Coupling parallel asynchronous multisplitting methods with Krylov methods to solve pseudo-linear evolution 3D problems

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 Maréchal Juin, BP 527, 90016 Belfort cedex, France

Abstract

The present study deals with pseudo - linear problems solving using parallel asynchronous multisplitting methods combined with Krylov methods. With appropriate and realistic assumptions, the behavior of such parallel iterative algorithms will be analyzed by partial ordering techniques in relation with the discrete maximum principle. Applications to discretized boundary value problems are presented, the implementation of the algorithms is described and parallel experiments are analyzed.

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

1. Introduction and motivation

Advances in the modeling of physical, chemical, biological and economic phenomena lead to the consideration of highly nonlinear mathematical problems. This is particularly true when the mathematical model is a boundary value problem. Except in some rare cases, it is practically difficult if not impossible to solve these partial differential equations by analytical means. In this case, numerical methods are used, which requires the discretization of the problem to be solved. However with such numerical methods other difficulties appear. Indeed, the implementation of these methods on a computer requires the temporal as well as the spatial discretization of the equations to be solved; due to the numerical instabilities which propagate and which can lead to unstable temporal discretization schemes, it is usually more interesting to use implicit or semi-implicit time marching schemes. The spatial and temporal discretizations by implicit or semi-implicit schemes, when combined, lead to the solution of large scale nonlinear algebraic systems; in this case, due to the large amount of computation required, parallel numerical algorithms seem well adapted to an efficient numerical solution of this class of problems. The design of such parallel numerical methods is a real challenge and an important literature exist in this field.

Usually, due to the fact that the matrices are sparse, in order to limit the effect of rounding errors, it is preferred to solve these large problems by iterative methods. Parallel iterative resolution of these very large problems requires the partitioning of these problems into interconnected sub-problems, each sub-problem being solved by one processor or by a block of processors taking into account the values computed by the other processors. Such a coupling will generate synchronizations between the parallel processes, which will generate phases of inactivity due to the expectations of the results emitted by the other processors. To overcome this drawback, asynchronous parallel methods have been developed (see [6, 7, 10, 18]). To analyse the behavior of these iterative methods, many studies have been developed using contraction methods (see [6, 10, 18]); thus, if the fixed point application associated with the problem to be solved is contracting, the asynchronous parallel methods will converge towards the unique solution of the discretized problem.

*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)

However, another type of analysis of the behavior of these methods was also developed (see [3, 13, 19, 20, 21, 29]). In this type of method analysis, the authors considered a monotonous convergence towards the solution, either starting from an initial condition so that the iterates produced converge towards the solution of the problem to be solved in decreasing order, or starting the iterative process so that the produced iterates converge towards the solution in increasing order. This type of study is related to the use of the discrete maximum principle and such a monotonous behavior of the iterates is obtained when the problem to be solved is an M-function in the sense of Rheinboldt [26], i.e. a monotonous inverse application and off-diagonal decreasing. Such an application is for example obtained when solving pseudo-linear problems, obtained by the perturbation of an affine application $AU-G$, where A is an M-matrix [24] perturbed by a diagonal increasing operator or more generally monotonous diagonal operator; note that the property of A is usually verified when a linear boundary value problem is discretized.

Among the newly developed parallel methods, we can mention the multisplitting methods which have the advantage of unifying the presentation of parallel subdomains methods with or without overlapping with both synchronous and asynchronous communications. These methods were introduced by O'Leary and White [23, 31] and have also been extensively studied due to their generality ([1, 2, 4, 5, 9, 11, 12, 14, 15, 16, 30]). The study of the behavior of these methods has been carried out mainly by contraction methods.

It can be noted that the analysis of multisplitting methods by partial ordering methods has not been studied. So the aim of the present work is, on the one hand, to analyze the behavior of multisplitting methods using partial order techniques to solve pseudo-linear evolutionary problems in relation to various applications and, on the other hand, to combine an external parallel iteration allowing to couple the resolution of the interconnected sub-problems to be solved with an efficient iterative method constituted in this study by the use of a Krylov method ([27, 28]). The external iteration is in our study case either constituted by a synchronous or an asynchronous iterative method. The efficiency of both approaches, synchronous or asynchronous will be compared experimentally on a Grid Computing. Moreover, taking into account the evolutionary nature of the problems to be solved, it will be necessary during the implementation of the algorithms to implement a synchronization barrier in order to respect the implicit algorithm used to solve the considered problems.

This paper is organized as follows. Section 2 is devoted to present on the one hand the mathematical background used in the presented study and on the other hand the modelling of parallel asynchronous algorithms. Section 3 presents the numerical aspects used for the parallel solution of stationary pseudo-linear problems, in particular the discretization techniques, the choice of the initial data of the iterative process and of Newton's method in relation to the use of the discrete maximum principle. Section 4 presents the multisplitting method and the analysis of its behavior by partial order techniques. Section 5 presents various types of pseudo-linear evolutionary problems. The next section describes the implementation of the studied methods and the last section presents the results of parallel synchronous and asynchronous experiments. Finally, the conclusion offers a synthesis of the presented study.

2. Mathematical background

2.1. Notation and definitions

Let $n \in \mathbb{N}$ and consider the n-dimensional space $\mathbb{R}^n$ denoted also in the sequel by E; consider the following decomposition of E in a finite product of α subspaces E_i, such that $E = \prod_{i=1}^{\alpha} E_i$, where α is a positive integer, $E_i = R^{n_i}$, and $\sum_{i=1}^{\alpha} n_i = n$; in the sequel, let $U \in E$ and consider the following block-decomposition of U defined as follows

$$U = \{U_1, \ldots, U_i, \ldots, U_\alpha\} \in \prod_{i=1}^{\alpha} E_i,\ U_i \in E_i \text{ for } i = 1, \ldots, \alpha; \tag{1}$$

moreover for $i \in \{1, \ldots, \alpha\}$, each subspace E_i is endowed with the natural partial ordering (i.e. component by component) denoted by $U_i \underset{i}{\leq} V_i$ for $U_i, V_i \in E_i$. Let us also denote by $\leq$ the partial ordering considered for

$E = \prod_{i=1}^{\alpha} E_i$ and then defined by:

$$\text{for } U \text{ and } V \text{ two vectors of } E\text{; then } U \leq V \text{ if } U_i \underset{i}{\leq} V_i, \forall i \in \{1, \ldots, \alpha\}.$$

Moreover, the notion of order interval, defined below, will be used thereafter

Definition 1. *Let $V_i, U_i \in E_i$ such that $V_i \underset{i}{\leq} U_i$; then the order interval $\langle V_i, U_i \rangle_i$ is defined as $\{W_i \in E_i \mid V_i \underset{i}{\leq} W_i \underset{i}{\leq} U_i\}$.*

The definition is similar for $\langle U, V \rangle$, $V, U \in E$, i.e. $\langle U, V \rangle = \{W \in E \mid V \leq W \leq U\}$.

Let $\mathcal{A}$ a continuous mapping from $E = \mathbb{R}^n$ onto itself. Recall the following definition (see [24] and [26])

Definition 2. *The mapping $\mathcal{A}$ is an M-function if $\mathcal{A}$ is inverse monotone, i.e.*

$$\mathcal{A}(U) \leq \mathcal{A}(V) \textit{ implies } U \leq V, \textit{ for any } U, V \in \mathbb{R}^n \tag{2}$$

and furthermore $\mathcal{A}$ is off-diagonally antitone, i.e. for any $U \in \mathbb{R}^n$, the functions defined as follows

$$\begin{cases} \mathcal{A}_{lk} : \{\tau \in \mathbb{R} | U + \tau e_k \in \mathbb{R}^n\} \to \mathbb{R}, \\ \mathcal{A}_{lk}(\tau) = \mathcal{A}_l(U + \tau e_k), l \neq k, l, k = 1, \ldots, n, \end{cases} \tag{3}$$

where $\mathcal{A}_l$ denotes the block number l of the M-function $\mathcal{A}$, are monotone decreasing, $e_k \in \mathbb{R}^n$, $k = 1, \ldots, n$, denoting the unit canonical basis vectors.

Lemma 1. *If $\mathcal{A}$ is an M-matrix $A = (a_{l,k})$, i.e. $\mathcal{A}(U) = AU$, then $\mathcal{A}$ is a linear M-function.*

Proof. Indeed, in this case, for $l \neq k$, let us consider a point decomposition of the operator, $\mathcal{A}_l(U + \tau e_k) = \sum_{j=1}^{\alpha} a_{l,j} u_j + a_{l,k} \tau$; let us assume that τ_1 and τ_2 are two real numbers such that $\tau_1 \leq \tau_2$; then, since $a_{l,k} \leq 0$,

$$\mathcal{A}_l(U + \tau_2 e_k) = \sum_{j=1}^{\alpha} a_{l,j} u_j + a_{l,k} \tau_2 \leq \sum_{j=1}^{\alpha} a_{l,j} u_j + a_{l,k} \tau_1 = \mathcal{A}_l(U + \tau_1 e_k),$$

and $\mathcal{A}$ is off-diagonally antitone; moreover A being an M-matrix, then A^{-1} is a nonnegative matrix and then $A^{-1}U \leq A^{-1}V$ implies $U \leq V$, so that $\mathcal{A}$ is inverse monotone, which achieves the proof. ■

Remark 1. *Note that such a result also follows to the fact that the mapping $\tau \to \mathcal{A}_l(U + \tau e_k)$ admits a negative derivative with respect to τ equal to $a_{l,k}$.*

Remark 2. *More generally, an affine mapping $U \to AU - G$ is a linear M-function if and only if A is an M-matrix (see [24] on page 468 and [26] Lemma 2.9 on page 278).*

For supplementary assumptions and properties about the notion of M-function, we refer to [26]). Assume also that

$$\mathcal{A} \text{ is a continuous surjective M-function.} \tag{4}$$

Let $\mathcal{A}$ be a continuous surjective M-function and let us consider the solution of the following system of equations

$$\mathcal{A}(U) = 0; \tag{5}$$

Under the above assumptions, problem (5) has a unique solution U^* (see [24] and [26]). According to the block-decomposition of U let us also consider the corresponding block-decomposition of $\mathcal{A}$

$$\mathcal{A}(U) = \{\mathcal{A}_1(U), \ldots, \mathcal{A}_i(U), \ldots, \mathcal{A}_\alpha(U)\} \in \prod_{i=1}^{\alpha} E_i.$$

For all $i \in \{1, \ldots, \alpha\}$, we introduce the following mapping from E_i onto itself

$$U_i \to \mathcal{A}_i(U_i; V) = \mathcal{A}_i(V_1, \ldots, V_{i-1}, U_i, V_{i+1}, \ldots, V_\alpha).$$

Since $\mathcal{A}$ is a continuous surjective M-function it follows from Theorem 3.5 of [26] that for all $i \in \{1, \ldots, \alpha\}$ and all $V \in E$, the mapping $U_i \longrightarrow \mathcal{A}_i(U_i; V)$ is a continuous surjective M-function. Moreover, for all $i \in \{1, \ldots, \alpha\}$ and $V \in E$, the system

$$\mathcal{A}_i(U_i; V) = 0, \tag{6}$$

has a unique solution.

So, a fixed point mapping $F : E \to E$, associated with problem (5) can be defined such that

$$U = F(V) \text{ for all } V \in E; \tag{7}$$

where $V \mapsto F(V)$ applies from $D(F) \subset E$ into $D(F)$; taking into account the $\alpha-$decomposition of E,

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

it can be stated according to [26] that the mapping F is well defined and monotone increasing on E and then for all $U, V \in E$ such that $U \leq V$, $F(U) \leq F(V)$. In order to solve problem (5), we will consider general fixed point iterative methods where the initial guess V^0 for the fixed point iterations satisfies the following definition

Definition 3. *A vector $V^0 \in R^n$ is an $\mathcal{A}$-supersolution (respectively $\mathcal{A}$-subsolution) if $\mathcal{A}(V^0) \geq 0$ (respectively $\mathcal{A}(V^0) \leq 0$).*

2.2. *Parallel asynchronous algorithms and monotone convergence*

In the sequel, we consider the general following fixed point problem associated to problem (5)

$$\begin{cases} \text{Find } U^* \in E \text{ such that} \\ U^* = F(U^*). \end{cases} \tag{8}$$

In order to solve the fixed point problem (8), let us consider now the parallel asynchronous iterations defined as follows : let $U^0 \in E$ be a given $\mathcal{A}$-supersolution, i.e. $\mathcal{A}(U^0) \geq 0$, (or a $\mathcal{A}$-subsolution , i.e. $\mathcal{A}(U^0) \leq 0$;) then for all $p \in \mathbb{N}$, U^{p+1} is recursively defined by

$$U_i^{p+1} = \begin{cases} F_i(U_1^{\rho_1(p)}, \ldots, U_j^{\rho_j(p)}, \ldots, U_\alpha^{\rho_\alpha(p)}) \text{ if } i \in s(p) \\ U_i^p \text{ if } i \notin s(p) \end{cases} \tag{9}$$

where

$$\begin{cases} \forall p \in \mathbb{N}, s(p) \subset \{1, \ldots, \alpha\} \text{ and } s(p) \neq \emptyset \\ \forall i \in \{1, \ldots, \alpha\}, \{p \mid i \in s(p)\} \text{ is countable} \end{cases} \tag{10}$$

and $\forall j \in \{1, \ldots, \alpha\}$,

$$\begin{cases} \forall p \in \mathbb{N}, \rho_j(p) \in \mathbb{N}, 0 \leq \rho_j(p) \leq p \text{ and } \rho_j(p) = p \text{ if } j \in s(p) \\ \lim\limits_{p \to \infty} \rho_j(p) = +\infty. \end{cases} \tag{11}$$

The previous asynchronous iterative scheme models computations that are carried out in parallel without order nor synchronization and describe a subdomain method without overlapping. Particularly, these computations enable one to consider distributed computations whereby processors go at their own pace according to their intrinsic characteristics and computational load. The parallelism between the processors is well described by the set $s(p)$ which contains the number of components relaxed by each processor on a parallel way at each relaxation while the use of delayed components in (9) enables one to model nondeterministic behavior and does not imply inefficiency of the considered distributed scheme of computation. Note that, theoretically, each component of the vector must be relaxed an infinity of time. The choice of the relaxed components may be guided by any criterion, and, in particular, a natural criterion is to pick-up the most recently available values of the components computed by the processors. Note also that in the model of asynchronous parallel iterations the information communicated between the processors was no longer bounded but could have unbounded communication delays, which made it possible to take into account possible temporary failures of the multiprocessors. It can be noticed that when many synchronizations are necessary to solve the target problem, then parallel asynchronous algorithms suppress synchronizations. Thus, in the asynchronous mode, a better interaction is obtained between the communications of the processors and the computations they carry out. This flexibility of communication offers the possibility to acquire the updated values of the components by the other processors. Moreover, once the communications are carried out by a processor, the latter can continue its own calculations, which is not the case in synchronous mode. Therefore a better overlap between computations and communications is achieved, which reduces processor idle time due to synchronizations.

Remark 3. *Such asynchronous iterations describe various classes of parallel algorithms, such as parallel synchronous iterations if* $\forall j \in \{1, \ldots, \alpha\}$, $\forall p \in \mathbb{N}, \rho_j(p) = p$. *In the synchronous context, for particular choice of* $s(p)$, *then (9)-(11) describe classical sequential relaxation algorithms; among them, the Jacobi method if* $\forall p \in \mathbb{N}, s(p) = \{1, \ldots, \alpha\}$ *and the Gauss-Seidel method if* $\forall p \in \mathbb{N}, s(p) = \{1 + p \bmod \alpha\}$ *(see[18]).*

Let us consider now the parallel asynchronous fixed point iterations $\{U^p\}_{p \in N}$ defined by (9) - (11) and recall now an important result (see [21]).

Proposition 1. *Let* $\mathcal{A}$ *be a continuous surjective M-function, F the fixed point mapping associated with* $\mathcal{A}$ *defined by (7),* $U^0 \in E$ *an* $\mathcal{A}$*-supersolution (respectively* $V^0 \in E$ *an* $\mathcal{A}$*-subsolution). Then, the asynchronous iteration* $\{U^p\}_{p \in \mathbb{N}}$ *(respectively* $\{V^p\}_{p \in \mathbb{N}}$*) given by (9) - (11) is well defined and satisfies*

$$U^p \downarrow U^\star, p \to \infty, \tag{12}$$

(respectively

$$V^p \uparrow U^\star, p \to \infty), \tag{13}$$

where (12) means that $\lim_{p \to \infty} U^p = U^\star$ *and* $U^\star \leq \cdots \leq U^{p+1} \leq U^p \leq \cdots \leq U^0$ *(respectively (13) means that* $\lim_{p \to \infty} V^p = U^\star$ *and* $V^p \geq \cdots \geq V^{p+1} \geq V^p \geq \cdots \geq V^0$*).*

The numerical termination of parallel asynchronous iterations is a very challenging and quite difficult to implement. Nevertheless the result of Proposition 1 allows to obtain a suitable numerical stopping criterion, when two sequences of iterate vectors associated respectively to an $\mathcal{A}$-supersolution and to an $\mathcal{A}$-subsolution are considered.

Corollary 1. *If we consider the parallel asynchronous fixed point iterations (9) - (11) with on one hand the initial guess* U^0, *an* $\mathcal{A}$*-supersolution, and on the other hand* V^0, *an* $\mathcal{A}$*-subsolution, then, since* $\mathcal{A}$ *is continuous, we have*

$$V^0 \leq V^1 \leq \cdots \leq V^p \leq \cdots \leq V^\star = U^\star \leq \cdots \leq U^q \leq \cdots \leq U^1 \leq U^0,$$

where $V^\star$ *and* $U^\star$ *are the same fixed point of (8); if now* ϵ *is a convenient threshold for the termination of the iterative process, then the following stopping criterion can be implemented*

$$U^q - V^p \leq \epsilon, \forall q, p \in \mathbb{N}.$$

3. Parallel solution of pseudo-linear problems

Among the nonlinear boundary value problems, the class of pseudo-linear problems plays an important role in several applications. So, let us consider the following nonlinear convection - diffusion problem

$$\begin{cases} -\Delta u + c^t \nabla u + \phi(u) + d \cdot u = g, \text{ everywhere in } \Omega, d \geq 0, \\ \qquad u = 0, \text{ everywhere in } \partial\Omega, \end{cases} \tag{14}$$

where Ω is an open domain included in R^3, $\partial\Omega$ is the boundary of Ω, $g \in L^2(\Omega)$ is a given function and ϕ is a diagonal monotone increasing, convex and continuously differentiable nonlinear operator.

3.1. Discretisation

For nonlinear pseudo-problem (14) a spatial discretization by finite difference method with an uniform spatial discretization step-size h is considered; discretization by finite volume methods or some class of finite element methods can also be considered. More precisely we consider the discretization of the problem (14) using the five-point difference scheme in the two-dimensional case and the seven-point difference scheme in the three-dimensional case for the Laplacian and for the first derivative only one-sided backward or forward difference equations, according to the sign of the first derivative arising in problem (14); for example for the first derivative with respect to x the following scheme is considered

$$c_1\frac{\partial u(x,y,z)}{\partial x} \approx \begin{cases} c_1.\frac{u(x,y,z)-u(x-h,y,z)}{h} + O(h), & \text{if } c_1 > 0, \\ c_1.\frac{u(x+h,y,z)-u(x,y,z)}{h} + O(h), & \text{if } c_1 < 0. \end{cases}$$

and accordingly for the first derivative with respect to y and z; the first scheme, associated to the case where $c_1 > 0$ corresponds to a backward scheme while the other associated to the case where $c_1 < 0$ defines a forward scheme. Such discretisation of problem (14) leads to solve the following algebraic system

$$AU + \Phi(U) = G, \tag{15}$$

where $A \in \mathcal{L}(\mathbb{R}^n)$ is the discretisation matrix, $U \in \mathbb{R}^n$, $G \in \mathbb{R}^n$, and Φ and G result from the discretisation of ϕ and g. Note that, due to the fact that the discretisation matrix is irreducible diagonal dominant, then

$$A \text{ is an M-matrix}, \tag{16}$$

and

$$\Phi \text{ is a diagonal continuous increasing operator.} \tag{17}$$

For the analysis of the behavior of the parallel iterative method presented in subsection 2.2, according to the assumptions of Proposition 1, the discretized problem (14) has interesting properties. Then, the following result can be stated

Lemma 2. *Let A be an M-matrix, $G \in \mathbb{R}^n$ and $U \rightarrow \Phi(U)$ be a continuous diagonal monotone maximal mapping. Then, the following mapping*

$$\mathcal{A}(U) = AU - G + \Phi(U) \tag{18}$$

is an M-function

Proof. Indeed, in a similar way than the one used in the proof of Lemma 1, it can be verified that $\mathcal{A}(U)$ is off-diagonally antitone; moreover, applying the Theorem 3.4 of [26] on page 286, since $\mathcal{A}(U)$ is obviously continuous and off-diagonally antitone, then $\mathcal{A}(U)$ is a surjective M-function. ■

Remark 4. *For another proof see also Theorem 4.5 of [26] on page 294 and Theorem 13.5.6 of [24] on page 467.*

3.2. Link with the Newton method

3.2.1. Determination of $\mathcal{A}-$supersolution and $\mathcal{A}-$subsolution

Let us indicate below a process of choice of $\mathcal{A}$-supersolution and a $\mathcal{A}$-subsolution.

Consider the discretized problem (15); assume that A verifies assumption (16), i.e. A is an M-matrix and that $U \rightarrow \Phi(U)$ satisfies assumption (17), i.e. $\Phi(U)$ is a diagonal monotone operator. Consider the block decomposition of problem (15)

$$\sum_{k=1}^{\alpha} A_{l,k}U_k + \Phi_l(U_l) = \mathcal{B}_l, \forall l \in \{1, \ldots, \alpha\}, \tag{19}$$

and the following application $U \mapsto \mathcal{A}(U)$ defined by (18) where, due to the considered assumptions and according to the result of Lemma 2 $U \rightarrow \mathcal{A}(U)$ is an M-function. Let $\mathcal{A}_i(U_i;\, W)$ be defined by

$$\mathcal{A}_i(U_i;W) = A_{i,i} \cdot U_i + \Phi_i(U_i) + \sum_{j \neq i}^{\alpha} A_{i,j} W_j - \mathcal{B}_i,\ \forall i \in \{1, \ldots, \alpha\}. \tag{20}$$

A fixed point mapping F can be implicitly defined from $\mathbb{R}^n$ in $\mathbb{R}^n$ associated to the sub-problem (19). Let W, U^0 and V^0 three vectors of $\mathbb{R}^n$ such that

$$\mathcal{A}_i(W_i;W) \geq 0,\ U_i^0 = W_i \text{ and } V_i^0 = F_i(W),\ \forall i \in \{1, \ldots, \alpha\}. \tag{21}$$

For $i \in \{1, \ldots, \alpha\}$, let $\theta_i = \min(\mathcal{A}_i(W_i;W), 0)$ and $\zeta_i = \max(\mathcal{A}_i(W_i;W), 0)$. Due to the first inequality of (21) note that $\theta_i = 0$; then $\zeta_i = \mathcal{A}_i W_i;W)$. Since V_i^0 verify

$$A_{i,i} \cdot V_i^0 + \Phi_i(V_i^0) = \mathcal{B}_i - \sum_{j \neq i}^{\alpha} A_{i,j} W_j, \forall i \in \{1, \ldots, \alpha\}$$

and according to the first inequality of (21)

$$A_{i,i} \cdot W_i + \Phi_i(W_i) \geq \mathcal{B}_i - \sum_{j \neq i}^{\alpha} A_{i,j} W_j, \forall i \in \{1, \ldots, \alpha\},$$

then

$$A_{i,i} \cdot W_i + \Phi_i(W_i) \geq A_{i,i} \cdot V_i^0 + \Phi_i(V_i^0), \forall i \in \{1, \ldots, \alpha\},$$

which implies $W_i \geq V_i^0, \forall i \in \{1, \ldots, \alpha\},$ since due to the fact that A is an M-matrix, then $A_{i,i}$, $\forall i \in \{1, \ldots, \alpha\}$ are also M-matrices and the mapping $W_i \rightarrow A_{i,i} \cdot W_i + \Phi_i(W_i), \forall i \in \{1, \ldots, \alpha\},$ is an M-function. In addition

$$\mathcal{A}_i(W_i;W) \geq 0 = \mathcal{A}_i(U_i^\star;U^\star), \forall i \in \{1, \ldots, \alpha\},$$

which finally implies $W_i \equiv U_i^0 \geq U_i^\star, \forall i \in \{1, \ldots, \alpha\}$.

Let now $V^0 = \mathcal{A}^{-1}(\theta)$ with $\theta = 0$ or equivalently $\mathcal{A}(V^0) = 0$ and $U^0 \equiv W = \mathcal{A}^{-1}(\zeta)$ or equivalently $\mathcal{A}(U^0) = \mathcal{A}(W) \geq 0$. Note also that $V^0 \equiv U^\star$. Conversely in the opposite case where

$$\mathcal{A}_i(W_i;W) \leq 0 \tag{22}$$

then $\theta_i = \min(\mathcal{A}_i(W_i;W)$, and $\zeta_i = 0$; thus, in the same way $\mathcal{A}(V^0) = \mathcal{A}(W) \leq 0$ and $U^0 \equiv U^\star$. So, in both cases, we have

$$\mathcal{A}(V^0) \leq \mathcal{A}(U^\star) = 0 \leq \mathcal{A}(U^0), \tag{23}$$

and in addition, based on the above assumptions, the application $\mathcal{A}_i$ is an M-function on the order interval $\langle V_i^0, U_i^0 \rangle_i, \forall i \in \{1, \ldots, \alpha\}$. So, starting with any vector W verifying the first inequality of (21) or (22), we get a $\mathcal{A}$-supersolution and/or a $\mathcal{A}$-subsolution.

Remark 5. *In practical implementations, this type of initialization is made possible since the matrix A is an M-matrix. Indeed, we can use the following property of M-matrices : if A is an M-matrix, then there exists a vector $e > 0$, $e \in \mathbb{R}^n$, such that $\tau = Ae > 0$. Then consider now the vector $\bar{\tau} = (1 + \mu)e$ where μ is a positive number chosen such that*

$$(1 + \mu)Ae - G = (1 + \mu)\tau - G > 0.$$

If $U \rightarrow \Phi(U)$ is a positive vector, then the previous value of μ allows to define the positive vector $W = (1 + \mu) \cdot e$. Otherwise, if $\Phi(U)$ is a negative vector, then, since the mapping $U \rightarrow \Phi(U)$ is increasing, we can then further increase the value of μ so that $\mathcal{A}(W) \equiv \mathcal{A}((1 + \mu) \cdot e) = AW + \Phi(W) - G > 0$. Note also that instead of choosing the vector $\bar{\tau} = (1 + \mu)e$ the vector $\mu \cdot e$ with $\mu > 1$ sufficiently large can be chosen.

3.2.2. The Newton method

We introduce the block-diagonal matrix $C(W)$, derived from Newton's method, with diagonal blocks $C_{i,i}(W)$ given by

$$C_{i,i}(W) = A_{ii} + \Phi'_i(W).$$

Obviously, since the matrix A is an M-matrix then A_{ii} is also an M-matrix; since Φ_i is increasing, then its derivative is positive and the matrix $C_{i,i}(W)$ is an M-matrix. Let $U_i^0 = W_i$, and U_i^p is the p-th iteration of the following algorithm:

$$U^{p+1} = U^p - C^{-1}(U^{\rho'(p)}).\mathcal{A}(U^p), k = 0, 1, \ldots, \tag{24}$$

where $0 \leq \rho'(p) \leq p$, according to (11). Then the following result is obtained

Proposition 2. *Assume that the assumptions (16) and (17) hold. Then, the sequence $\{U^p\}_{p\in\mathbb{N}}$ defined by (24) satisfies $U^p \downarrow U^\star$, $p \to \infty$, where U^* is the solution of problem (15).*

Proof. It follows from the definition of U^0 that $\mathcal{A}(U^0) \geq 0$, since $\mathcal{A}(W) \geq 0$ and $U^0 = W$. Moreover $C^{-1}(U^0) \geq 0$ since $C(U^0)$ is an M-matrix. Then

$$U^1 - U^0 = -C^{-1}(U^0)\mathcal{A}(U^0) \leq 0,$$

and then $U^1 \leq U^0$. Moreover, since $U \to \Phi(U)$ is increasing, then the gradient of Φ is monotone, so that Φ is convex. It follows from the convexity of Φ (see [24] on page 448) that

$$\mathcal{A}(U^0) - \mathcal{A}(U^\star) \leq C(U^0)(U^0 - U^\star).$$

Thus if $U^\star$ is the solution of the problem $\mathcal{A}(U^\star) = 0$ then

$$U^\star = U^\star - C^{-1}(U^0)\mathcal{A}(U^\star) = U^1 - (U^0 - U^\star) + C^{-1}(U^0)(\mathcal{A}(U^0) - \mathcal{A}(U^\star)).$$

and

$$U^1 - (U^0 - U^\star) + C^{-1}(U^0)(\mathcal{A}(U^0) - \mathcal{A}(U^\star)) \leq U^1 - (I - C^{-1}(U^0).C(U^0))(U^0 - \mathcal{A}) = U^1.$$

Thus

$$U^\star \leq U^1.$$

Moreover it follows from the convexity of Φ that

$$0 = \mathcal{A}(U^0) + C(U^0)(U^1 - U^0) \leq \mathcal{A}(U^1).$$

Therefore by induction it can be analogously proven that that

$$U^\star \leq \ \ldots\ \leq U^p \leq U^{p-1} \leq \ \ldots\ \leq U^0 \text{ and}$$

$$\mathcal{A}(U^p) \geq 0.$$

■

Remark 6. *It follows from (17) that the mapping $\mathcal{A}(W)$ given by (18) is continuous.*

4. The multisplitting method

Let us consider now the solution of the discretized problem (15) when assumptions (16) and (17) are verified. Let the following regular splittings of matrix A

$$A = M^l - N^l, \quad l = 1, \ldots, m, m \in \mathbb{N} \tag{25}$$

where $(M^l)^{-1} \geq 0$ and $N^l \geq 0$. Let $F^l : R^n \to R^n$, $l = 1, \ldots, m$ be m fixed point mappings associated implicitly with problem (15), respectively, and defined by:

$$U = F^l(V) \text{ such that } M^l U = N^l V - \Phi(V), l = 1, \ldots, m. \tag{26}$$

A formal multisplitting associated with problem (15) is defined by the collection of fixed point problems (see [2])

$$U^* - F^l(U^*) = 0,\ l = 1, ..., m. \tag{27}$$

Let now $\mathcal{E} = (R^n)^m$ and consider the following block-decomposition of $\mathcal{E}$

$$\mathcal{E} = \prod_{l=1}^{m} E_l,$$

where $E_l = R^n$. Each subspace E_l is always endowed with the natural (or componentwise) partial ordering considered in subsection 2.1. Let $\tilde{U} \in \mathcal{E}$. The following block-decomposition of $\tilde{U}$ can be considered

$$\tilde{U} = \{U^1, \ldots, U^l, \ldots, U^m\} \in \prod_{l=1}^{m} E_l.$$

Definition 4. *The extended fixed point mapping $\tilde{T} : \mathcal{E} \to \mathcal{E}$ associated with the formal multisplitting is given as follows*

$$\tilde{T}(\tilde{V}) = \tilde{U},\ \textit{such that } U^l = F^l(V^l) \textit{ with } V^l = \sum_{k=1}^{m} D_{lk} V^k, l = 1, ..., m,$$

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

$$\sum_{k=1}^{m} D_{lk} = I_l,$$

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

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

Note that considerable saving in computational work may be possible by using such a method, since a component of V^k does not need to 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.

Note that for a particular choice of the weighting matrices D_{lk}, various iterative methods can be obtained and particularly, on the one hand, a subdomain method without overlapping and, on the other hand, the classical Schwarz alternating method (see [2]). According to [2] the block - Jacobi method corresponds to the following choice of $\mathcal{M}^l$

$$\mathcal{M}^l = diag(I_1, \ldots, I_{l-1}, A_{l,l}, I_{l+1}, \ldots, I_n)\ , \tag{28}$$

and to the choice of $D_{lk} \equiv \bar{D}_l$ given by

$$\bar{D}_l = diag(0, \ldots, 0, I_l, 0, \ldots, 0), \tag{29}$$

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 $\bar{D}_l$ have positive entries smaller than one. The reader is referred to reference [23] and to other various references for other choices of weighting diagonal matrices and splittings for the definition of various multisplitting methods.

Consider the problem (15) with assumptions (16) and (17). Let the following block-decomposition of the mapping $\tilde{T}$

$$\tilde{T}(\tilde{V}) = \{\tilde{T}^1(\tilde{V}), \ldots, \tilde{T}^l(\tilde{V}), \ldots, \tilde{T}^m(\tilde{V})\} \in \prod_{l=1}^{m} E_l.$$

The extended fixed point mapping $\tilde{T}$ is associated with the following extended nonlinear problem

$$a^e(\tilde{U}^*) = 0, \tag{30}$$

where the mapping $a^e : \mathcal{E} \to \mathcal{E}$ is given by

$$a^e(\tilde{U}) = A^e\tilde{U} + \Phi^e(\tilde{U}) - G^e,$$

the mapping $\Phi^e : \mathcal{E} \to \mathcal{E}$ being the extended monotone perturbation operator of Φ and for all $l \in \{1, \dots, m\}$

$$A^{e,l}\tilde{U} = M^l U^l - N^l \sum_{k=1}^{m} D_{lk} V^k. \tag{31}$$

In the sequel $a^e_l(V^1, \dots, V^{l-1}, U^l, V^{l+1}, \dots, V^m) = A^{e,l}\tilde{U} + \Phi^e_l(\tilde{U}) - G^e_l, l = 1, \dots, m$, will also be denoted by $a^e_l(U^l\,;\,V)$.

Proposition 3. *Let the above assumptions (16) and (17) hold. The mapping a associated to problem (15) is a continuous surjective M-function.*

Proof. The proof is similar to the one considered in Lemma 2. Indeed, in a similar way, it can be verified that $a^e(\tilde{U})$ is off-diagonally antitone since $a^e_l(U^l\,;\,V)$ is also off-diagonally antitone; moreover, applying the Theorem 3.4 of [26] on page 286, since, for the same reason, $a^e(\tilde{U})$ is obviously continuous and off-diagonally antitone, then $a^e(\tilde{U})$ is a surjective M-function. Note also that the continuity and surjectivity of $a^e(.)$ follows from the continuity and maximal monotonicity of $\Phi^e(.)$. ■

Then it follows from Proposition 3 that we are in the theoretical framework of the study previously developed. Thus, the monotone convergence of asynchronous multisplitting iterations can be derived. So, for problem (15), it follows from Proposition 3 that

$$\begin{cases} \text{For all } l \in \{1, \dots, m\} \text{ and all } \tilde{U} \in \mathcal{E}, \text{ the mapping: } \quad U_l \to a^e_l(U_l, V), \\ \qquad \text{is a continuous surjective } M\text{-function from } E_l \text{ onto } E_l; \end{cases}$$

for more details the reader is referred to theorem 3.5 of [26]). Moreover it follows from Proposition 3 that

$$\begin{cases} \text{for all } l \in \{1, \dots, m\} \text{ and } \tilde{U} \in \mathcal{E}, \text{ the problem: } \quad a^e_l(U_l\,;\,V) = 0, \\ \qquad\qquad \text{has a unique solution } U_l. \end{cases}$$

It also follows from the above assumptions that $\tilde{T}$ is isotone on $\mathcal{E}$ (see [20]). Then the parallel asynchronous multisplitting method associated to the Newton method for problem (15), with an initial guess constituted by an $\mathcal{A}$−supersolution or an $\mathcal{A}$−subsolution converge monotonically to the solution of the discretized problem (15).

Remark 7. *Interested readers are referred to [21] for the parallel asynchronous Schwarz alternating method with flexible communications between the processors, corresponding to a more general model of parallel asynchronous iterations. In fact the Schwarz alternating method is a particular class of multisplitting methods. More generally, it is possible to consider the parallel synchronous or asynchronous multisplitting methods with flexible communications, corresponding to a more general iterative method. The convergence of such parallel asynchronous multisplitting methods can also be studied by partial ordering techniques [21]. Nevertheless the Implementation of such parallel iterative methods can be delicate due to the fact that it is very hard to minimize the weight of communications.*

5. Application to the solution of various pseudo-linear problems

The present section focuses in solving partial differential equations of pseudo-linear evolution problems. Let us consider the situation where a linear operator is perturbed by a diagonal increasing nonlinear operator. For example, problem (14) arises from the implicit temporal discretization of parabolic problems that appear in plasma physics and have been studied in [22]; such discretization with respect to the time produces a sequence of stationary problems similar to (14) ; for example the evolution problem occurring in plasma physics can be written as follows

$$\begin{cases} \frac{\partial u(t,x)}{\partial t} - \Delta u(t,x) + c^t\nabla u + e^{bu} = \bar{g}(t,x), \text{ everywhere in } [0,T]\times\Omega, b>0, \\ u(t,x) = 0, \text{ everywhere in } [0,T]\times\partial\Omega, \\ u(0,x) = u_0(x), \text{ everywhere in } \Omega, \end{cases} \tag{32}$$

where $T > 0$, $u_0 : \Omega \to \mathbb{R}$ is the initial condition; after temporal discretization the stationary problem associated to the implicit time marching scheme, is defined as follows

$$\begin{cases} \frac{u}{\delta_\tau} - \Delta u + c^t\nabla u + e^{bu} = g, \text{ everywhere in } \Omega \subset R^3, \\ u = 0, \text{ everywhere in } \partial\Omega, \end{cases} \tag{33}$$

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

Another application similar to the formulation (14) concerns the study of the Stefan problem formulated as follows

$$\begin{cases} \frac{\partial u(t,x)}{\partial t} - \Delta e^{u(t,x)} = \bar{g}(t,x), \text{ everywhere in } [0,T]\times\Omega, \\ u(t,x) = 0, \text{ everywhere in } [0,T]\times\partial\Omega, \\ u(0,x) = u_0(x), \text{ everywhere in } \Omega, \end{cases} \tag{34}$$

the other variables being similar to the previous ones; (34) is a strongly nonlinear problem which can be replaced by using the following change of variable

$$v(t,x) = e^{u(t,x)} - 1,$$

in such a way that (34) can now be written as follows

$$\begin{cases} \frac{\partial Log(1+v(t,x))}{\partial t} - \Delta v(t,x) = \bar{g}(t,x), \text{ everywhere in } [0,T]\times\Omega, \\ v(t,x) = 0, \text{ everywhere in } [0,T]\times\partial\Omega, \\ v(0,x) = v_0(x), \text{ everywhere in } \Omega; \end{cases} \tag{35}$$

if one considers once again an implicit time marching to solve problem (35), then one has to solve a similar problem to (14).

The same type of problem also appears in the modified porous medium equation, formulated as follows when it is equipped with Dirichlet boundary condition; then one has to find the solution $u(t,x)$, the nonnegative density function, which satisfies the following boundary value problem

$$\begin{cases} \frac{\partial u(t,x)}{\partial t} - \Delta\left((\frac{u(t,x)}{u(t,x)+\theta})^m \right) = 0, \text{ everywhere in } [0,T]\times\Omega, \\ u(0,x) = \psi(x), \\ u(t,x) = 0 \text{ on } \partial\Omega, \end{cases} \tag{36}$$

where ψ is the initial condition, θ is a strictly positive real number and m is an integer such that $m > 1$. Clearly, the problem (36) is a nonlinear problem. Consider now the change of variables

$$v(t,x) = (\frac{u(t,x)}{u(t,x)+\theta})^m;$$

then the problem (36) can be written as follows

$$\begin{cases} \frac{\partial}{\partial t}(\frac{\theta.(v(t,x))^{\frac{1}{m}}}{1-(v(t,x))^{\frac{1}{m}}}) - \Delta v(t,x) = 0, \text{ everywhere in } [0,T]\times\Omega, \\ v(0,x) = (\frac{\psi(x)}{\psi(x)+\theta})^m, \\ v(t,x) = 0 \text{ on } \partial\Omega. \end{cases} \tag{37}$$

Since $0 \leq u(t,x)$, then the mapping $u \to \frac{u}{u+\theta}$ is an increasing function with respect to u. Moreover, $u = \frac{\theta.v^{\frac{1}{m}}}{1-v^{\frac{1}{m}}}$ is well defined; indeed obviously the mapping $v \to \frac{\theta.v^{\frac{1}{m}}}{1-v^{\frac{1}{m}}}$ is increasing with respect to v and since obviously $v^{\frac{1}{m}} < 1$,

$\theta > 0$ and consequently $1 - v^{\frac{1}{m}} > 0$, which involves $1 - v^{\frac{1}{m}} \neq 0$. If one considers now the temporal discretization of problem (37) by an implicit time marching scheme, so, at each time step, one has to solve the following stationary problem

$$\begin{cases} \phi^{(q+1)}(v^{(q+1)}(x)) - \delta_t.\Delta v^{(q+1)}(x) = \phi^{(q)}(v^{(q)}(x)) \equiv \bar{\phi}, e.w. \text{ in } \Omega, q \geq 0, \\ v^{(q+1)}(x) = 0 \text{ on } \partial\Omega, q \geq 0. \end{cases} \tag{38}$$

where $v^{(q+1)}(x) \approx v((q+1)\delta_t, x)$, $\bar{\phi}$ is given and defined thanks to the result obtained at the previous time step, and the mapping $v^{(q)} \to \phi^{(q)}(v^{(q)})$ is given by

$$\phi^{(q)}(v^{(q)}(x)) = \frac{\theta.(v^{(q)}(x))^{\frac{1}{m}}}{1 - (v^{(q)}(x))^{\frac{1}{m}}}, \forall q$$

such that $\phi^{(0)}(v^{(0)}(x)) = \frac{\theta.(v(0,x))^{\frac{1}{m}}}{1-(v(0,x))^{\frac{1}{m}}}$. Note that, in a similar way, the same problem equipped with homogeneous Dirichlet-Neumann boundary condition, can also be considered.

Problem (14) occurs also in some application in biology and matches, for example, in the optimal control of the blood outlet concentration of an artificial vein modeled by [17] as follows

$$\begin{cases} \frac{\partial u(t,x)}{\partial t} - d\Delta u(t,x) + c^t \nabla u + \rho \frac{u}{1+u} = \bar{g}(t,x), \text{ everywhere in } [0,T] \times \Omega, b > 0, \\ u(t,x) = 0, \text{ everywhere in } [0,T] \times \partial\Omega, \\ u(0,x) = u_0(x), \text{ everywhere in } \Omega, \end{cases} \tag{39}$$

where d is the diffusion coefficient, c is the rate of blood flow, $\rho > 0$ is a constant parameter; after implicit discretization with respect to time, a problem similar to (14) is obtained. Note that the derivative of $\frac{u}{1+u}$ is equal to $\frac{1}{(1+u)^2}$ and is positive so that the mapping $u \to \frac{u}{1+u}$ is increasing.

Another possible last application is relative to the continuous casting corresponding to a process between the metal making and rolling; this process allows the transformation of a liquid metal into a solid metal in a continuous way. The mathematical model governing this industrial application can be summarized as follows for $i = 1, 2, 3$

$$\begin{cases} \rho_i c_i \frac{\partial u_i}{\partial t} - div(\lambda_i \nabla u_i) = 0, & in \ [0, t_{final}] \times \Omega_i, \\ u_i(x, t = 0) = u_{i,0}(x), & I.C. \\ -\lambda_i \frac{\partial u_i}{\partial n} = \psi_i(u_i), & on \ \gamma_1, \\ \text{B.C.} & \end{cases} \tag{40}$$

where $u_i, i = 1, 2, 3$, denotes the temperature, $i = 1$ (respectively $i = 2$ and $i = 3$) refers to a liquid zone Ω_1 (respectively to a mushy zone Ω_2 and to a solid zone Ω_3), *I.C.* denotes the initial conditions, B.C. describes Dirichlet or Neumann boundary conditions depending on the physical reality of each zone $\Omega_i, i = 1, 2, 3$, $\rho_\rangle$ is the specific heat, $c_\rangle$ is the metal density, $\lambda_\rangle$ is the thermal conductivity and

$$\psi_i(u_i) = h_{cv,i}(u_i - u_{ext}) + \zeta(u_i^4 - u_{ext}^4), \ i = 1, 2, 3,$$

$h_{cv,i}$ and ζ are physical constants. Since, obviously, in each zone $\Omega_i, i = 1, 2, 3$, the temperature in an industrial casting is strictly positive, then the mapping $u_i \to \psi_i(u_i)$ is increasing and the problem of steel solidification in continuous casting falls within the general formalism here considered except that nonlinearity intervenes on the border.

6. Implementation of the implicit time marching scheme for the solution of pseudo-linear problem by the Newton-multisplitting method

The parallel implementation of an implicit or semi-implicit time marching scheme to solve a 3D pseudo-linear evolution problem requires two complementary phases: first the implementation of an implicit or semi-implicit time marching scheme where at each time step one has to solve a stationary problem and secondly, in the time loop, the solution by an iterative method of the corresponding algebraic system associated to this stationary problem.

In this work, the Newton-multisplitting method is used to solve each stationary problem at each time step. The data of this latter problem are partitioned among processors of the computing platform. Figure 1 shows an example of data partitioning of a stationary problem of size $nx \times ny \times nz$ among two blocks of four processors. In this case, each block of processors is in charge of a portion of data of size $\frac{nx}{2} \times ny \times nz$ and in turn each processor within a block is in charge of a portion of data of size $\frac{nx}{4} \times \frac{ny}{2} \times nz$. In such a partitioning, data on intra and inter block borders are shared between neighboring processors. It should be noted that in this paper no difference is made between processors or cores.

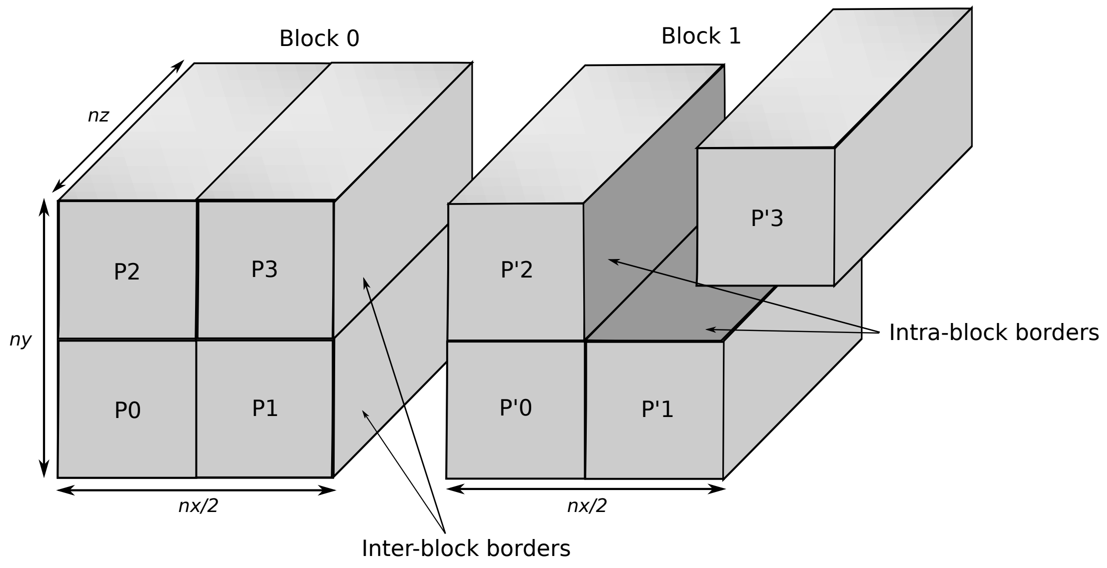


Figure 1: Example of a data partitioning of a problem among blocks of processors.

All codes of the algorithms presented hereafter are written in C language and parallelized with facilities provided by the high-performance message passing library OpenMPI v2. The parallel algorithms are implemented in two ways: synchronous and asynchronous. In the synchronous version, *MPI_Send*() and *MPI_Recv*() routines are used to perform the communications. In contrast, in the asynchronous version, the MPI non-blocking communications *MPI_Isend*(), *MPI_Irecv*() and *MPI_Test*() are used.

6.1. Implementation of the time marching scheme

Algorithm 1 presents the implementation of the first step of the solution of an evolution problem, i.e. the implicit time marching scheme. At each time step a Newton-multisplitting iteration is computed to solve each stationary problem described by an algebraic system (from line 2 to line 7 in Algorithm 1). It is necessary to set-up a synchronization point at each time step between the blocks of processors before proceeding to the next time step. This in order to ensure the coherence of the computation performed by the solution using an implicit or semi-implicit time marching scheme to solve an evolution problem. In Algorithm 1, line 6, the synchronization point is implemented by the MPI collective communication routine *MPI_Barrier*().

6.2. Implementation of the Newton-multisplitting method at each time step

In order to solve each stationary pseudo-linear problem (15) at each time step one the Newton-multisplitting method is used. The pseudo-linear problems presented in Section 5 are solved on a computing platform composed of m blocks of q processors physically adjacent or geographically distant. Each block of processors is assigned to a splitting of the problem to be solved as presented in the mathematical description in Section 4. In a block, processors solve the splitting using a parallel iterative method with a synchronous iteration corresponding to an inner iteration of the multisplitting method. In contrast, the global algebraic system is solved by an asynchronous iteration corresponding to an outer iteration. Communications to exchange the intra and inter block shared data are

Algorithm 1: Implementation of the implicit time marching scheme.

```
   Output: Solution $U_{new}$
1  Set initial value $U_0$
2  for each time step do
3      compute the right-hand side of the scheme : $\bar{G} \leftarrow dt.G + U_0$
4      solve on each block the algebraic system derived from the stationary problem : $U_{new}$
5      $U_0 \leftarrow U_{new}$
6      barrier of synchronization
7  end
```

illustrated in the example given in Figure 2. In addition, as can be seen in the figure, the communications intra-blocks are synchronous and those inter-blocks are synchronous or asynchronous. Thus, each block of processors can solve a splitting independently from other blocks which solve other splittings of the same stationary problem.

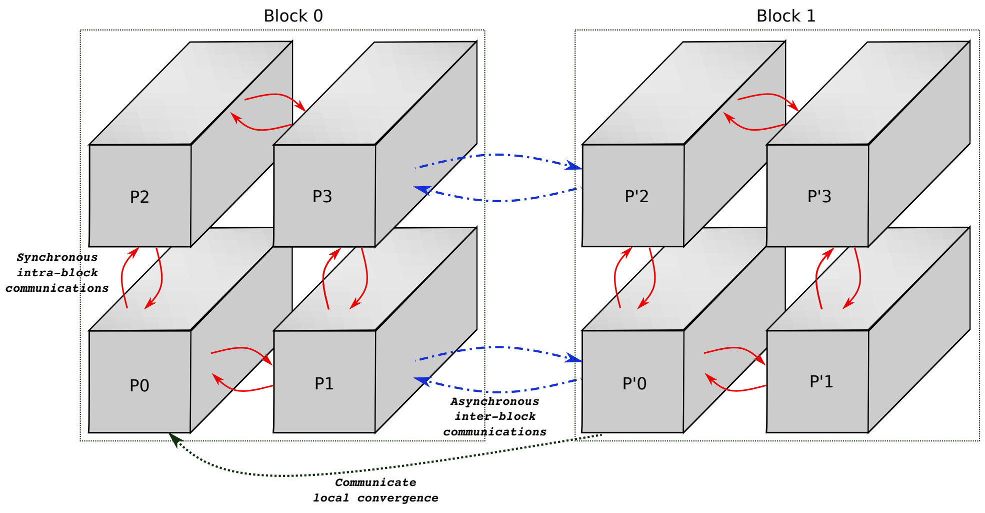

Block 0
Block 1
P2
P3
P'2
P'3
P0
P1
P'0
P'1
Synchronous intra-block communications
Asynchronous inter-block communications
Communicate local convergence

Figure 2: Example of an interconnection of two blocks of four processors.

Algorithm 2 presents the parallel Newton-multisplitting method to solve the target problem, such that at each time step a pseudo-linear stationery problem (15) is solved. It presents the main key points of the Newton-multisplitting method performed in a block of processors. The same algorithm is executed by m blocks of processors to solve in parallel a stationary pseudo-linear problem (15). All variables indexed with l in Algorithm 2 are local to a block of processors. First, according to the previous algorithm (i.e. Algorithm 1), a loop for the time marching scheme is implemented (see from line 2 to line 16 in Algorithm 2). Then, the Newton method is used to linearize the nonlinear system to be solved (see from line 3 to line 14 in Algorithm 2). The multisplitting method is used to solve the linear systems issued from the linearization (see from line 7 to line 11), so that each system is associated to m splittings as shown in (42).

For each splitting l, $l \in \mathbb{N}$ and $1 \leq l \leq m$, starting with an initial guess $U^{l,(0)}$ computed according to the initialization presented in subsection 3.2.1, the following equations should be solved by the Newton-multisplitting method

$$AU^l + \Phi(U^l) - G = 0. \tag{41}$$

Globally, at iteration i of the Newton method and considering for example the block Jacobi method obtained by choosing $\mathcal{M}^l$ and $\tilde{W}_l$ given by (28)-(29), the following algebraic systems are solved with an iterative multisplitting method

$$\hat{C}_{l,l}(U^{l,(i)})\delta U_l^{l,(i)} = B_l, 1 = 1, \ldots, m, \tag{42}$$

Algorithm 2: Parallel Newton-multisplitting method performed on a block of processors

```
Output: Solution $U^{l,(i)}$
1  Set initial solution: $U^{l,(0)} = 1.0$
2  for each time step do
3      while $\|\delta \mathbf{U}^{(i+1)}\|_2 \geq \varepsilon_{Newton}$ do
4          Reset the initial local solution $\delta U^{l,(i)}$ to an arbitrary value
5          Compute the global right-hand side of Newton $\hat{G}(U^{l,(i)})$ : Formula (45)
6          Update local sparse matrix $\hat{C}_{l,l}$: Formula (43)
7          while $\|\delta U^{l,(i)} - \delta U^{l,(i-1)}\|_\infty \geq \varepsilon_{Multisplitting}$ do
8              Compute local right-hand side: $B_l$ Formula (44)
9              Parallel GMRES to solve $\delta U^{l,(i)}$: Formula (42)
10             Exchange local shared values of $\delta U^{l,(i)}$ with neighbor blocks
11         end
12         Compute the local solution on block l: $U^{l,(i+1)} = U^{l,(i)} + \delta U^{l,(i)}$
13         Compute the global correction: $\delta \mathbf{U}^{(i+1)}$
14     end
15     Barrier of synchronization
16 end
```

where according to the choice of the weighting matrices $D_{l,k}$

$$\hat{C}_{l,l}(U^{l,(i)}) = (A + \frac{\partial \Phi(U^{l,(i)})}{\partial U})_{l,l}, \tag{43}$$

denotes the block diagonal resulting from a block partitioning of the matrix

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

and B_l is given by

$$B_l = \hat{G}_l(U^{l,(i)}) - \sum_{k \neq l} A_{l,k} \delta U_k^{k,(j(k))}, \tag{44}$$

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

$$\hat{G}(U^{l,(i)}) = G - \Phi(U^{l,(i)}) - AU^{l,(i)}, \tag{45}$$

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. In addition, the values of the components of the 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 an iterative method.

So, at each iteration i of the Newton method, a linear system

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

is solved in parallel by a block of processors using an iterative multisplitting method, and the solution is updated

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

until convergence of the iterative method. Since the matrix $\hat{C}(U^l)$ is an M-matrix, the multisplitting method will converge according to the results presented in Section 4.

In other words, each sub-system (42) is solved independently by a block of processors and communications are required to update the right-hand side of each sub-system. The local vectors updated according to (44) by each block represent the data dependencies between the different blocks. In the multisplitting method, as previously mentioned, there are two levels 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 (42) by an iterative method, due to the sparsity of the matrix A. Iterative methods are well adapted to this kind of problem. In the proposed implementation a Krylov method was chosen to solve each block (42). 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.

In Algorithm 2 each splitting l, $l \in \mathbb{N}$ and $1 \leq l \leq m$, of a linear system is assigned to a block of processors. First the local right-hand side B_l involved in the formula (42) and defined by (44) (see line 8 in Algorithm 2) is computed. Then each sub-system is solved in parallel by using the well-known Krylov method GMRES [28] (see line 9 in Algorithm 2). GMRES iterations represent the inner-iteration of the multisplitting method. Iterations of GMRES method are synchronous inside a block of processors. When the inner-iteration of the multispitting method has converged, each local solution $\delta U^{l,(i)}$ in formula (42) of the sub-system is exchanged to all neighbor block (see line 10 in Algorithm 2).

After each sub-system is solved, the outer-iteration of the multisplitting method is applied to compute the solution of the global linear system. The outer-iteration (*i.e.* intra-blocks communications) can be either synchronous or asynchronous. When the outer-iteration of the multisplitting method has converged, the solution of the nonlinear system is then updated at each Newton iteration (see line 12 in Algorithm 2). Then, the global correction $\delta\mathbf{U}^{(i+1)}$ is performed.

For the solution of each linearized system, the convergence of the multisplitting method is detected when the value of $\delta U^{l,(i)}$ is stabilized corresponding to the following stopping test

$$||\delta U^{l,(i)} - \delta U^{l,(i-1)}||_\infty \leq \varepsilon_{Multisplitting} \tag{46}$$

where $\varepsilon_{Multisplitting}$ is the tolerance threshold used to stop the iterative process. The global convergence of the synchronous Newton-multisplitting method is detected when the value of $\delta\mathbf{U}^{(i+1)}$ is stabilized corresponding to the following global stopping test

$$||\delta\mathbf{U}^{(i+1)}||_2 \leq \varepsilon_{Newton} \tag{47}$$

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

In the asynchronous version, the global convergence is detected when all blocks of processors have locally converged. In fact, in each block, a processor is designed as a *master* of the block (for example processor p_0) and one of these masters is considered as a *super-master* (for example processor p_0 of block b_0). As shown in Figure 2, the masters communicate the local convergence state of their blocks to the super-master processor. This latter maintains an array of m elements, each corresponding to a block, such that each element l, $l \in \mathbb{N}$ and $1 \leq l \leq m$, has a boolean value representing the convergence of the block l. An element l of the array is $True$ if the local convergence of block l is detected or $False$ otherwise. The global convergence of the asynchronous iteration is detected when all elements of the array are set to $True$. In this case, the super-master processor broadcasts a message to all other masters of other blocks to stop the computation.

7. Parallel experiments

Parallel experiments have been performed on the Grid'5000 a grid environment located in France [8]. Grid'5000 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 and provides access to a large amount of resources on 10 sites, 29 clusters, 1060 nodes, 10474 cores, grouped in homogeneous clusters, and featuring various technologies and resources (Intel, AMD, Myrinet, Infiniband, GPU clusters, 10G Ethernet ...). Figure 3 illustrates the sites of the Grid'5000 architecture.

Parallel simulations for the solution of our 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 machines on each cluster and links between different sites range from 10-Gbps.

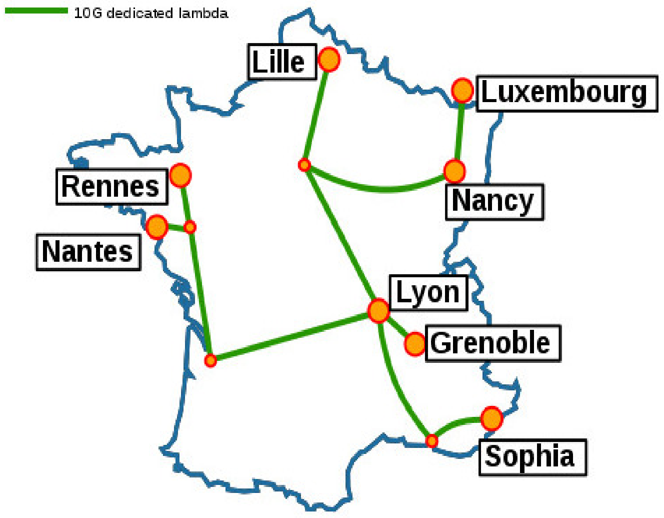


Figure 3: Grid'5000 architecture.

Let us consider two parallel asynchronous evolution problems i.e. a problem occurring in plasma physics (32) and a problem occurring in biology which can be applied, for example, to the optimal control of the blood outlet concentration of an artificial vein (39). For the parallel simulations performed on distant and heterogeneous coupled clusters of the Grid'5000 platform, several sizes of the algebraic systems to solve have been chosen. So problems are defined on a square $\Omega \equiv [0, 1[^3$ and we consider in the achieved simulations on each axis 240, 360 and 480 discretization points. The size of the algebraic systems to solve are respectively then 240^3, 360^3 and 480^3 (see Table 1).

Size of the algebraic systems
$240^3 = 13\ 824\ 000$
$360^3 = 46\ 656\ 000$
$480^3 = 110\ 592\ 000$

Table 1: Size of the algebraic systems.

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

Site	Cluster	Processors type	GHz	CPU	cores/CPU	RAM	NIC	details
Nantes	ecotype	Intel Xeon E5-2630L v4	1.8	2	10	128 GB	10 Gbps	[32]
Rennes	parapide	Intel Xeon X5570	2.9	2	4	25 GB	1 Gbps	[33]

Table 2: Characteristics of machines on each site.

So, the paper presents results for 240^3, 360^3 and 480^3 size of algebraic systems discretized by a finite difference scheme, on a grid composed of two distant clusters and up to 600 cores have been used to make these experiments. As the high-performance message passing library MPI has been used, the MPI machine file used for experimentation alternates dedicated computers on clusters to reach the desired number of cores to perform the experiments.

According to the duration of the experiments and to constraints of Grid'5000 exploitation, for the time marching scheme, the number of time steps has been limited to 3 and $m = 2$ has been fixed for the number of blocks, so in all the results $m \times q$ configuration has been 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.

Moreover, 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 39 of b, c, d and ρ are respectively $b = 0.2$, $c = 0.3$, $d = 1$ and $\rho = 1$.

7.1. *Parallel simulations for the synchronous and asynchronous plasma physics algorithm and biology algorithm.*

The following Table 3 resumes grid tests on Grid'5000 platform.

Experimentations				Newton - multisplitting			Biology Multisplitting		
Clusters (nb. of machines)		Cores	Machines	Tables		Figures	Tables		Figures
				Sync	Async		Sync	Async	
ecotype (22)	parapide (20)	600	42	4	5	4	6	7	5

Table 3: Simulations on GRID'5000.

The results of grid simulations for the synchronous and asynchronous versions are summarized in Tables 4 and 5 for the plasma physics algorithm algorithm and in Tables 6 and 7 for the biology algorithm.

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 both algorithms, 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 5 and 7 which contain asynchronous results, the values of τ for each problem are given.

The results of the synchronous and asynchronous elapsed times with respect to the different domain sizes are presented in Figure 4 for the first problem and in Figure 5 for the second problem.

7.1.1. *Plasma physics algorithm.*

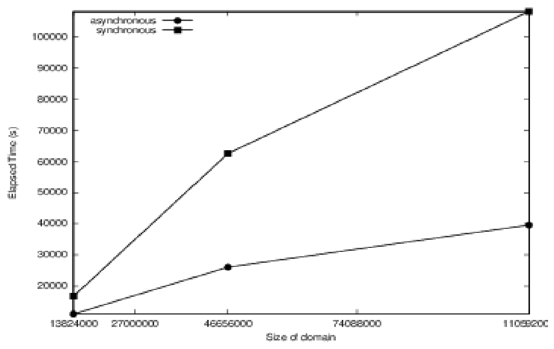
Results with 600 *cores on* 2 *clusters with* $m = 2$ *and* $q = 300$.

Table 4: Domain size, iterations, elapsed time on grid (ecotype, parapide) with synchronous and asynchronous parallel algorithm.

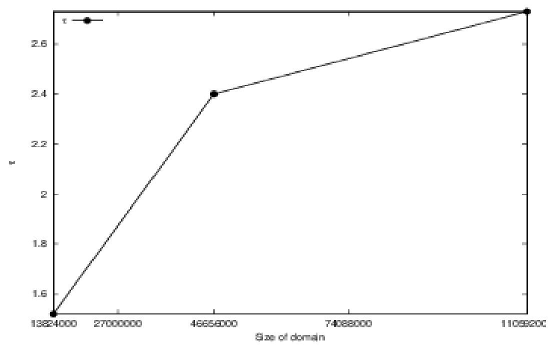
Synchronous results			
Size	Iterations		Elapsed time (sec)
	Multi (Nwt)	GMRES	Total
240^3	16 226 (19)	81 130	16 687
360^3	33 598 (16)	167 990	62 516
480^3	56 328 (18)	281 640	108 040

Table 5: 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	11 566 (17)	57 830	10 977	1.52
360^3	21 546 (24)	107 732	26 021	2.40
480^3	31 492 (21)	157 462	39 519	2.73



(a) Elapsed time with respect to domain sizes.



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

Figure 4: Plasma physics algorithm : Results with 600 cores on 2 clusters

7.1.2. Biology algorithm.

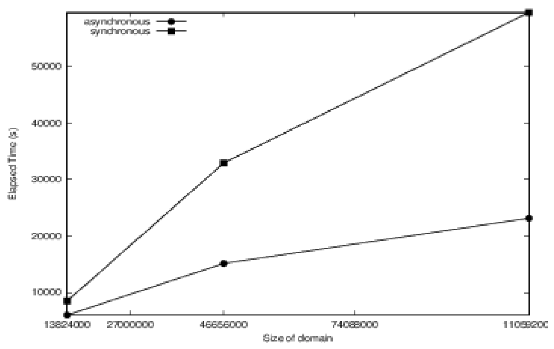
Results with 600 *cores on* 2 *clusters with* $m = 2$ *and* $q = 300$.

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

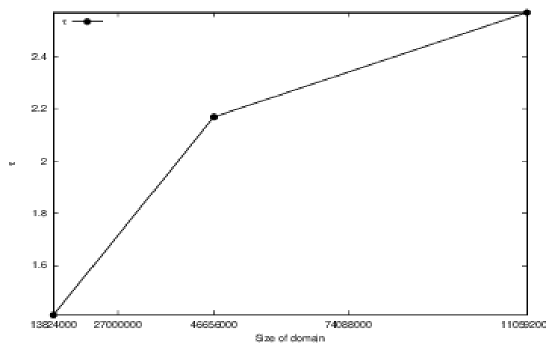
Synchronous results			
Size	Iterations		Elapsed time (sec)
	Multi (Nwt)	GMRES	Total
240^3	8 294 (9)	41 470	8 523
360^3	17 557 (9)	87 785	32 913
480^3	31 214 (9)	156 070	59 478

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	6 369 (13)	31 847	6 046	1.41
360^3	12 522 (10)	62 612	15 161	2.17
480^3	23 897 (11)	119 485	23 124	2.57



(a) Elapsed time with respect to domain sizes.



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

Figure 5: Biology algorithm : Results with 600 cores on 2 clusters

7.2. Experiments analysis.

In our experiments, the elapsed times obtained for the asynchronous method is significantly better than the ones obtained for the synchronous method; this is essentially due to the latency of the network which implies costly communications. 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 good performances. Moreover, the value of τ increases with the value of n the size of the algebraic system to solve.

In Table 4 and Table 5, parallel simulations for the Plasma physics algorithm on grid show that the asynchronous mode is more efficient than the synchronous one. The parallel asynchronous simulation elapsed times are inferior to those obtained in the synchronous mode and it can be observed that the values of τ vary between 1.52 and 2.73. So, for the size of 480^3, the asynchronous simulation works almost three times faster than the synchronous version.

Similarly, in Table 6 and Table 7, the parallel simulations for the Biology algorithm 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 1.41 and 2.57. In this case, the size of the algebraic system to solve is equal to 480^3 and the asynchronous simulation works two and a half times faster than the synchronous version.

In view of the results indicated in Table 4 to 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.

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.

8. Conclusion

In the present study we have presented a mixed method combining parallel asynchronous methods such as an outer-iteration combined with the inner-iteration constituted by the Krylov method for the solution of diagonal subproblems. Such a calculation method has been used for the solution of pseudo-linear evolution problems and implemented in a computing platform composed of m clusters, each of them composed of q processors, physically adjacent or geographically distant.

It should be noted that in the numerical method used, the synchronous inner-iteration part resulting from the use of GMRES is irreducible which may degrade the performances of the asynchronous outer-iterations. Moreover, it is important to note that despite this drawback, the use of asynchronous outer-iterations is still very efficient.

In future work, from an experimental point of view, the use of such mixed methods for the solution of pseudo-linear problem, arising in boundary value, will be implemented on cloud architecture.

9. 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).

References

[1] J. Arnal, V. Migallón, J. Penadés, Parallel Newton two-stage multisplitting iterative methods for nonlinear systems, BIT Numerical Mathematics, 43, 849 - 861, 2003.
[2] J.M. Bahi, J.C. Miellou, K. Rhofir, Asynchronous multisplitting methods for nonlinear fixed point problems, Numerical Algorithms, 15, 315 - 345, 1997.
[3] Z.Z. Bai, The monotone convergence rate of the parallel nonlinear AOR method, Computers Math. Appl., 31-7, 21 - 31, 1996.

[4] Z.Z. Bai, Asynchronous multisplitting AOR methods for a class of systems of weakly nonlinear equations, Appl. Math. Comp., 98, 49 - 59, 1999.
[5] Z. Z. Bai, D.R. Wang, Improved comparison theorem for the nonlinear multisplitting relaxation method, Computers Math. Appl., 31-8, 23 - 30, 1996.
[6] G. Baudet, Asynchronous iterative methods for multiprocessors, Journal Assoc. Comput. Mach., 25 , 226-244, 1978.
[7] D. Bertsekas, J. Tsitsiklis, Parallel and Distributed Computation, Numerical Methods, Prentice Hall Englewood Cliffs N.J., 1989.
[8] 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, Grid'5000: A large scale and highly reconfigurable experimental grid testbed, International Journal of High Performance Computing Applications 20 (4), 481–494, 2006.
[9] 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.
[10] D. Chazan, W. Miranker, Chaotic relaxation, Linear Algebra Appl., 2 , 199-222, 1969.
[11] R. Couturier, C. Denis, F. Jézéquel, GREMLINS: a large sparse linear solver for grid environment, Parallel Comput 34(6–8), 380 - 391, 2008.
[12] R. Couturier, L. Ziane Khodja, A scalable multisplitting algorithm to solve large sparse linear systems, The Journal of Supercomputing, 69 (1), 200 - 224, 2014.
[13] M. El Tarazi, Algorithmes mixtes asynchrones. Etude de convergence monotone, Numerische Mathematik, 44, 363-369, 1984.
[14] A. Frommer, Parallel nonlinear multisplitting methods, Numerische Mathematik, 56, 269 - 282, 1989.
[15] 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.
[16] MT Jones, DB. Szyld, Two-stage multisplitting methods with overlapping blocks, Numerical Linear Algebra with Applications, 3, 113 - 124, 1996.
[17] J.P. Kernevez, Enzyme mathematics, Studies in Mathematics and its Applications, vol. 10, Ed. J.L. Lions, G. Papanicolaou, R.T. Rockafellar, Nort-Holland, 1980.
[18] J.-C. Miellou, Algorithmes de relaxation chaotique à retards, RAIRO Analyse numérique, 1 , 55-82, 1975.
[19] J.-C. Miellou, Itérations chaotiques à retards, étude de la convergence dans le cas d'espaces partiellement ordonnés, CRAS Paris, 280 (1975), 233-236.
[20] J.-C. Miellou, Asynchronous iterations and order intervals, *In Parallel Algorithms and Architectures,* eds. M. Cosnard et al., North Holland, 85-96, 1986.
[21] Miellou J.C., El Baz D., Spiteri P., A new class of asynchronous iterative algorithms with order interval, Mathematics of Computation, 67-221, 237-255, 1998.
[22] J. Mossino, Sur certaines inéquations quasi-variationnelles apparaissant en physique, CRAS Paris, 282, 187 - 190, 1976.
[23] 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.
[24] J. Ortega, W. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York, 1970.
[25] 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.
[26] W. Rheinboldt, On M-functions and their application to nonlinear Gauss-Seidel iterations and to network flows, J. Math. Anal. and Appl., 32 (1970), 274–307.
[27] Y. Saad, Iterative methods for sparse linear systems, SIAM, 2003.
[28] 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.
[29] P. Spiteri, J.-C. Miellou, D. El Baz, Asynchronous Schwarz alternating methods with flexible communication for the obstacle problem, Réseaux et systèmes répartis, 13 [1), 47 - 66, 2001.
[30] D.R. Wang, Z.Z. Bai, D.J. Evans, On the monotone convergence of multisplitting method for a class of system of weakly nonlinear equations, Intern. J. Computer Math., 60, 229 - 242, 1996.
[31] R. E. White, Parallel algorithms for nonlinear problems, SIAM J. Alg. Discrete Meth., 7, 137 - 149, 1986.
[32] `https://www.grid5000.fr/w/Nantes:Hardware#ecotype`
[33] `https://www.grid5000.fr/w/Rennes:Hardware#parapide`