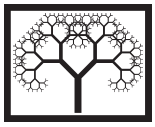
Asynchronous parallel multi-splitting mixed methods

Pierre Spitéri, Lilia Ziane Khodja, Raphael Couturier

Paper 15



Proceedings of the Sixth International Conference on
Parallel, Distributed, GPU and Cloud Computing for Engineering,
P. Iványi and B.H.V Topping (Editors)
Civil-Comp Press, Stirlingshire, Scotland

Asynchronous parallel multi-splitting mixed methods

P. SPITERI[1], **L. ZIANE-KHODJA**[2], **R. COUTURIER**[2]
[1]IRIT- INPT, University of Toulouse, Toulouse, France
[2]FEMTO – ST Institute, University of Bourgogne Franche Comté, Belfort, France

Abstract

The present study deals with the solution of univalued pseudo - linear problems using parallel asynchronous multisplitting methods . With appropriate and realistic assumptions, the behavior of such parallel iterative algorithms will be analyzed by contraction techniques. An application to a discretized boundary value problem is presented and the parallel experiments are analyzed.

Keywords: asynchronous parallel algoritm, high performance computing, multisplitting methods, Krylov method, Newton method, discretized pseudo-linear problem, large scale systems, nonlinear boundary value problems

1 Introduction

The present study focuses on the analysis and application of mixed multisplitting methods to solve pseudo - linear stationary problems. These problems are stationary either intrinsically or as the 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

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

where in the sequel A has the property of being a large scale M-matrix, F 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 scheme, finite volumes scheme or finite elements method provided that, in this last case, the angle condition is verified. In the present paper, the operator $U \rightarrow \Phi(U)$ in problem (1) is considered as a strongly non-linear univalued operator.

In such a case, problem (1) will be solved by a specific method resulting from a local linearization corresponding to the implementation of the iterative Newton method. Thus the calculation method consists in solving a large sparse linear system. This linear system is then associated with a fixed point problem which will be solved 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 can be seen in the sequel 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 iterations asynchronous towards the solution of the target problem.

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, ZZ bai, R. Bru, A. Frommer, V. Migalon, J. Penades. D. Szyld, ... etc ... in collaboration with several coauthors (see [9]-[20]) for the solution of linear and nonlinear problems. Nevertheless, it should be noted that these previous works do not concern solution of multivalued problem excepts for the work of J. Bahi et al. [21] developed in an hilbertian context. So in the sequel, this last study is extended to the case of the non hilbertian context. These multisplitting methods are then applied to solve the target problem (1), the convergence analysis being still carried out by contraction techniques with respect to a weighted uniform norm. To effectively solve the model problems, efficient methods are used to solve 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) is solved by this last type of algorithm.

As application, we consider a diffusion-convection problem perturbed by an increasing diagonal operator [23], the problem being solved by a mixed Newton - multisplitting method, each linearized subsystem being solved by the generalized minimal residual method (GMRES). Thus we present and discuss the results of parallel simulation achieved on a cluster.

The present paper is organized as follows. In section 2 the formulation of synchronous and, more generally, asynchronous parallel algorithms is presented 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 detailled. Section 4 is devoted to the presentation of boundary value problems, which, after appropriate discretization lead to solve pseudo – linear algebraic systems. Thus the following section is devoted to the presentation of implementation of the studied parallel numerical methods. Section 6 presents the results of parallel experiments achieved on a local cluster. Finally a conclusion and some future studies conclude the paper.

2 Parallel asynchronous algorithms associated to pseudo linear problems

Consider problem (1) and let us transform such problem into a fixed point problem as follows

$$U = F(V) \tag{2}$$

where F is a fixed point mapping defined in an implicit way; in the case of problem (1) such fixed point mapping is naturally defined as follows by considering a block decomposition in α blocks

$$A_{i,i}U_i + \Phi_i(U_i) = F_i - \sum_{j=1 j\neq i}^{\alpha} A_{i,j}V_j,\ j = 1, \ldots, \alpha, \tag{3}$$

where $A = (A_{i,j})$ corresponds to the block decomposition,U_i and F_i are the block components of the vectors U and F.

Using the fixed point equation (2), 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 such a computational method, to make the most of computing power by eliminating idle times due to blocking expectations, synchronizations are not necessarily required which avoids waiting for the communication of the values computed by the other processors; thus each processor 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 component selection strategy at each step of the calculation; this strategy is in fact a non-empty subset of the set $\{1, 2, \ldots, \alpha\}$ which 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. During the first work on asynchronous parallel iterations, delays were bounded (see [1] and [2]); in fact in [3] G. Baudet has extended the framework of the study to cover cases in which delays are no longer bounded, allowing for cases of failure of one or more processors to be taken into account.

The formulation of the parallel asynchronous method is general. Indeed when the values of the components of the iterate vector, representing the interactions between the parallel processes, are not delayed, the corresponding algorithm is in fact the parallel synchronous method; such a situation is in fact the parallel method of successive iteration and corresponds to the parallel block Jacobi matrix. Moreover, in the context of parallel synchronous methods, for particular choice of strategy we find the block Jacobi method, or the block Gauss – Seidel method and the Alternating Direction Implicit (A.D.I.) method.

Assuming the following assumptions

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

$$\text{the mapping } U \to \Phi(U) \text{ is a diagonal monotone operator}, \tag{5}$$

then, thanks to the use of a result in [24]-[25] we know that, for any block decomposition the fixed point mapping is contractive with respect to a uniform weighted norm; thus this result allows us to state that on the one hand the solution of pseudo – linear problem (1) exists and is unique and on the other hand that the parallel synchronous and asynchronous methods applied to the parallel solution of problem (1) are convergent toward $U^\star$ solution of problem (1) whatever the initial guess is.

In practice we do not have as many processors as the blocks in matrix A. Let $\beta << \alpha$ the number of processors. So, from an algorithmic point of view we gather several adjacent blocks of the matrix A in β large blocks and we consider large blocks decomposition of U, F and Φ

accordingly, such decomposition corresponding in fact to a subdomain method without overlapping. Now, using the result stated in [24]-[25], since assumptions (4) and (5) are satisfied, due to the fact that any fixed point mapping associated to any decomposition of the problem to solve is contractive, the parallel synchronous and asynchronous subdomain methods without overlapping converge toward $U^\star$ solution of problem (1) whatever the initial guess is.

Moreover we can also solve the considered pseudo – linear problems 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

$$\bar{A}\bar{U} - \bar{F} + \bar{\Phi}(\bar{U}) = 0. \tag{6}$$

Using a result of D.J. Evans and Van Deren (see [26]), if A is an M-matrix, then $\bar{A}$ is also an M-matrix. Moreover, by applying the augmentation process, the diagonal operator $\bar{\Phi}$ is still diagonal monotone operator. So we are in the framework of the study of [24]-[25] and can still apply the results of this paper concerning the convergence of parallel synchronous and asynchronous subdomain methods without overlapping.

3 The multisplitting method

Consider now the solution of problem (1) by the parallel synchronous or, more generally, asynchronous multisplitting method. In our case, due to the fact that a pseudo – linear problem has to be solved by the Newton method, for the solution of the linearized system derived from this last method $\hat{C}\delta U = \hat{F}$, where $\hat{C}$ is an M-matrix, we consider m regular splittings [27] of the matrix A, such that

$$\hat{C} = \mathcal{M}^l - \mathcal{N}^l, l = 1, \ldots, m,$$

where $\mathcal{M}^l, l = 1, \ldots, m$, are M-matrices and in the general case the system $\hat{C}\delta U = \hat{F}$ needs to be solved. For that we associate m fixed point mappings in a similar way than the one considered in (3). Moreover, for the efficient application of the parallel asynchronous multisplitting method, it is usually necessary that each of the m fixed point mappings associated with the problem to solve, are contracting; in our case, since $\hat{C}$ is an M-matrix, such condition is not restrictive thanks to the use of the results previously obtained in [24]-[25]. Indeed, as precised in section 2, that under assumptions (4) and (5), using the result of this latter reference, any fixed point mapping associated with any decomposition of the problem was contracting; since $\hat{C}$ is an M-matrix, the results of the two latter references can also be applied. Consequently, under assumptions (4) and (5), there are guarantees that asynchronous parallel multisplitting methods can be used successfully (see [21]).

Interested readers can refer to the work presented in [7] - [21] for a detailed presentation of asychronous parallel multisplitting methods and, mainly to [21] for target applications. The algorithmic principle of this type of method can be defined as follows. Let us denote by $F^l, l = 1, \ldots, m$, each of the fixed point applications associated with the problem to be solved; then the numerical algorithm consists in computing the m following vectors

$$U^l = F^l(\sum_{k=1}^{m} W_{l,k} V^k), l = 1, \ldots, m, \tag{7}$$

where $V^1, \ldots, V^m$ are m vectors of the space $\mathbb{R}^N$, where N is the dimension of the matrix A, or $\hat{C}$ and $W_{l,k}$ are nonnegative diagonal weighting matices satisfying for all $l \in \{1, \ldots, m\}$

$$\sum_{k=1}^{m} W_{l,k} = Id_l, l = 1, \ldots, m,$$

where, for $l = 1, \ldots, m$, Id_l is the identity.

Note that (7) allows to define an extended fixed point mapping $\mathcal{F}$ and since each of the fixed point mapping F^l associated with the problem to be solved is contracting, then using a similar result to the one stated in [21], the extended fixed point mapping $\mathcal{F}$ is also contracting with respect to an adapted uniform weighted norm. As a consequence the parallel asynchronous multisplitting methods applied to the solution of the linear problem converge.

Note also that considerable saving in computational work may be possible by using such numerical methods, since a component of V^k needs not be used if the corresponding diagonal entry of the weighting matrices are zero; then, in parallel computing, the role of such matrices $W_{l,k}$ may be regarded as determining the distribution of the computational work of the individual processors.

Note finally that according to the weighting matrices $W_{l,k}$ 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. According to [21] the block Jacobi method corresponds then to the following choice of $\mathcal{M}^l$

$$\mathcal{M}^l = diag(Id, .., Id, \hat{C}_{l,l}, Id, .., Id) \tag{8}$$

and to the choice of $W_{l,k} \equiv \tilde{W}_l$ given by

$$\tilde{W}_l = diag(0, .., 0, Id, 0, .., 0) \tag{9}$$

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

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

For each splitting $l, l = 1, \ldots, m$, starting with an initial guess $U^{l,(0)}$ we have to solve

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

by the Newton method; then, globally, at step i of the Newton method we have to solve

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

and then

$$U^{l,(i+1)} = U^{l,(i)} + \delta U^{l,(i)}, \tag{10}$$

until the 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

$$\hat{C}_{l,l}(U^{l,(i)}) = (A + \frac{\partial \Phi(U^{l,(i)})}{\partial U})_{l,l} \tag{11}$$

denotes the block diagonal of the matrix $\hat{C}(U^{l,(i)})$, and since the operator $U \rightarrow \Phi(U)$ 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 $\hat{C}(U^{l,(i)})$, denoted $\hat{A}_{l,k}$, 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

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

which will therefore be solved by a multi-splitting method; since the matrix $\hat{C}(U^l)$ is an M-matrix, the multi-splitting method will converge (see [21]).

For the solution of problem (1) by the Newton method, by considering for example the block Jacobi method obtained by choosing $\mathcal{M}^l$ and $\tilde{W}_l$ given by (8)-(9), the implemented multisplitting method associated to the iteration number i of the Newton method leads to solve iteratively in parallel for $l = 1, \ldots, m$, the algebraic sub-systems

$$\hat{C}_{l,l}(U^{l,(i)})\delta U_l^{l,(i)} = B_l, \tag{12}$$

where B_l is given by

$$B_l = \hat{F}_l(U^{l,(i)}) - \sum_{k \neq l} \hat{A}_{l,k}\delta U_k^{k,(j(k))} \tag{13}$$

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

$$\hat{F}(U^{l,(i)}) = F - \Phi(U^{l,(i)}) - AU^{l,(i)}, \tag{14}$$

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

Then, each sub-system (12) is solved independently by a processor or a set of processors and communications are required to update the right-hand side of each sub-system, such that the vectors updated by the other processors represent the data dependencies between the different blocks. For the target applications, in the implemented multi-splitting method, we have in fact a two level iteration; an external parallel iteration and an inner iteration due to the fact that, since the matrices $\hat{C}_{l,l}$ are also sparse, it is highly recommended to solve the subsystems (12) by an iterative method. In our implementation a Krylov method has been chosen for the solution of each sub-problem (12). It should be noted that for the external parallel iteration, the considered computing method fits well within the formulation of general parallel asynchronous methods described in section 2, since inter-processors communications can be synchronous or asynchronous.

4 Application to the numerical solution of non-linear boundary value problem

There are several kinds of partial differential equations which, after discretization, lead to the solution of pseudo-linear algebraic systems such as the one found in (1). In the sequel we will denote by Ω an open domain included in $\mathbb{R}^3$, $\partial\Omega$ the boundary of Ω, f a sommable

square function and $u \rightarrow \phi(u)$ a diagonal monotone increasing, convex and continuously differentiable nonlinear operator. So the following nonlinear convection – diffusion problems can be considered

$$\begin{cases} -\nu\Delta u + a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} + c\frac{\partial u}{\partial z} + du + \phi(u) = f, \text{ everywhere in } \Omega, \\ u = 0, \text{ everywhere in } \partial\Omega, \end{cases} \tag{15}$$

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

Problem (15) can occur in plasma physics or to model solar ovens [23]; such a problem arises from the implicit temporal discretization of parabolic problems that appear in similar applications modeled as the one modeled below

$$\begin{cases} \frac{\partial u(t,x)}{\partial t} - \nu\Delta u(t,x) + Q^t\nabla u + e^{ru} = g(t,x), \text{ everywhere in } [0,T]\times\Omega, b > 0, \\ u(t,x) = 0, \text{ everywhere in } [0,T]\times\partial\Omega, \\ u(0,x) = u_0(x), \text{ everywhere in } \Omega, \end{cases} \tag{16}$$

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

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

$$\begin{cases} -\nu\Delta u + Q^t\nabla u + \frac{u}{\delta_\tau} + e^{ru} = f, \text{ everywhere in } \Omega \subset R^3, \\ u = 0, \text{ everywhere in } \partial\Omega, \end{cases} \tag{17}$$

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

After spatial discretization the aim is to solve a pseudo-linear algebraic systems similar to (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.

5 Nonlinear multisplitting method implementation

This section presents the implementation of our multi-splitting method to solve nonlinear stationary systems like nonlinear convection-diffusion problems presented in Section 4. It should be noticed that we do not no difference is made between processors or cores.

The focus was put on solving 3D nonlinear systems of equations involving a single variable which can be formulated as in (1). The well-known Newton iteration method was used to linearize the nonlinear problem. Then the parallel multi-splitting iteration scheme was applied to solve each algebraic linear system issued from the linearization, in such a way that each system is associated to m splittings as shown in (12).

The Newton-multisplitting method was implemented on a simulated parallel platform composed of m blocks that correspond to the m splittings in Formula (12). In this case, each splitting was solved in parallel on a group (or a block) of p processors by using the well-known Krylov iterative method GMRES [29]. The outer iterations of the multi-splitting method (*i.e.*

Algorithm 1: Parallel Nonlinear multi-splitting method performed on a cluster

Output: Solution U_{newt}

1 Set initial solution: $U_{newt} = 1.0$
2 **while** $\|U^{new}_{multi}\|_2 \geq \varepsilon_{Newton}$ **do**
3 Reset the initial local solution U^{old}_{multi} to an arbitrary value
4 Update global right-hand side: Formula (14)
5 Update local sparse matrix A_{multi}: Formula (11)
6 **while** $\|U^{old}_{multi} - U^{new}_{multi}\|_\infty \geq \varepsilon_{Multisplitting}$ **do**
7 Compute local right-hand side: B_{multi}: Formula (13)
8 Parallel GMRES to solve: $A_{multi} \times U^{new}_{multi} = B_{multi}$: Formula (12)
9 Exchange local shared values of U^{new}_{multi} with neighbor blocks
10 $U^{old}_{multi} = U^{new}_{multi}$
11 **end**
12 Compute solution: $U_{newt} = U_{newt} + U^{new}_{multi}$: Formula (10)
13 **end**

intra-blocks communications) are either synchronous or asynchronous, but the inner iterations (GMRES iterations) are synchronous.

Algorithm 1 presents the main key-points of our multi-splitting method executed in parallel to solve nonlinear systems. All variables are local to all processors which are gathered in m blocks of p processors or cores.

The algorithm uses the Newton iteration to linearize the nonlinear system to be solved (lines from 2 to 13). From line 6 to line 11, each linear subsystem issued from the linearlization is solved in parallel using a multi-splitting method. First the local right-hand side B_{multi}, corresponding to B_l involved in the formula (12) and defined by (13) (see line 7) is computed, then the GMRES method is applied in parallel to solve the subsystem like (12) by a block of p processors (line 8). The GMRES iteration represents the inner iteration of the multi-splitting method. At each outer iteration, blocks exchange the data of their local solution U_{multi}, corresponding in fact to $\delta U_l^{l,(i)}$ involved in the formula (12), shared with their corresponding neighbors (see line 9). The solution of the nonlinear system is updated at each Newton iteration (line 12).

The outer iterations of the multi-splitting method can be either synchronous or asynchronous. In the synchronous version, the global convergence of the multi-splitting method is detected when the value of U_{multi} is stabilized corresponding to the following stopping test

$$\|U^{old}_{multi} - U^{new}_{multi}\|_\infty < \varepsilon_{Multisplitting} \tag{18}$$

where $\varepsilon_{Multisplitting}$ is the tolerance threshold for the computation of U_{multi}. However in the asynchronous version, the global convergence is detected when all blocks have locally converged. The convergence detection implemented was implemented as in [30].

6 Experiments

In the following, the conducted experiments are described. The problem considered is described in section 4. Each dimension of the 3D problem is discretized in 150 elements using

Nb. Proc.	Mode	Exec. Times (in s)	Asyn. Gain
2×8	Sync	127.9	
2×8	Async	111.86	1.14
2×16	Sync	61.19	
2×16	Async	48.51	1.26
2×32	Sync	25.79	
2×32	Async	21.93	1.17

Table 1: Execution times of both synchronous and asynchronous iteration modes of the 3D problem of size 150^3.

a finite difference scheme. So there are 150^3 elements to take into consideration. In all the experiments, the following parameters have been chosen: $\varepsilon_{Multipsplitting} = 1 \times 10^{-8}$ and $\varepsilon_{Newton} = 1 \times 10^{-4}$. Up to 64 cores were used to conduct our experiments. Experiments have been achieved on the mesocentre of the University of Franche-Comté. Machines are composed of Xeon(R) CPU E5-2640 v3 @ 2.60GHz processors. They are linked with an Infiniband network. The code is parallelized with MPI. As the mesocentre is used by many users, jobs are run automatically by the scheduler and users cannot have any control on the cores used.

In Table 1, execution times of the synchronous version and the asynchronous version are reported using respectively 16, 32 and 64 cores. The number of blocks was fixed to 2 because it was observed that this would lead to obtain the best performances possible. So in this table, 2×8 represents a case with 16 cores using 2 blocks with 8 cores. It can be seen that the asynchronous version is always faster than the synchronous version. Moreover, each solution computed with the synchronous and asynchronous mode has been compared with the solution computed with a standard solver in order to be able to compute the error. For all the experiments, the error with the max norm is between 1×10^{-7} and 1×10^{-8}. As the machine used only enables one to submit parallel jobs with 64 cores, larger scale experiments could not be run.

7 Conclusion

In the present study, a mixed method combining parallel asynchronous method as an outer iteration with the Krylov method for the solution of diagonal subproblems, was presented. Such a calculation method has been used for the solution of univalued pseudo - linear stationary problems and implemented in a cluster. In future work we will consider the use of such mixed methods for the solution of multivalued pseudo - linear stationary problems. These problems arise in boundary value problems where the solution is subjected to some constraints. We will also implement the proposed mixed method on grid architecture for the parallel solution of univalued or multivalued pseudo - linear stationary problems.

Acknowledgment

The authors wish to thank very much Doctor Thierry Garcia for helpful discussions and exchanges concerning the present study. Also we would like to thank the Mesocentre of Franche-Comté for the computing facilities.

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] Z.Z. Bai, V. Migallón, J. Penadés, D.B. 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(6–8), 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] M. Chau, A. Laouar, T. Garcia, P. Spitéri, “Grid solution of problem with unilateral constraints”, Numerical Algorithms, Springer-Verlag, 75 - 4, 879 - 908, 2017.
[25] J.-C. Miellou, P. Spiteri, “Un critère de convergence pour des méthodes générales de point fixe”, M2AN, 19, 645-669, 1985.
[26] D.J. Evans, W. Deren, “An asynchronous parallel algorithm for solving a class of nonlinear simultaneous equations”, Parallel Computing, 17, 165–180, 1991.
[27] J. Ortega, W. Rheinboldt, “Iterative Solution of Nonlinear Equations in Several Variables”, Academic Press, New York, 1970.
[28] J. Peinado, A.M. Vidal, “A parallel Newton-GMRES algorithm for solving large scale nonlinear systems”, in Palma J.M.L.M. et all, Vecpar 2002, Lecture Notes in Computer Science, 2565, 328 - 342, 2002
[29] 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.
[30] 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.