

3.2.1. The case of singlevalued problem and Newton method

For the solution of problem (3), the block-diagonal matrix $C(W)$, derived from Newton's method is introduced, with diagonal blocks $C_i(W)$ given by

$$C_i(W) = A_{ii} + \Phi_i'(W).$$

Obviously, matrix $C_i(W)$ is an M-matrix since matrix A_{ii} is an M-matrix and Φ_i is increasing and then its derivative Φ_i' is positive.

Let U^0 given and U^k is the k-th iteration of the following algorithm:

$$U^{p+1} = U^p - C^{-1}(U^{\rho'(p)}).(AU^k + \Phi(U^k) - G), p = 0, 1, \ldots, \tag{31}$$

where $0 \le \rho'(p) \le p$.

Proposition 4. *Assume that assumptions (4) and (5) hold. Then, the sequence $\{U^p\}$ defined by (31) converges to $U^\star$ solution of problem (3).*

Proof. Obvious since C is an M-matrix. ■

3.2.2. The case of multivalued problem

Consider now the solution of the multivalued semi-linear problem (7) when assumptions (4) and (8) are verified. For such a problem, by using the same kind of proof than the previous ones, and mainly due to the monotony of the disturbing multivalued diagonal operator, a result similar to the one presented in Proposition 4 is obtained. Then it can stated that

Proposition 5. *Assume that the assumptions (4) and (8) hold. Then, the sequence $\{U^p\}$ obtained by using the parallel asynchronous multisplitting method converges to $U^\star$ solution of problem (7).*

Proof. The convergence of the considered method is analyzed by combining the monotony of the diagonal operator with the main property of the discretization matrix. Indeed, since the subdifferential of the indicator function is monotone and diagonal, the multivalued formulation allows to prove the convergence of the parallel synchronous and asynchrounous iterations in a similar way than the one used for the proof of Proposition 1. ■

Remark 4. *In the continuous case, for the solution of the convection - diffusion problem with constraints on the solution, the formulation of the problem is achieved by the perturbation of the continuous convection - diffusion operator by the multivalued increasing diagonal operator $\partial\psi(u)$. This multivalued formulation takes some interest only from a theoretical point of view. For the solution of the discretized problem (7) it is necessary to take into account the constraints and so to project the relaxed updates computed by the previous iterative algorithm on a convex set which defines such constraints as follows:*
1. first to compute an intermediate value of a component of the iterate vector by solving for example, one equation of the algebraic linear system without taking into account the constraint,
2. and then to project this intermediate value of the component on the convex set until convergence of the iterative process; more precisely, if the constraint is satisfied, the intermediate value of the component is not changed and if the constraint is saturated, for example, if a component of U is less to the corresponding component of ϱ, then the result of the projection on the convex set is such that the final update considered consists in setting the value of the component to ϱ.

4. Application to nonlinear partial differential equations

There are several kinds of partial differential equations which, after discretization, lead to the solution of pseudo-linear algebraic systems like (3) and (7). In the sequel two distinct kinds of pseudo-linear algebraic systems are considered.

4.1. Newton - multisplitting method for unconstrained boundary values problems

In the sequel Ω will denote an open domain included in $\mathbb{R}^3$, $\partial\Omega$ the boundary of Ω, g a sommable square function and $u \to \phi(u)$ a diagonal monotone increasing, convex and continuously differentiable nonlinear operator. So let us consider the following nonlinear convection diffusion problem

$$\begin{cases} -\nu\Delta u + a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} + c\frac{\partial u}{\partial z} + du + \phi(u) = g, \text{ everywhere in } \Omega, \\ u = 0, \text{ everywhere in } \partial\Omega, \end{cases} \tag{32}$$

where ν, a, b, c, d are some constant coefficients, $\nu > 0, d \geq 0$.

For example, problem (32) occurs in plasma physics or to model solar ovens [23]; such a problem arises from the implicit temporal discretization of a parabolic problem that appears in such applications modeled as follows

$$\begin{cases} \frac{\partial u(t,x)}{\partial t} - \nu\Delta u(t,x) + Q^t\nabla u + e^{ru} = g(t,x), \text{ everywhere in } [0,T]\times\Omega, b > 0, \\ u(t,x) = 0, \text{ everywhere in } [0,T]\times\partial\Omega, \\ u(0,x) = u_0(x), \text{ everywhere in } \Omega, \end{cases} \tag{33}$$

where $T > 0$, $u_0 : \Omega \to \mathbb{R}$ is the initial condition, Q is a vector with components (a, b, c).

After temporal discretization the stationary problem associated to the implicit time marching scheme, is defined as follows

$$\begin{cases} -\nu\Delta u + Q^t\nabla u + \frac{u}{\delta_\tau} + e^{ru} = g, \text{ everywhere in } \Omega \subset R^3, \\ u = 0, \text{ everywhere in } \partial\Omega, \end{cases} \tag{34}$$

where δ_τ is the time step arising in the implicit scheme.

After spatial discretization we have to solve a pseudo-linear algebraic systems of kind (1) by combining the Newton method with the parallel asynchronous multisplitting method. Note that, by choosing appropriate finite difference approximation, particularly for the convection term where according to the sign of the coefficients a, b, c, forward or backward schemes are considered so that the discretization matrix is an M-matrix; moreover since $\phi(u)$ is a diagonal monotone increasing nonlinear operator, assumptions (4) and (5) are well verified. Thus the previous parallel synchronous or asynchronous multisplitting studied method for the parallel solution of this problem can be applied.

4.2. Multisplitting method for constrained boundary values problems

Lastly note also that in the previous mathematical models, the solution u can be subject to some constraints, for example $u_{min} < u$ or $u < u_{max}$ or $u_{min} < u < u_{max}$; in such situations, the associated boundary value problem is classically formulated as the following multivalued problem (see [25])

$$\begin{cases} -\Delta u + \phi(u) + \partial\psi(u) - g \ni 0, \text{ everywhere in } \Omega, \\ u = 0, \text{ everywhere in } \partial\Omega, \end{cases} \tag{35}$$

where $\partial\psi(u)$ is the subdifferential of the indicator function $\psi(u)$ of the convex set $\mathcal{K}$ defining the constraints. It can be noted that this multivalued formulation expresses the fact that the problem is subject to some constraints. This approach is very convenient and is mainly applied to analyze the behavior of iterative algorithms used for the numerical solution of problems like (35) since the continuous diffusion operator is perturbed by the multivalued increasing operator $\partial\psi(u)$. From a practical point of view, once the problem is discretized, after each update of a block component, a projection on the convex set $\mathcal{K}$ is processed.

5. Implementation of the multisplitting method

The algorithms of the Newton-multisplitting method and the multisplitting method with projection have been implemented, both used to solve univalued and multivalued pseudo-linear problems (3) and (7) respectively.

In this paper the 3D nonlinear problems (34) presented in Section 4 are solved on a computing platform composed of n blocks, each of them composed of q processors, physically adjacent or geographically distant. Figure 1 illustrates an example of such a computing platform composed of four blocks, each of them composed of four processors. In each block a master processor is designed and the inter-communications between blocks are performed via the masters.

A 3D nonlinear problem is split into n blocks and each block (which represents a splitting in the previous mathematical description) is in turn split into q portions, as it is presented in the example of Figure 2. In this case each block of data is assigned to a block of processors and each portion to a processor.

5.1. Case of Newton-multisplitting method

In this section the multisplitting algorithm to solve nonlinear stationary convection-diffusion problems (34) presented in sub-section 4.1 is described. It should be noted that no difference is made between processors or cores.

For each splitting $l, l = 1, \ldots, n$, starting with an initial guess $U^{l,(0)}$ the following equations should be solved

$$AU^l + \Phi(U^l) - G = 0,$$

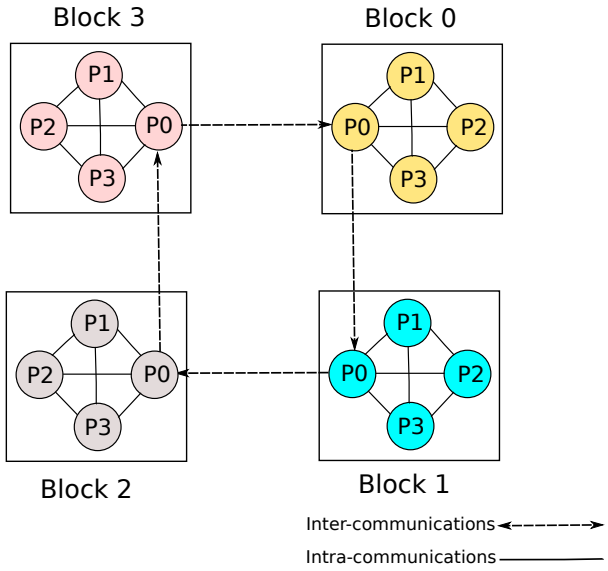


Figure 1: Example of an interconnection of four blocks.

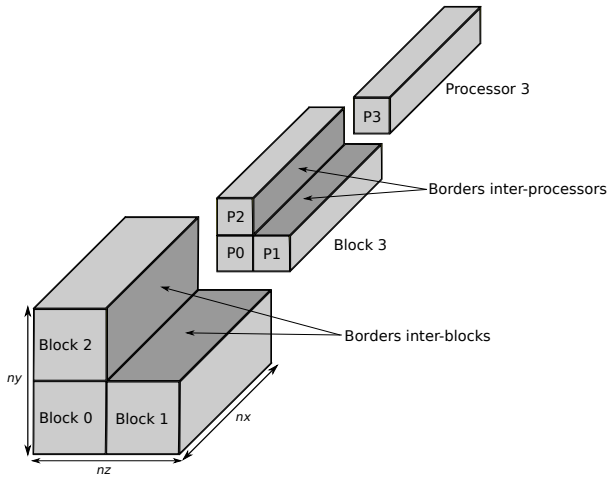


Figure 2: Example of a decomposition of a 3D problem among blocks of processors.

by the Newton method; then, globally, at step i of the Newton method it follows that

$$(A + \frac{\partial \Phi(U^{l,(i)})}{\partial U})\delta U^{l,(i)} = G - AU^{l,(i)} - \Phi(U^{l,(i)})$$

and then

$$U^{l,(i+1)} = U^{l,(i)} + \delta U^{l,(i)},$$

until convergence of the iterative method.

Let $\hat{C}(U^{l,(i)}) = A + \frac{\partial \Phi(U^{l,(i)})}{\partial U}$ and according to the choice of the weighting matrices $W_{l,k}$ let us consider a block decomposition of the matrix $\hat{C}(U^{l,(i)})$ such that

$$\hat{C}_{l,l}(U^{l,(i)}) = (A + \frac{\partial \Phi(U^{l,(i)})}{\partial U})_{l,l} \tag{36}$$

denotes the block diagonal of the matrix $\hat{C}(U^{l,(i)})$, and since the operator $U \to \Phi(U)$ is diagonal increasing, then

the Jacobian matrix of Φ, given by $\frac{\partial \Phi(U^{l,(i)})}{\partial U}$, is a positive diagonal matrix; for the same reason, due to the fact that $\Phi(U)$ is diagonal the off-diagonal blocks of $\hat{C}(U^{l,(i)})$ are reduced to the blocks $A_{l,k}$ of the matrix A. Consequently $\hat{C}(U^{l,(i)})$ is an M-matrix.

So the implementation of the Newton method requires the solution of the following linear system

$$\hat{C}(U^l)\delta U^l = \hat{G}(U^l),$$

which will therefore be solved by a multisplitting method; since the matrix $\hat{C}(U^l)$ is an M-matrix, the multisplitting method will converge according to the results stated in subsection 3.1.

For the solution of problem (1) by the Newton method and considering for example the block Jacobi method obtained by choosing $\mathcal{M}^l$ and $\tilde{W}_l$ given by (29)-(30), the implemented multisplitting method associated to the iteration number i of the Newton method leads to solve iteratively in parallel for $l = 1, \ldots, n$, the algebraic sub-systems

$$\hat{C}_{l,l}(U^{l,(i)})\delta U_l^{l,(i)} = B_l, \; l = 1, \ldots, n, \tag{37}$$

where B_l is given by

$$B_l = \hat{G}_l(U^{l,(i)}) - \sum_{k \neq l} A_{l,k}\delta U_k^{k,(j(k))}, \; l = 1, \ldots, n \tag{38}$$

and $\hat{G}(U^{l,(i)})$ is the right hand side resulting from the Newton process, i.e.

$$\hat{G}(U^{l,(i)}) = G - \Phi(U^{l,(i)}) - AU^{l,(i)}, \tag{39}$$

and the values of the components of the local vectors $\delta U_k^{k,(j(k))}$ come from the computation performed on splitting number $k, k \neq l$, and performed by other processors by using the iterate number $j(k)$ of the iterative method.

Then, in other words, each sub-system (37) is solved independently by a block (or splitting) of processors and communications are required to update the right-hand side of each sub-system. The local vectors δU_k updated according to (38) by each block represent the data dependencies between the different blocks. In the multisplitting method there are two level of iterations; outer and inner parallel iterations. In fact, since the matrices $\hat{C}_{l,l}$ are also sparse, it is highly recommended to solve the subsystems (37) by an iterative method. It is well known that iterative methods scale well. In our implementation a Krylov method was chosen to solve each block (37). It should be noted that the outer-iteration fits well within the general formulation of parallel asynchronous iterations described in sub-section 2.2, since the inter-block communications can be either synchronous or asynchronous.

The focus was put on solving 3D nonlinear single-valued systems of equations which can be formulated as shown in (1). Algorithm 1 presents the main key-points of the Newton-multisplitting method. The algorithm uses the Newton method to linearize the nonlinear system to solve (lines from 2 to 12). Then it applies the parallel multisplitting method to each linear system issued from the linearization (lines from 5 to 9), such that each system is associated to n splittings as shown in (37).

In the previous description, it should be noticed that all variables indexed with l are local to a block. First, a loop on the Newton iteration is computed. When starting a new Newton iteration, the right-hand side of the Newton process is updated. And the local sub-matrix $\hat{C}_{l,l}$ is updated. Then the second loop based on the Multisplitting iteration is computed. In the multisplitting loop, first the local right-hand side B_l is involved, taking into account the neighbor's values computed by the other processors. Then each sub-system is solved in parallel by using the well-known Krylov method GMRES [31] (line 7 in Algorithm 1). GMRES iterations represent the inner-iteration of the multisplitting method. It should be noticed that the iterations and the communications performed within the GMRES solver are synchronous inside a block of processors. At the end of the Multisplitting iteration, each local solution of the sub-system is exchanged to all neighbor blocks (see line 8). On line 10, the local solution on block l is updated, then on line 11, the global correction $\delta\mathbf{U}^{(i+1)}$ is performed.

The global convergence of the synchronous Newton-multisplitting method is detected when the global value of $\delta\mathbf{U}^{(i+1)}$ is stabilized corresponding to the following stopping test

$$\|\delta\mathbf{U}^{(i+1)}\|_2 \geq \varepsilon_{Newton} \tag{40}$$

where ε_{Newton} is the tolerance threshold for the computation of the Newton method.

The parallel algorithms described in this section are implemented in two ways: synchronous and asynchronous. For the parallel *synchronous* algorithms, we use $MPI\_Send()$ and $MPI\_Recv()$ routines to perform the communications. In contrast, for the parallel asynchronous algorithms, we use the MPI non-blocking communications $MPI\_Isend()$, $MPI\_Irecv()$ and $MPI\_Test()$. The reduction operations are implemented using $MPI\_Allreduce()$.

In the asynchronous version, the global convergence is detected when all blocks have locally converged. The convergence detection implemented in [32] was used. In fact, as is shown in Figure 1, a virtual unidirectional ring network is implemented between masters of the blocks of processors. At the beginning of the resolution, a Boolean token is set to *False* by the master of block 0. During the computation, the Boolean token circulates around the virtual ring network until the global convergence. Locally at the end of the resolution of a sub-system, each master of block i updates the value of the token by evaluating the following logical expression:

```
token = token && is_locally_converged_i()
```

where $is\_locally\_converged_i()$ returns *True* if the local convergence is achieved or *False* otherwise. Then the token is sent to the master of block $i+1$ if $i < n-1$ or to the master of block 0 otherwise. The global convergence is detected when the master of block 0 receives from the master of block $n-1$ a token set to *True*. To this effect, the master of block 0 broadcasts a message to all masters of other blocks of processors in order to stop the computation.

Algorithm 1: Parallel nonlinear multi-splitting method performed on a block of processors

Output: Solution $U^{l,(i+1)}$
Set initial solution: $U^{l,(0)} = 1.0$
while $\|\delta \mathbf{U}^{(i+1)}\|_2 \geq \varepsilon_{Newton}$ **do**
    Compute the right-hand side of Newton $\hat{G}(U^{l,(i)})$: Formula (39)
    Update local sub-matrix $\hat{C}_{l,l}$: Formula (36)
    while $\|\delta U^{l,(i)} - \delta U^{l,(i-1)}\|_\infty \geq \varepsilon_{Multisplitting}$ **do**
        Compute local right-hand side: B_l Formula (38)
        Parallel GMRES to solve $\delta U^{l,(i)}$: (Formula 37)
        Exchange local shared values of $\delta U^{l,(i)}$ with neighbor blocks
    end
    Compute the local solution on block l: $U^{l,(i+1)} = U^{l,(i)} + \delta\, U^{l,(i)}$
    Compute the global correction $\delta \mathbf{U}^{(i+1)}$
end

5.2. *Case of multisplitting method with projection*

This sub-section shows the implementation of the multisplitting method for solving a boundary value problem of the kind (35) where the solution is subject to inequality constraints. The principle of implementation is the same as the one considered in the previous sub-section. In fact, the implementation is even simpler since we have to solve the linear system associated to the sub-problem without taking into account the constraints and then project the intermediate solution $\bar{U}_l$ on the convex set defining the constraints as follows.

$$A_{l,l}.\bar{U}_l = G_l - \sum_{k \neq l} A_{l,k} U_k \tag{41}$$

and then $\bar{U}_l$ is obtained by a projection

$$\bar{U} = Proj(\bar{U}_l) \tag{42}$$

It should also be noted that the formulation (35) is not used directly to code the algorithms, because this last formulation is a formal formulation of the problem to be solved. In particular, it is not necessary to discretize the sub-differential operator of the indicator function of the convex set defining the constraints on the solution. It is therefore sufficient to discretize by classical methods, on the one hand the partial derivative operator taking into account the boundary conditions, and on the other hand the second member of the partial derivative equation.

Algorithm 2: Parallel projected multi-splitting method

Output: Solution U^l

Compute local sub-matrix $A_{l,l}$
norm=1
while $norm \geq \varepsilon_{global}$ **do**
    Compute local right-hand side: $B_l = G_l - \sum_{k \neq l} A_{l,k} U_k$
    Parallel GMRES to solve $\bar{U}^l$ from $A_{l,l}\bar{U}_l = B_l$
    $\bar{U}_l$=max($\bar{U}_l$,constraint)
    $norm_l = |(U_l - \bar{U}_l)|$;
    $norm = max_l(norm_l)$
    $U_l = \bar{U}_l$
    Exchange local shared values of U^l with neighbor blocks
end

Algorithm 2 presents the main key-points of our synchronous or asynchronous multisplitting method in order to solve the discrete problem subject to constraints. All variables are local to a block, except *norm* which represents the global norm defined as the max of all the local norms. As Algorithm 2 has many similarities to Algorithm 1 except that there is no Newton iteration, all the previous explanations and remarks are valid.

6. Parallel experiments

In paper [24], experiments were achieved on a single cluster. In present version, larger experiments are made on a grid. The reason is that making parallel computation on a distributed grid with geographically distant sites is more difficult and challenging than using a supercomputer. In addition, when one have to solve very large algebraic systems, impossible to solve on an unique cluster with few processors, grid environment is very interesting to use.

So, parallel experiments have been performed on Grid'5000 a French grid platform which is a large-scale and flexible testbed for experiment-driven research in all areas of computer science, with a focus on parallel and distributed computing [33]. Actually this environment provides access to a large amount of resources: 15358 cores, 840 compute-nodes grouped in 37 homogeneous clusters on 8 sites, connected with a dedicated 10-Gbps backbone network and featuring various technologies for example GPU, different CPU type, ssd, 10G and 25G Ethernet.

Our parallel simulations for the solution of the different problems have been performed on various distant clusters connected by highspeed communication of the Grid5000 platform. At that time, on Grid'5000, the speed of communication was about a Gigabit Ethernet network for local machine on each cluster and links between different sites range from 10-Gbps.

Figure 3 illustrates the sites of the Grid'5000 architecture.

Let us consider problems (34) and (35), solved respectively by parallel asynchronous Newton-multisplitting method and parallel asynchronous multisplitting algorithm with projection. For the parallel simulations performed on distant clusters of the Grid'5000 platform, several sizes of the algebraic systems to solve have been considered. Indeed the problems are defined on a cube $\Omega \equiv [0, 1[^3$ and we consider in the achieved simulations on each axis 240, 300, 360, 420 and/or 480 discretization points. So the size of the algebraic systems to solve are respectively 240^3, 300^3, 360^3, 420^3 and/or 480^3 (see Table 1).

The parallel experiments were carried out by using the cluster named "parapide" or "paravance" of Rennes, the cluster named "ecotype" of Nantes and the cluster named "suno" of Sophia. Table 2 gives the characteristics of each machine used.

So, readers can find below the results for 240^3, 300^3, 360^3, 420^3 and/or 480^3 size of algebraic systems discretized by a finite difference scheme, on a grid composed of two or three distant clusters. The code written in C language is parallelized with facilities provided by Open MPI v2. The MPI machine file alternates dedicated computers on clusters to reach the desired number of cores to perform the experiments.

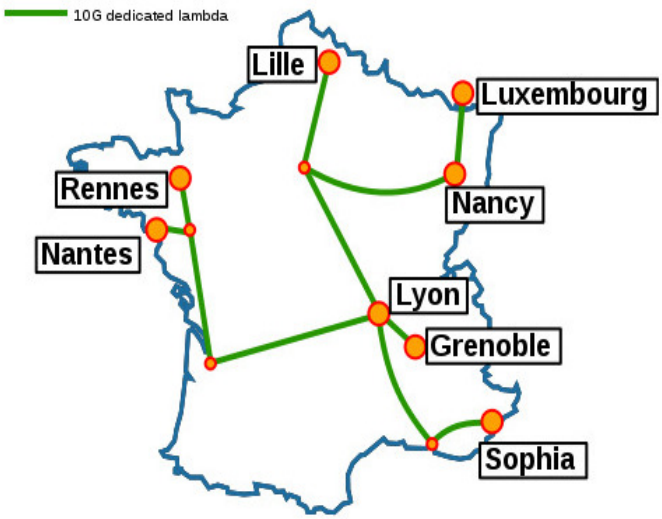


Figure 3: Grid'5000 architecture.

Size of the algebraic systems	
$240^3 = 13\,824\,000$	$420^3 = 74\,088\,000$
$300^3 = 27\,000\,000$	$480^3 = 110\,592\,000$
$360^3 = 46\,656\,000$	

Table 1: Size of the algebraic systems.

Site	Cluster	Processors type	GHz	CPU	cores/CPU	RAM size	more information
Nantes	ecotype	Intel Xeon E5-2630L v4	1.8	2	10	128 GB	[34]
Rennes	parapide	Intel Xeon X5570	2.9	2	4	25 GB	[35]
Rennes	paravance	Intel Xeon E5-2630 v3	2.4	2	8	128 GB	[36]
Sophia	suno	Intel Xeon E5520	2.27	2	4	32 GB	[37]

Table 2: Characteristics of machines on each site.

In these simulations, the number of blocks was fixed to $n = 2$ since it has been observed that this enables one to obtain the best performances. So in all the results, $n \times q$ configuration were used. Moreover, the cores of each block q are randomly deployed on the distant clusters used.

In order to measure the efficiency of the asynchronous methods compared to the synchronous ones, the values of τ which is the ratio of the synchronous and asynchronous computation time is used.

For each simulation, the tolerance thresholds are used: for GMRES 10^{-12}, for the Multisplitting 10^{-7} and for the Newton method 10^{-4}. The values for problem (32) of a, b, c and d are respectively $a = 0.1$, $b = 0.2$, $c = 0.3$ and $d = 0$.

6.1. Parallel simulations for synchronous and asynchronous Newton - multisplitting algorithm and multisplitting algorithm with projection.

Several simulations have been carried out for each problem. The following Table 3 resumes cluster and grid tests on Grid'5000 platform.

The results of each grid simulations for synchronous and asynchronous versions are summarized in Tables 4 to 11 for the Newton - multisplitting algorithm and in Tables 14 to 19 for the multisplitting algorithm with projection.

Experimentations					Newton - multisplitting			Multisplitting with projection		
Clusters			Cores	Machines	Tables		Figures	Tables		Figures
					Sync	Async		Sync	Async	
paravance			600	about 38	4	5	4			
ecotype		parapide	600	about 40	6	7	5	14	15	9
ecotype	parapide	suno	600	about 48	8	9	6	16	17	10
ecotype	paravance	suno	1000	about 64	10	11	7	18	19	11

Table 3: Simulations on GRID'5000.

In these Tables, the values of domain size, average iteration number by core to reach convergence followed by GMRES number of iterations, elapsed and communication times are detailed. For the Newton - multisplitting algorithm, the number of iterations necessary to reach convergence for linear systems derived from the Newton method is mentioned and the Newton iterations is indicated in parentheses. In addition, in Tables which contain asynchronous results, the values of τ for each problem is given.

The results of the synchronous and asynchronous elapsed times and communications times with respect to the different domain sizes are presented on Figures 4 to 7 for the first problem and on Figures 9 to 11 for the problem with projection.

6.1.1. Newton - multisplitting algorithm

Results with 64 *cores on single cluster.*

As previously said, in paper [24], experiments for Newton - multisplitting algorithm were achieved on a single cluster where the dimension of the 3D problem was discretized in 150 elements, with a size of algebraic system to solve of 150^3 using up to 64 cores. Experiments have been achieved on the mesocentre of the University of Franche-Comt and machines was composed of Xeon(R) CPU E5-2640 v3 @ 2.60GHz processors.

In the present experiments, we have performed additional tests on single cluster constituting one cluster of Grid'5000.

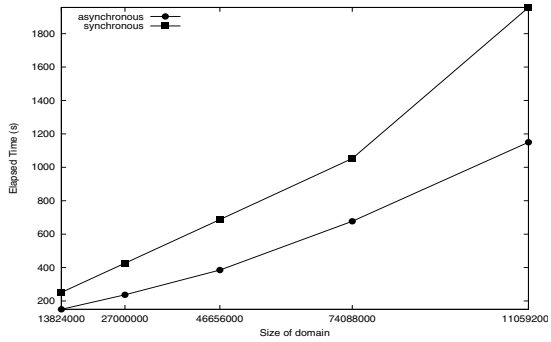
Results with 600 *cores on single cluster with* $n = 2$ *and* $q = 300$*.*

Table 4: Domain size, iterations, elapsed and communication time on cluster (parapide) with synchronous parallel algorithm.

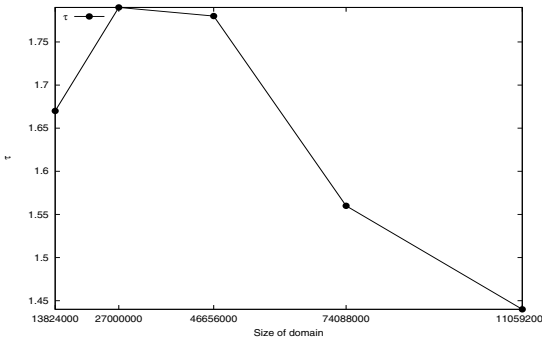
Synchronous results				
Size	Iterations		Elapsed time (sec)	Communication time (sec)
	Multi (Nwt)	GMRES		
240^3	4 229 (3)	21 145	251	63
300^3	6 387 (3)	31 935	426	103
360^3	8 827 (3)	44 135	687	157
420^3	11 785 (3)	58 925	1 053	224
480^3	15 025 (3)	75 125	1 657	363

Table 5: Domain size, iterations, elapsed and communication time on cluster (parapide) with asynchronous parallel algorithm.

Asynchronous results					
Size	Iterations		Elapsed time (sec)	Communication time (sec)	τ
	Multi (Nwt)	GMRES			
240^3	3 191 (17)	15 957	150	38	1.67
300^3	4 125 (5)	20 625	237	55	1.79
360^3	6 006 (26)	30 030	385	86	1.78
420^3	7 641 (39)	38 207	677	137	1.56
480^3	9 745 (33)	48 725	1 150	205	1.44



(a) Elapsed time with respect to domain sizes.



(b) Variation of the value τ with respect to the domain sizes.

Figure 4: Newton - multisplitting algorithm : Results with 600 cores on 1 cluster

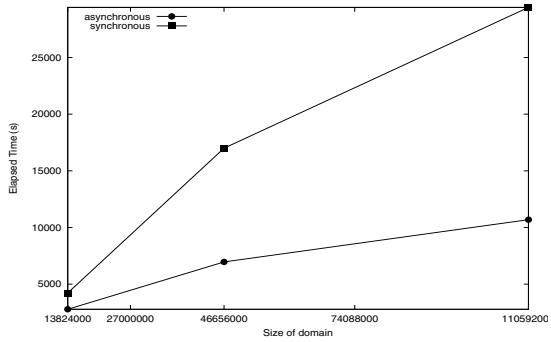
Results with 600 *cores on* 2 *clusters with* $n = 2$ *and* $q = 300$*.*

Table 6: Domain size, iterations, elapsed time on grid (ecotype, parapide) with synchronous and asynchronous parallel algorithm.

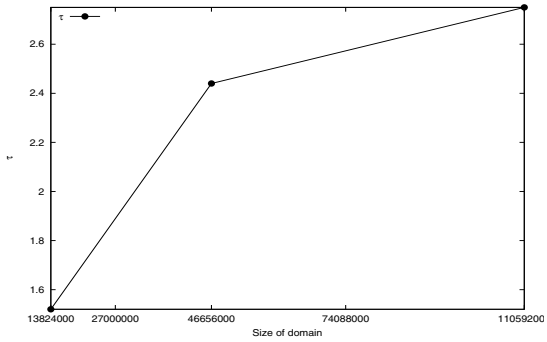
Synchronous results			
Size	Iterations		Elapsed time (sec)
	Multi (Nwt)	GMRES	Total
240^3	4 229 (3)	21 145	4 213
360^3	8 827 (3)	44 135	16 999
480^3	15 025 (3)	75 125	29 425

Table 7: Domain size, iterations, elapsed time on grid (ecotype, parapide) with asynchronous parallel algorithm.

Asynchronous results				
Size	Iterations		Elapsed time (sec)	
	Multi (Nwt)	GMRES	Total	τ
240^3	3 197 (10)	15 987	2 779	1.52
360^3	6 036 (39)	30 180	6 959	2.44
480^3	9 001 (53)	45 005	10 689	2.75



(a) Elapsed time with respect to domain sizes.



(b) Variation of the value τ with respect to the domain sizes.

Figure 5: Newton - multisplitting algorithm : Results with 600 cores on 2 clusters

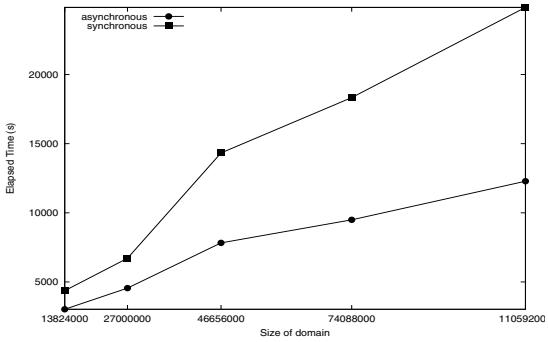
Results with 600 *cores on* 3 *clusters with* $n = 2$ *and* $q = 300$.

Table 8: Domain size, iterations, elapsed and communication time on grid (ecotype, parapide, suno) with synchronous parallel algorithm.

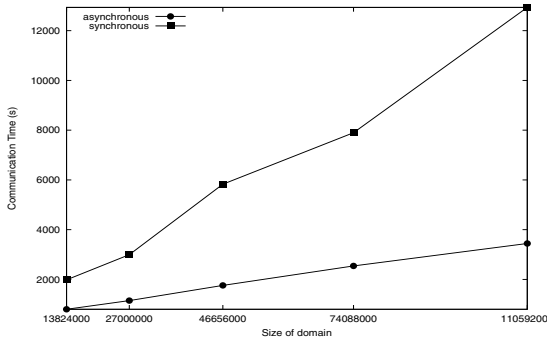
Synchronous results				
Size	Iterations		Elapsed time (sec)	Communication time (sec)
	Multi (Nwt)	GMRES		
240^3	4 229 (3)	21 145	4 368	1 099
300^3	6 387 (3)	31 935	6 713	1 711
360^3	8 827 (3)	44 135	14 332	3 217
420^3	11 785 (3)	58 925	18 331	4 833
480^3	15 025 (3)	75 125	24 855	6 933

Table 9: Domain size, iterations, elapsed and communication time on grid (ecotype, parapide, suno) with asynchronous parallel algorithm.

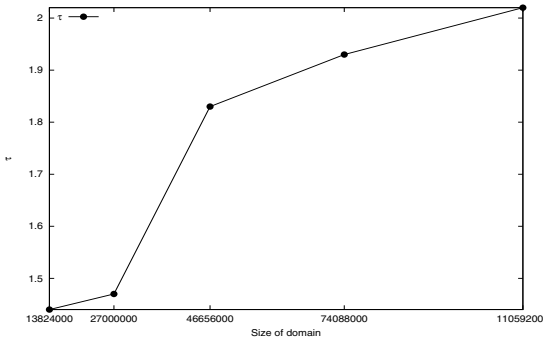
Asynchronous results					
Size	Iterations		Elapsed time (sec)	Communication time (sec)	τ
	Multi (Nwt)	GMRES			
240^3	3 272 (8)	16 360	3 025	800	1.44
300^3	5 809 (9)	29 045	4 553	1 150	1.47
360^3	5 375 (9)	26 875	7 828	1 760	1.83
420^3	6 836 (19)	34 180	9 495	2 541	1.93
480^3	8 147 (39)	40 735	12 287	3 446	2.02



(a) Elapsed time with respect to domain sizes.



(b) Communication time with respect to domain sizes.



(c) Variation of the value τ with respect to the domain sizes.

Figure 6: Newton - multisplitting algorithm : Results with 600 cores on 3 clusters

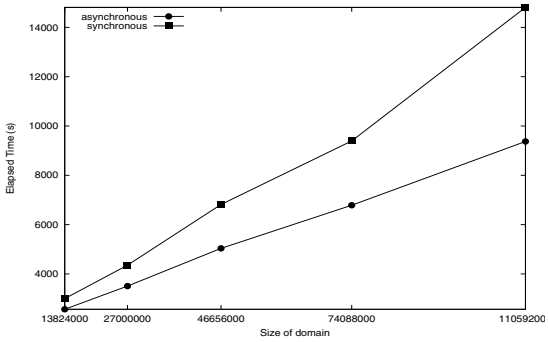
Results with 1000 *cores on* 3 *clusters with* $n = 2$ *and* $q = 500$.

Table 10: Domain size, iterations, elapsed and communication time on grid (ecotype, paravance, suno) with synchronous parallel algorithm.

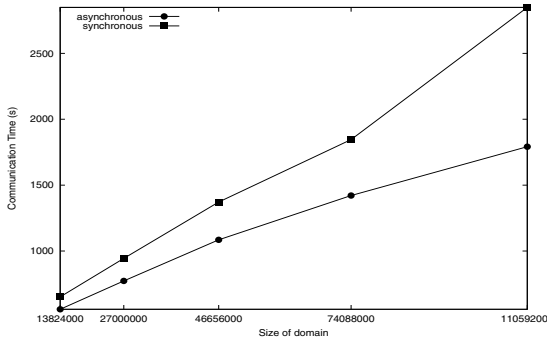
Synchronous results				
Size	Iterations		Elapsed time (sec)	Communication time (sec)
	Multi (Nwt)	GMRES		
240^3	2 626 (20)	13 130	2 998	652
300^3	3 756 (14)	18 780	4 355	944
360^3	4 964 (63)	24 820	6 816	1 372
420^3	6 710 (62)	33 550	9 395	1 846
480^3	8 939 (7)	44 695	14 813	2 849

Table 11: Domain size, iterations, elapsed and communication time on grid (ecotype, paravance, suno) with asynchronous parallel algorithm.

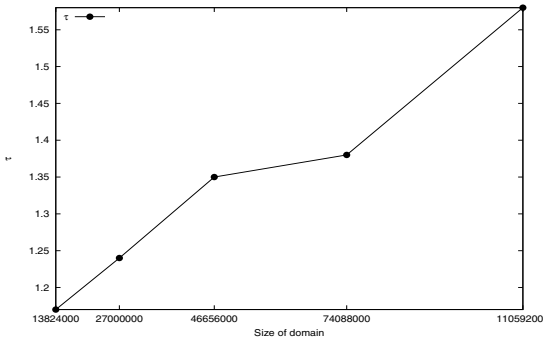
Asynchronous results					
Size	Iterations		Elapsed time (sec)	Communication time (sec)	τ
	Multi (Nwt)	GMRES			
240^3	2 276 (393)	11 382	2 565	558	1.17
300^3	3 158 (415)	15 792	3 505	773	1.24
360^3	4 324 (843)	21 622	5 036	1 085	1.35
420^3	5 603 (1 235)	28 017	6 787	1 421	1.38
480^3	6 313 (1 064)	31 567	9 374	1 792	1.58



(a) Elapsed time with respect to domain sizes.



(b) Communication time with respect to domain sizes.



(c) Variation of the value τ with respect to the domain sizes.

Figure 7: Newton - multisplitting algorithm : Results with 1000 cores on 3 clusters

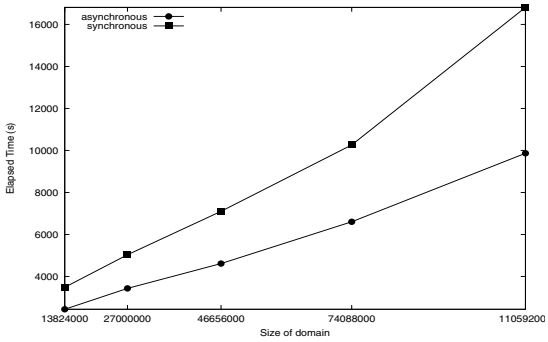
Results with 1000 *cores on* 3 *clusters with* $n = 4$ *and* $q = 250$*.*

Table 12: Domain size, iterations, elapsed and communication time on grid (ecotype, paravance, suno) with synchronous parallel algorithm.

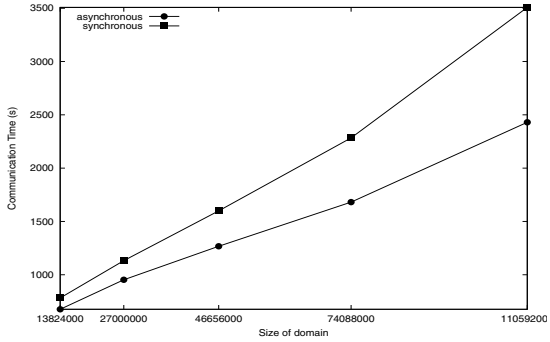
Synchronous results				
Size	Iterations		Elapsed time (sec)	Communication time (sec)
	Multi (Nwt)	GMRES		
240^3	3 186 (7)	15 930	3 483	782
300^3	4 586 (5)	22 930	5 035	1 132
360^3	6 412 (10)	32 065	7 100	1 599
420^3	8 400 (10)	42 000	10 261	2 284
480^3	10 491 (8)	52 455	16 812	3 508

Table 13: Domain size, iterations, elapsed and communication time on grid (ecotype, paravance, suno) with asynchronous parallel algorithm.

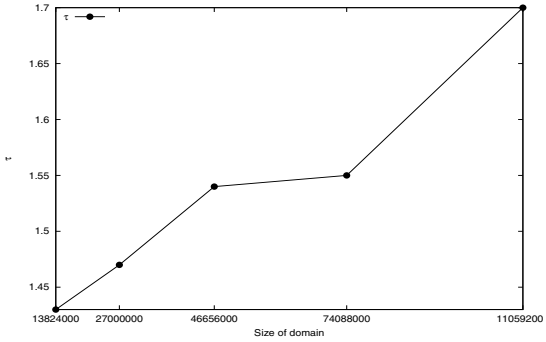
Asynchronous results					
Size	Iterations		Elapsed time (sec)	Communication time (sec)	τ
	Multi (Nwt)	GMRES			
240^3	15 664 (17)	78 322	2 437	676	1.43
300^3	21 210 (24)	106 051	3 431	953	1.47
360^3	25 563 (71)	132 817	4 613	1 267	1.54
420^3	32 612 (448)	163 061	6 604	1 681	1.55
480^3	42 609 (597)	213 047	9 864	2 430	1.70



(a) Elapsed time with respect to domain sizes.



(b) Communication time with respect to domain sizes.



(c) Variation of the value τ with respect to the domain sizes.

Figure 8: Newton - multisplitting algorithm : Results with 1000 cores on 3 clusters

6.1.2. Multisplitting problem with projection

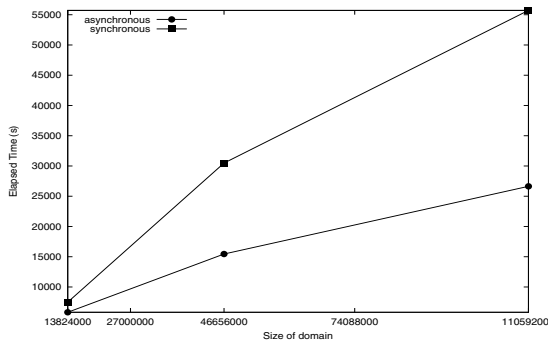
Results with 600 *cores on* 2 *clusters with* $n = 2$ *and* $q = 300$.

Table 14: Domain size, iterations, elapsed time on grid (ecotype, parapide) with synchronous parallel algorithm.

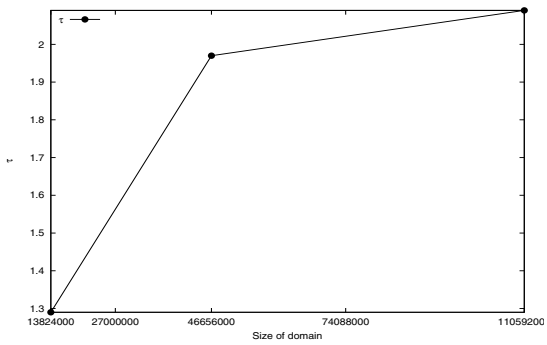
Synchronous results			
Size	Iterations		Elapsed time (sec)
	Multi	GMRES	Total
240^3	7 481	37 405	7 496
360^3	15 814	79 070	30 484
480^3	28 538	142 690	55 725

Table 15: Domain size, iterations, elapsed time on grid (ecotype, parapide) with asynchronous parallel algorithm.

Asynchronous results				
Size	Iterations		Elapsed time (sec)	
	Multi	GMRES	Total	τ
240^3	6 501	32 507	5 823	1.29
360^3	12 844	64 222	15 452	1.97
480^3	21 221	106 105	26 642	2.09



(a) Elapsed time with respect to domain sizes.



(b) Variation of the value τ with respect to the domain sizes.

Figure 9: Multisplitting problem with projection : Results with 600 cores on 2 clusters

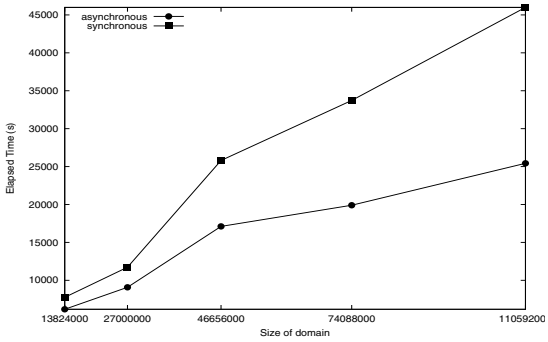
Results with 600 *cores on* 3 *clusters with* $n = 2$ *and* $q = 300$*.*

Table 16: Domain size, iterations, elapsed and communication time on grid (ecotype, parapide, suno) with synchronous parallel algorithm.

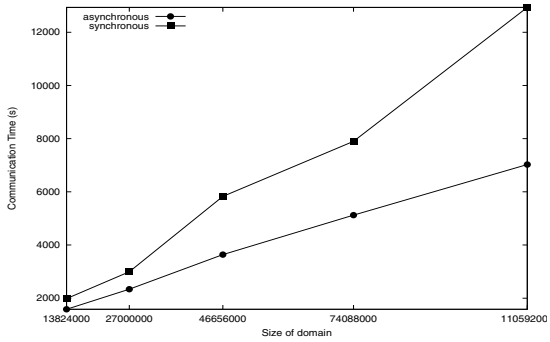
Synchronous results				
Size	Iterations		Elapsed time (sec)	Communication time (sec)
	Multi	GMRES		
240^3	7 481	37 405	7 759	1 984
300^3	11 249	56 245	11 734	2 994
360^3	15 814	79 070	25 808	5 828
420^3	21 030	105 150	33 727	7 898
480^3	28 538	142 690	45 997	12 936

Table 17: Domain size, iterations, elapsed and communication time on grid (ecotype, parapide, suno) with asynchronous parallel algorithm.

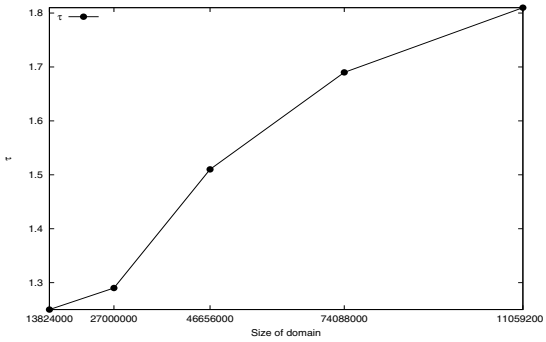
Asynchronous results					
Size	Iterations		Elapsed time (sec)	Communication time (sec)	τ
	Multi	GMRES			
240^3	6 483	32 415	6 185	1 583	1.25
300^3	9 429	47 145	9 080	2 339	1.29
360^3	11 241	56 205	17 114	3 639	1.51
420^3	13 430	67 150	19 903	5 123	1.69
480^3	16 733	83 665	25 437	7 026	1.81



(a) Elapsed time with respect to domain sizes.



(b) Communication time with respect to domain sizes.



(c) Variation of the value τ with respect to the domain sizes.

Figure 10: Multisplitting problem with projection : Results with 600 cores on 3 clusters

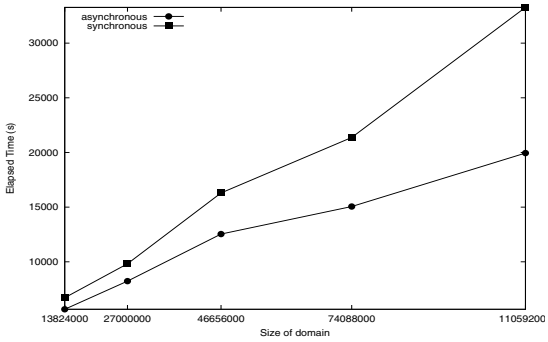
Results with 1000 *cores on* 3 *clusters with* $n = 2$ *and* $q = 500$*.*

Table 18: Domain size, iterations, elapsed and communication time on grid (ecotype, paravance, suno) with synchronous parallel algorithm.

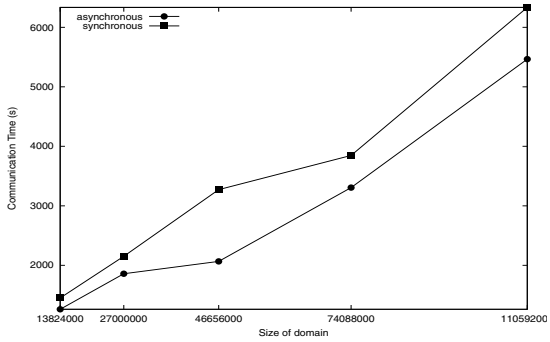
Synchronous results				
Size	Iterations		Elapsed time (sec)	Communication time (sec)
	Multi	GMRES		
240^3	5 838	29 190	6 707	1 451
300^3	8 551	42 755	9 820	2 151
360^3	11 898	59 490	16 298	3 272
420^3	15 296	76 480	21 368	3 847
480^3	19 448	97 240	33 272	6 335

Table 19: Domain size, iterations, elapsed and communication time on grid (ecotype, paravance, suno) with asynchronous parallel algorithm.

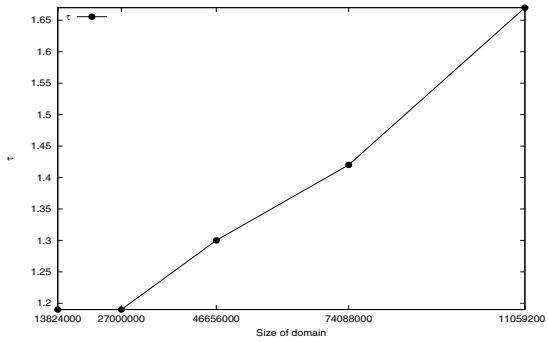
Asynchronous results					
Size	Iterations		Elapsed time (sec)	Communication time (sec)	τ
	Multi	GMRES			
240^3	5 286	26 432	5 654	1 261	1.19
300^3	7 680	38 400	8 227	1 858	1.19
360^3	9 330	46 715	12 537	2 066	1.30
420^3	13 427	67 135	15 056	3 306	1.42
480^3	18 749	93 347	19 943	5 465	1.67



(a) Elapsed time with respect to domain sizes.



(b) Communication time with respect to domain sizes.



(c) Variation of the value τ with respect to the domain sizes.

Figure 11: Multisplitting problem with projection : Results with 1000 cores on 3 clusters

6.2. Experiments analysis.

Results on a single cluster.

For the experiment on a single cluster, the asynchronous elapsed time is better than the synchronous one but the ratio τ shows a degradation when the size increases (see Tables 4, 5 and Figure 4). The communications and the execution times are very fast between the cores of a cluster which removes the benefit of the asynchronous compared to the synchronous. Indeed, the different simulations show that the best performances for asynchronous algorithms are in the context of grid use with remote and heterogeneous configurations. In view of these results, no further experiments were carried out on a single cluster because of degraded communications and limited simulation size. So, the simulations focused on grid experiments for the two problems considered.

Results on grid.

The experiments were carried out on grids with machines distributed over several remote sites using 600 and 1000 cores on 2 and 3 clusters.

In order to verify our assertion concerning the choice of $n = 2$, a simulation on 3 clusters was carried out on 1000 cores and $n = 4$ blocks (see Tables 12, 13 and Figure 8).

This simulation shows good results for the asynchronous algorithm. Concerning the synchronous algorithm, $n = 4$ blocks increases the execution time and the communication time compared to a $n = 2$ blocks. For the asynchronous, this configuration increases the number of iterations to reach convergence and the communication time. In view of these results, the simulations were carried out by taking $n = 2$ blocks (see Table 3).

In our experiments, the elapsed times obtained for the asynchronous method is largely better than the ones obtained for the synchronous method; this is essentially due to the latency of the network which implies costly communications and also due, as previously said, to the choice of a number of blocks equal to 2. It can be observed that the parameter τ, the ratio of the synchronous and asynchronous computation time which allows to

measure suitably the efficiency of the asynchronous methods compared to the synchronous ones, shows us good performances.

In Table 6 and Table 7, parallel simulations for the Newton - multisplitting algorithm on grid composed of two clusters with 600 cores show that the asynchronous mode is more efficient than the synchronous one. The parallel asynchronous simulation times are clearly inferior to those obtained in the synchronous mode and it can be observed that the values of τ vary between 1.52 and 2.75. So for the size of 480^3 the asynchronous simulation works almost three times faster than the synchronous version. In view of the results indicated in Table 6 and Table 7, it appears that the Newton method converges quickly. It is therefore the resolution of linear systems resulting from the Newton method that requires a large amount of computation (see Tables 6, 7 and Figure 5).

Similarly, in Table 14 and Table 15, the parallel simulations for the multisplitting algorithm with projection achieved on grid also show that the asynchronous mode is, once again, more efficient than the synchronous mode and the values of τ vary between 1.29 and 2.09. In this case, the size of the algebraic system to solve is equal to 480^3 and the asynchronous simulation works almost twice as fast as the synchronous version (see Tables 14, 15 and Figure 9).

The results obtained for 600 cores on 3 clusters in Tables 8 and 9 show that the execution times increase due to the increase of time communication. In the end, the asynchronous algorithm is more efficient than the synchronous algorithm and the values of τ vary between 1.44 and 2.02, i.e. for 480^3 the asynchronous simulation is twice as fast as the synchronous one. In Tables 16, 17, the simulations also show a degradation of the results due to the increased communications when increasing the number of the remote site but confirms the good performance of the asynchronous versus synchronous results, the value of τ varying between 1.25 and 1.81. (see Tables 8, 9, 16, 17 and Figures 6, 10).

Finally, for the results obtained for 1000 cores on 3 clusters in Tables 10, 11, 18 and 19, the execution times are lower due to the finer cutting date on cores. The asynchronous method is still interesting compared to the synchronous method with values of τ between 1.17 and 1.58 for the first problem and 1.19 to 1.67 for the second problem with projection (Tables 10, 11, 18, 19 and Figures 7, 11)

It should be noted that the asynchronous version of the calculation code requires fewer iterations than the synchronous one. In fact, this reduction of iterations is related to the process in which the global problem is divided into sub-problems, particularly for the resolution of the linear systems derived from the Newton linearization. With the splitting used, the asynchronous version has a more marked multiplicative behavior than the synchronous version. Thus, in the asynchronous mode, the updates of interaction values performed by processors lead to an acceleration convergence.

Thus, for the two non-linear problems tested, there is a difference in behaviour in sequential mode between the Gauss-Seidel method and the Jacobi method to solve a Poisson equation; in this case the Gauss-Seidel method converges twice as fast as the Jacobi method. Thus, when relaxation methods are parallelized, this type of algorithm behavior tends to be preserved for the resolution of pseudo-linear problems with or without constraints.

7. Conclusion

In the present study we have presented a formulation of multisplitting algorithms to find the solution of diagonal subproblems. Such a calculation method has been used for the solution of univalued or multivalued pseudo-linear stationary problems and implemented in a computing platform composed of n blocks, each of them composed of q processors, physically adjacent or geographically distant. The performance of asynchronous parallel iterative methods are better than the synchronous ones. In the following appendix, we will present an experimental and some theoretical arguments concerning the behavior of the parallelized Newton method. For synchronous implementation, the quadratic behavior of Newton's method is preserved. Conversely, for an asynchronous implementation due to chaotic communication between processes, the quadratic convergence is not preserved; however, the additional computation has the positive effect of improving the numerical quality of the solution. In future work, from an experimental point of view, the use of such mixed method for the solution of univalued pseudo-linear stationary problem, arising in boundary value, on cloud architecture, will be considered. Besides, from a theoretical point of view, we intend to analyze non-linear multisplitting methods, using the discrete maximum principle.

8. Acknowledgment

This study has been made possible with the support of Grid'5000 platform. This work was partially supported by the EIPHI Graduate School (contract "ANR-17-EURE-0002").

9. Appendix

In this appendix we have therefore plotted the maximum error obtained in both synchronous and asynchronous mode of Newton method, as a function of the iteration number, which makes it possible to visualize the behavior of the parallelized Newton method.

In the following the results of these experiments are commented, both on an experimental and a theoretical level, although on the latter point the question seems to be largely open. In view of the experimental results, it can be seen that the number of Newton iterations is not the same in synchronous and asynchronous modes.

The Newton method being a fixed point iteration of the type $\delta U = H(\delta V)$, what regulates the speed of convergence is the contraction constant μ defined by

$$\|H(\delta V) - H(\delta U^*)\| \leq \mu \|\delta V - \delta U^*\|, \forall\ \delta V, 0 < \mu < 1,$$

where δU^* is the fixed point.

In this case, a linear convergence is achieved. Note that this type of inequality in the result of Proposition 3 and Corollary 1 has been obtained.

Classically in the sequential case, it is known that the Newton method converges in a quadratic way towards the solution of the problem (see book [38] in particular the 10.7 theorem). The quadratic convergence of the Newton process, corresponding to the fixed point equation $\delta U = H(\delta V)$, comes in particular from the fact that the partial first derivatives of H with respect to the components of δV are zero at the fixed point δU^* according to the following theorem of [38].

Theorem 1. *Suppose δU^* is a solution of $H(\delta U) = \delta U$, where $H(\delta U) = \delta U - C^{-1}(U)(AU + \Phi(U) - G)$, for some function $H = (h_1, \dots h_M)$ mapping $\mathbb{R}^M$ into $\mathbb{R}^M$.*

If a number $\tau > 0$ exists with the property that

- *$\frac{\partial h_k}{\partial \delta v_l}$ is continuous on $N_\tau = \{\delta V \mid \|\delta V - \delta U^*\| < \tau\}$ for each $k, l = 1, \dots M$,*
- *$\frac{\partial^2 h_k}{\partial \delta v_l \partial \delta v_j}$ is continuous and $|\frac{\partial^2 h_k}{\partial \delta v_l \partial \delta v_j}| \leq P$ for some constant P, whenever $\delta V \in N_\tau$ for each $j, k, l = 1, \dots M$,*
- *$\frac{\partial h_k(\delta U^*)}{\partial \delta u_l} = 0$ for each $k, l = 1, \dots M$,*

then the sequence generated by $\delta U^{(i+1)} = H(\delta U^{(i)})$ converges quadratically to δU^* for any initial guess $\delta U^{(0)}$ provided that $\|\delta U^{(0)} - \delta U^*\| < \tau$.

Moreover

$$\|\delta U^{(i+1)} - \delta U^*\|_\infty \leq \frac{M^2 P}{2} \|\delta U^{(i)} - \delta U^*\|_\infty^2 \text{ for } i > 0.$$

Considering the synchronous parallel implementation of the Newton method, which in fact reproduces the sequential implementation of this method, a quadratic convergence is logically expected. Note that in this case the convergence will be faster than announced in the result of Proposition 3 and Corollary 1.

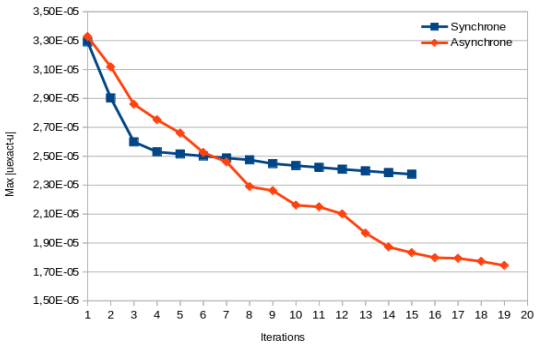
On the other hand, if we now consider an asynchronous parallel implementation of the Newton method, the Jacobian matrix will be calculated with component values delayed because of asynchronous communications. However, the asynchronous algorithm being non-deterministic, due to the fact that on the one hand the vector to be computed is decomposed into blocks and on the other hand, when asynchronous iterations are performed, the updates are chaotic contrary to the sequential or the synchronous methods, it is not obvious that the partial derivatives of H with respect to its components are continuous in a neighborhood of the solution. It should be noted that in a calculation process the component blocks may well be continuous but that there might be several discontinuities from one component block to another. This discontinuity leads to a discontinuity in the values of the derivatives of H with respect to its components. In these conditions, we are no longer in the hypotheses of the previous theorem and we are therefore not sure that the first and second derivatives of H with respect to its components are continuous.

Thus directly determining the order of convergence (quadratic or not) of the asynchronous parallel Newton method is a difficult problem and a widely open question that is outside the initial objectives of the submitted paper.

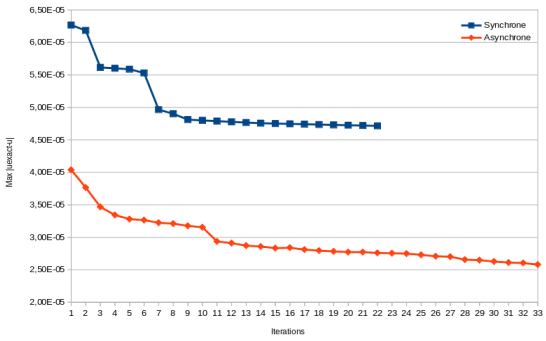
Moreover, in the general asynchronous case, it has been established in the case of the asynchronous multisplitting method that the fixed point application is a contraction, which corresponds to a linear and no longer quadratic convergence (see Proposition 3 and Corollary 1); this explains why the number of Newton iterations is not the same in synchronous or asynchronous mode. Thus, asynchronous communications between computation processes lead to the loss of the quadratic character of the Newton convergence, which is not the case in synchronous mode.

So, in conclusion, the order of the iterative process impacts the speed of convergence. This speed of convergence is quadratic in the sequential or synchronous case, and is linear in the asynchronous case in accordance with the results of Proposition 3 and Corollary 1. The character of quadratic convergence is therefore no longer retained.

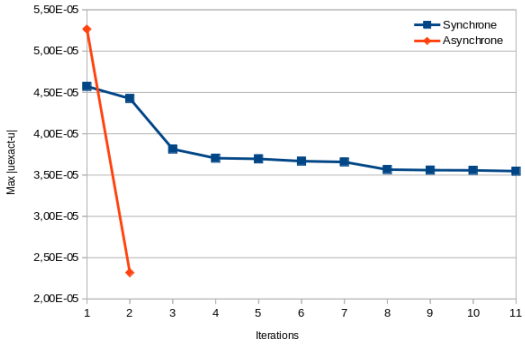
Results with 8 *cores on* 1 *cluster with* $n = 2$ *and* $q = 4$.



(a) $240^3 = 13\,824\,000$



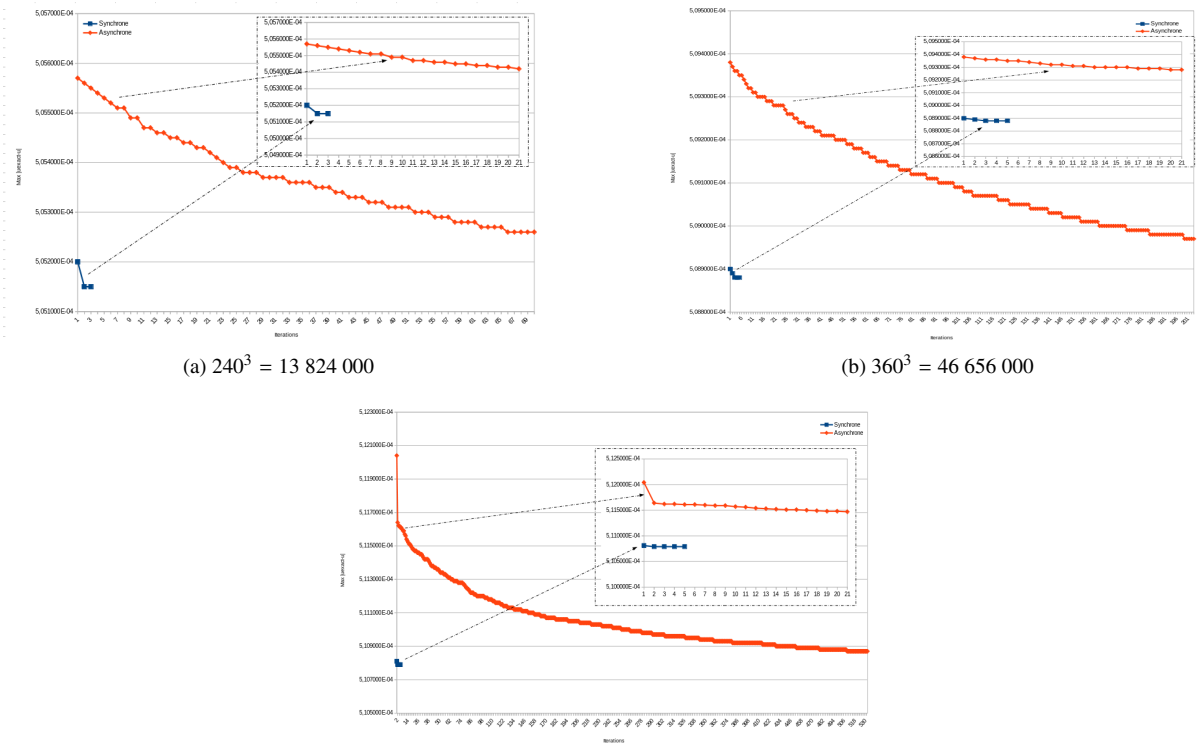
(b) $360^3 = 46\,656\,000$



(c) $480^3 = 110\,592\,000$

Figure 12: Newton Multisplitting problem : Results with 8 cores on 1 cluster

Results with 600 *cores on* 1 *cluster with* $n = 2$ *and* $q = 300$.



(a) $240^3 = 13\,824\,000$

(b) $360^3 = 46\,656\,000$

(c) $480^3 = 110\,592\,000$

Figure 13: Newton Multisplitting problem : Results with 600 cores on 1 cluster

At the experimental level, in our parallel experiments, in both synchronous and asynchronous modes, the same calculation codes were used. The only differences lie that in the synchronous mode the MPI_SEND and MPI_RECEIVE routines were used, while in the asynchronous mode the MPI_ISEND and MPI_IRECEIVE routines were used. This justifies the comparison between both versions. Otherwise the comparison of synchronous mode versus asynchronous mode would make no sense.

However, considering that the suno cluster on the Sophia site is no longer in the Grid'5000 network, we have limited our tests to the use of

- a local cluster with an 8-core Intel Xeon E5-2630 v4 @ 2.2 GHz processor and shared memory to exchange messages,
- the cluster named paravance (see table 2) of Grid'5000 composed of 600 cores where the machines are connected to each other by a network,
- two distant clusters geographically distant named paravance and ecotype (see table 2) composed of 600 cores connected by a network. There are several machines in these clusters and all the cores of each machine are used. The cores on the same machine use shared memory to exchange messages.

In each case the problem sizes corresponding to 240^3; 360^3 and 480^3 are considered and for each data set the maximum error is plotted according to the Newton iteration number (see Figures 12 to 14 where synchronous and asynchronous communications mode are considered).

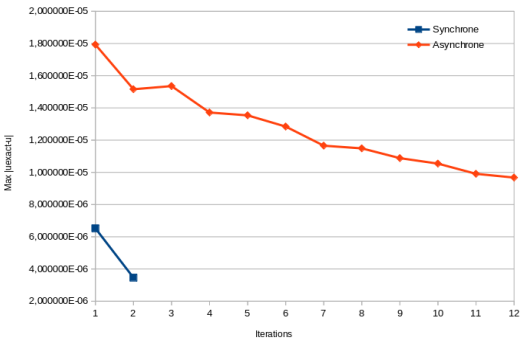
There is a different behavior when switching from one machine to another, the architecture of the machine having an influence on the result obtained. The essential elements influencing the behavior being the speed of

communication and the updates of the components. It is clear that the synchronous version converges faster than the asynchronous version, which is consistent with the heuristic explanation given above. However, the number of synchronous iterations on the local cluster is higher than on the Grid'5000 cluster or on the two geographically distant clusters of Grid'5000. Moreover, on the local cluster, except in the case 360^3, the error in asynchronous communication mode is lower than that obtained in synchronous mode. Additionally, we can see from the numerical values of the uniform error norm that these values are very close to the order of $1.0E-05$ on the local cluster and to the order of $1.0E-04$ on the Grid'5000 cluster.

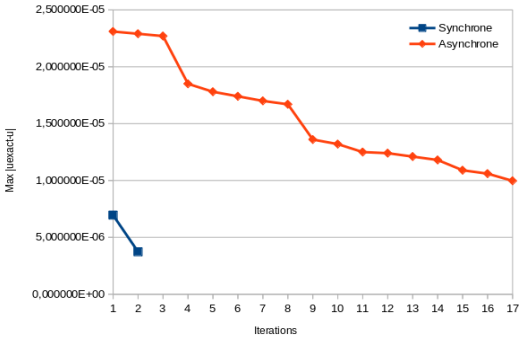
When two geographically remote clusters connected by a network are used, the numerical values of the uniform error norm are ten times higher in the asynchronous case than in the synchronous one. It is therefore normal under these conditions that convergence in asynchronous mode is slower.

Finally, in the synchronous case the decrease of the uniform norm of the error is uniform, whereas in the asynchronous case it is chaotic insofar as one notes alternatively increasing and then decreasing fluctuations of the graph. This is due to the non-deterministic mode of asynchronous communications and the limitation due to the distance between both clusters in the latter case. We note that the number of Newton iterations increases with the number of cores in the asynchronous mode. However, the restitution time is better in the asynchronous mode and this is what we are trying to do by eliminating idle time due to asynchronous mode expectations. Finally, the extra computation time in asynchronous mode improves the numerical quality of the resulting solution. It should also be noted that the asynchronous mode is not intended to accelerate convergence.

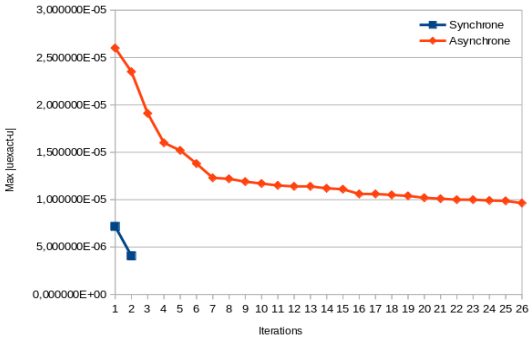
Results with 600 *cores on* 2 *clusters with* $n = 2$ *and* $q = 300$*.*



(a) $240^3 = 13\,824\,000$



(b) $360^3 = 46\,656\,000$



(c) $480^3 = 110\,592\,000$

Figure 14: Newton Multisplitting problem : Results with 600 cores on 2 clusters

References

[1] D. Chazan, W. Miranker, "Chaotic relaxation", Linear Algebra Appl., 2 , 199-222, 1969.
[2] J.-C. Miellou, " Algorithmes de relaxation chaotique à retards", RAIRO Analyse numérique, 1 , 55-82, 1975.
[3] G. Baudet, "Asynchronous iterative methods for multiprocessors", Journal Assoc. Comput. Mach., 25 , 226-244, 1978.
[4] D. Bertsekas, J. Tsitsiklis, "Parallel and Distributed Computation", Numerical Methods, Prentice Hall Englewood Cliffs N.J., 1989.
[5] A. Frommer, D. Szyld, "On asynchronous iterations", Journal of Computational and Applied Mathematics, 123 , 201-216, 2000.
[6] M. El Tarazi, "Some convergence results for asynchronous algorithms", Numerische Mathematik, 39 , 325-340, 1984.
[7] D.P. O'Leary, R.E. White, "Multi-splittings of matrices and parallel solution of linear systems", SIAM:Journal on Algebraic Discrete Methods, 6, 630 - 640, 1985.
[8] R. E. White, "Parallel algorithms for nonlinear problems", SIAM J. Alg. Discrete Meth., 7, 137 - 149, 1986.
[9] J. Arnal, V. Migallón, J. Penadés, "Parallel Newton two-stage multisplitting iterative methods for nonlinear systems", BIT Numerical Mathematics, 43, 849 - 861, 2003.
[10] Z.Z. Bai, "Asynchronous multisplitting AOR methods for a class of systems of weakly nonlinear equations", Appl. Math. Comp., 98, 49 - 59, 1999.
[11] Z. Z. Bai, D.R. Wang, "Improved comparison theorem for the nonlinear multisplitting relaxation method", Computers Math. Appl., 31-8, 23 - 30, 1996.
[12] ZZ. Bai, V. Migallón, J. Penadés, DB. Szyld, "Block and asynchronous two-stage methods for mildly nonlinear systems". Numerische Mathematik, 82(1), 1 - 20, 1999.
[13] R. Bru, V. Migallón, J. Penadés, D.B. Szyld, "Parallel, synchronous and asynchronous two-stage multisplitting methods", Electronic Transactions on Numerical Analysis, 3, 24 - 38, 1995.
[14] R. Couturier, C. Denis, F. Jézéquel, "GREMLINS: a large sparse linear solver for grid environment". Parallel Comput 34(68), 380 - 391, 2008.
[15] R. Couturier, L. Ziane Khodja, "A scalable multisplitting algorithm to solve large sparse linear systems", The Journal of Supercomputing, 69 (1), 200 - 224, 2014.
[16] A. Frommer, "Parallel nonlinear multisplitting methods", Numerische Mathematik, 56, 269 - 282, 1989.
[17] F. Jézéquel, R. Couturier, C. Denis, "Solving large sparse linear systems in a grid environment: the GREMLINS code versus the PETSc library", Journal of Supercomputing. 59 (3), 1517 - 1532, 2012.
[18] MT Jones, DB. Szyld, "Two-stage multisplitting methods with overlapping blocks", Numerical Linear Algebra with Applications, 3, 113 - 124, 1996.
[19] P. Spiteri, J.C. Miellou, D. El Baz, "Parallel asynchronous Schwarz and multisplitting methods for a non linear diffusion problem", Numerical Algorithms, 33, 461-474, 2003.
[20] D. Szyld, "Different models of parallel asynchronous iterations with overlapping block", Computational and Applied Mathematics, 17, 101-115, 1998.
[21] J.M. Bahi, J.C. Miellou, K. Rhofir, "Asynchronous multisplitting methods for nonlinear fixed point problems", Numerical Algorithms, 15, 315 - 345, 1997.
[22] Y. Saad, "Iterative methods for sparse linear systems", SIAM, 2003.
[23] J. Mossino, "Sur certaines inéquations quasi-variationnelles apparaissant en physique", CRAS Paris, 282, 187 - 190, 1976.
[24] P. Spiteri, L. Ziane-Khodja, R. Couturier, "Asynchronous parallel multi-splitting mixed methods", in P. Ivnyi, B.H.V. Topping, (Editors), "Proceedings of the Sixth International Conference on Parallel, Distributed, GPU and Cloud Computing for Engineering", Civil-Comp Press, Stirlingshire, UK, Paper 15, 2019. doi:10.4203/ccp.112.15
[25] V. Barbu, " Nonlinear semigroups and differential equations in Banach spaces", Noordhoff International Publishing, 1976.
[26] R. Glowinski, J.L. Lions, R. Tremolières, "Analyse numérique des inéquations variationnelles", DUNOD, tome 1 and 2, 1976.
[27] J.-C. Miellou, P. Spiteri, "Un critère de convergence pour des méthodes générales de point fixe", M2AN, 19, 645-669, 1985.
[28] M. Chau, A. Laouar, T. Garcia, P. Spitri. "Grid solution of problem with unilateral constraints", Numerical Algorithms, Springer-Verlag, 75 - 4, 879 - 908, 2017.
[29] J. Ortega, W. Rheinboldt, "Iterative Solution of Nonlinear Equations in Several Variables", Academic Press, New York, 1970.
[30] D.J. Evans, W. Deren, "An asynchronous parallel algorithm for solving a class of nonlinear simultaneous equations", Parallel Computing, 17, 165–180, 1991.
[31] Y. Saad, M. H. Schultz,"GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems", SIAM Journal on Scientific and Statistical Computing, 7 - 3, 856–869, 1986.
[32] C. E. Ramamonjisoa, L. Ziane Khodja, D. Laiymani, A. Giersch, R. Couturier, "Simulation of Asynchronous Iterative Algorithms Using SimGrid", In 2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS), 890-895, 2014.
[33] R. Bolze, F. Cappello, E. Caron, M. Daydé, F. Desprez, E. Jeannot, Y. Jégou, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, P. Primet, B. Quetier, O. Richard, E.-G. Talbi, I. Touche, Grid5000: A large scale and highly reconfigurable experimental grid testbed, International Journal of High Performance Computing Applications 20 (4), 481494, 2006.
[34] https://www.grid5000.fr/w/Nantes:Hardware#ecotype
[35] https://www.grid5000.fr/w/Rennes:Hardware#parapide
[36] https://www.grid5000.fr/w/Rennes:Hardware#paravance
[37] https://www.grid5000.fr/w/Sophia:Hardware#suno
[38] R.L. Burden and J.D. Faires, Numerical Analysis, BROOKS/COLE, CENGAGE Learning, Ninth edition, 2011, page 639-644

Authors Statements

T. Garcia : Conceptualization, Methodology, Development, Writing
P. Spiteri : Conceptualization, Methodology, Development, Writing
L. Ziane-Khodja : Conceptualization, Methodology, Development, Writing
R. Couturier : Conceptualization, Methodology, Development, Writing

Declaration of interests

■ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: