Parallel implementation of preconditioned iterative schemes by means of overlapping decomposition

Y. Notay [1]

Service de Métrologie Nucléaire
Université Libre de Bruxelles (C.P. 165)
50, Av. F.D. Roosevelt, B-1050 Brussels, Belgium.
email : ynotay@ulb.ac.be

Report GANMN 95-04

September 1995

[1]Supported by the "Fonds National de la Recherche Scientifique", Chercheur qualifié.

Abstract

We consider iterative solvers for large linear systems, and develop a theoretical analysis for parallel implementation techniques which resort to overlapping decompositions, i.e. in which some unknowns are replicated on two or more processors to facilitate the parallelization.

We investigate the conditions under which a given sequential preconditioner can be implemented by means of purely local computations and basic communication routines similar to those needed when no preconditioning is used. General algebraic results are given which are applicable to point and blockwise approximate factorization or SSOR preconditioners.

keywords: iterative methods for linear systems, parallel computing, preconditioning.

AMS classification : 65F10, 65B99, 65N20.

1 Introduction

To program an application on a parallel computer requires in general much more effort than programming it sequentially. Even if the algorithm has some natural parallelism well fitted to the considered architecture, some technical work is still needed to avoid any bottleneck and group the communications as much as possible to decrease the number of synchronization points.

As part of this art, it is a common practice to try to alleviate the communication cost by replicating some variables and repeating identical calculations in parallel. A careful dependency analysis is then needed to see whenever a given copy of a given variable has the right value or not.

Here, we consider the parallelization of linear iterative solvers, and develop mathematical tools which allow the design of easy-to-implement algorithm with minimal or close to minimal communication cost per iteration.

We follow the idea of replicating some variables on several processors, or, in more technical words, we consider the use of overlapping decompositions. But, as an essential condition to make things transparent to the user, we want to avoid any discrimination between the different copies of a same variable, and, as far as possible, between these replicated vaiables and those assigned to only one processor.

In this view, we propose to apply, in some sense, the iterative scheme to an augmented system whose variable set corresponds to the direct sum of the local variable sets, and in which replicated variables are therefore represented several times.

We first show that one may follow this approach and keep nevertheless a strict mathematical equivalence with the corresponding process applied to the original system. Besides the communications required to compute the iteration parameters and control convergence (which are similar as in any implementation), the only communication routine needed for the unpreconditioned version is a basic piece of program which exchanges the value at each replicated variable for the sum of local value and the values which, on the other processors, correspond to the same original variable.

Our main result is then a serie of algebraic theorems which give sufficient conditions for the *preconditioned* version being implemented with the same type of simple communication routines. These remain the same whatever the specific preconditioner, and to implement any of them which matches our assumptions requires only to supply routines for purely local matrix vector operations.

General results are given for explicit and implicit preconditioners in factorized form, allowing easy and efficient parallelization of SSOR and approximate factorization type preconditioners.

The stated sufficient conditions are actually restrictions on the nonzero pattern of the factors involved in the preconditioner. They are easy to check either by hand (matrices with regular structure) or by software. For incomplete factorization of matrices corresponding to a five point stencil, a reordering of the unknowns is generally sufficient to meet these requirements. Our results offer then a nice implementation scheme for the parallel ordering strategy proposed in [9] and generalized in [8], and which roughly consists in numbering separately the "interior nodes" (those which are not replicated) and the "interface nodes" (which form the overlapping part of the decomposition).

Very interesting results were obtained in [7, 10, 11] for this method on both MIMD and SIMD computers. In this specific case as well as in the more general applications outlined in Section 4, our results appear as an efficient tool to analyse what is to handle in a sequential preconditioner to put is in a nice parallelizable form; since the handling is performed at the level of the global preconditioning matrix, any knowledge on sequential preconditioners can be used to minimize its impact on the convergence rate. This approach represents therefore a worthwhile alternative to the empirical troncation of the preconditioners sometimes proposed in the literature in spite of its dramatic and unpredictible effects on the convergence rate, see a.o. [12] for an example in the incomplete factorization context.

The remainder of this paper is organized as follows: in Section 2, we explain how to apply to iterative scheme on the augmented system, derive the general form for the preconditioners, and outline our main results; these are proved in Section 3, whereas some existing and potential applications are outlined in Section 4; finally, in Section 5, we consider the case one applies our results to parallelize incomplete factorization preconditioners, and show how one can then also parallelize the computation of the triangular factors.

2 Parallelization of linear iterative schemes

Consider the following standard version of the conjugate gradient algorithm; here A is the system matrix, b the right hand side, $x^{(0)}$ the initial approximation and B the preconditioner.

$$
\begin{array}{lcl}
r^{(0)} = b - Au^{(0)} & & \\
\textit{For } k = 0, 1, \ldots & & \textit{until convergence :} \\
\qquad g^{(k)} & = & B^{-1}\, r^{(k)} \\
\qquad \alpha_k & = & (g^{(k)}, r^{(k)}) \;;\; \delta_k = \alpha_k / \alpha_{k-1} \; (\delta_0 = 0) \\
\qquad d^{(k)} & = & g^{(k)} + \delta_k\, d^{(k-1)} \; (d^{(0)} = g^{(0)}) \\
\qquad t^{(k)} & = & A\, d^{(k)} \\
\qquad \gamma_k & = & (t^{(k)}, d^{(k)}) \;;\; \beta_k = \alpha_k / \gamma_k \\
\qquad u^{(k+1)} & = & u^{(k)} + \beta_k\, d^{(k)} \\
\qquad r^{(k+1)} & = & r^{(k)} - \beta_k\, t^{(k)}
\end{array}
$$

We discuss this algorithm, but a similar discussion could be developed for many other iterative schemes as well, including methods for non symmetric or indefinite systems.

Here, we want to distribute the unknowns on the different processors (or independent tasks) in such a way that some unknowns may be represented on several processors. The nodes associated to these unknowns will be called interface nodes.

We need then to assume some relation between a vector defined on the global grid and its local representations on the different processors. Here, two kinds of representations will be considered. A vector will be said *distributed* whenever the true value at some node is recovered by summing the values at the different local representations of that node, whereas a vector will be said *replicated* whenever the value at any local representation of a node is a copy of the true value at that node.

Our starting point for the parallelization of the algorithm above is to let $b, r^{(k)}, t^{(k)}$ be distributed and $g^{(k)}, u^{(k)}, d^{(k)}$ replicated.

One easily checks that these choices are consistent if the multiplication by the system matrix is implemented in such a way that it takes a replicated vector as input to produce a distributed vector as output, whereas, conversely, the preconditioning step has to accept a distributed input vector and produce a replicated output vector.

With this choice, as easily checked (a formal proof will be given below), the inner products α_k and γ_k, because they both involve a replicated and a distributed vector, are recovered by summing on all nodes without discriminating interface nodes. Another desirable feature is that the approximate solution vector u^k is replicated, i.e. one has direct access on each processor to the computed solution at any node locally represented.

Now, to go further in our analyis, we need to introduce some additional notation and definitions.

Throughout this paper, we assume that there are N processors (or independent tasks), and denote P_p the set of nodes represented on processor p. $(P_p)_{p=1,\dots,N}$ is thus a covering of $[1,n]$, that is

$$P_p \subset [1,n] \qquad\qquad \text{for } p=1,\dots,N,$$

and

$$[1,n] = \bigcup_{p=1}^{N} P_p \ .$$

We further note n_p the number of nodes represented on processor p:

$$n_p = \# P_p \qquad\qquad \text{for } p=1,\dots,N.$$

To these "local node sets" is associated an extended node set $[1,\hat{n}]$, where

$$\hat{n} = \sum_{p=1}^{N} n_p \ .$$

A natural partitioning of this extended set is $\left(\hat{P}_p\right)_{p=1,\dots,N}$, where

$$\hat{P}_p = \left[\sum_{q=1}^{p-1} n_q + 1 \ , \ \sum_{q=1}^{p} n_q \right]$$

is the local node set (or local grid) for processor p. Then, each node in $[1,n]$ has one or more correspondants in $[1,\hat{n}]$, while each node in $[1,\hat{n}]$ has a unique correspondant in $[1,n]$. A mathematical representation of this correspondance relation is obtained with the $\hat{n}\times n$ boolean matrix $\mathcal{L}$ defined by

$$\mathcal{L}_{\hat{\imath}i} = \begin{cases} 1 & \text{if } \hat{\imath}\in\hat{P}_p \text{ and } i \text{ is the } \left(\hat{\imath}-\sum_{q=1}^{p-1} n_q\right)^{th} \text{ node in } P_p \\ 0 & \text{otherwise .}\end{cases}$$

$\mathcal{L}$ has exactly one nonzero per row and at least one nonzero per column and

$$\Sigma = \mathcal{L}\ \mathcal{L}^t$$

is the $\hat{n}\times\hat{n}$ matrix such that $\Sigma_{\hat{\imath}\hat{\jmath}}=1$ if and only if $\hat{\imath}$ and $\hat{\jmath}$ corresponds to a same node in $[1,n]$, with $\Sigma_{\hat{\imath}\hat{\jmath}}=0$ otherwise.

As easily seen a vector $\hat{x}$ in $R^{\hat{n}}$ is a distributed representation of a vector x in R^n if and only if

$$x = \mathcal{L}^t\,\hat{x}$$

and it is a replicated representation if and only if

$$\hat{x} = \mathcal{L}x.$$

Consequently, the matrix Σ transforms the distributed representation of a vector into its replicated representation.

Going back to the conjugate gradient algorithm, the above choice of distributed and replicated representations amounts to assume

$$\begin{aligned}
b &= \mathcal{L}^t\,\hat{b} && (2.1)\\
r^{(k)} &= \mathcal{L}^t\,\hat{r}^{(k)} && (2.2)\\
t^{(k)} &= \mathcal{L}^t\,\hat{t}^{(k)} && (2.3)\\
\hat{g}^{(k)} &= \mathcal{L}\,g^{(k)} && (2.4)\\
\hat{u}^{(k)} &= \mathcal{L}\,u^{(k)} && (2.5)\\
\hat{d}^{(k)} &= \mathcal{L}\,d^{(k)} && (2.6)
\end{aligned}$$

for all $k \geq 0$.

Now, let $\hat{A}$ be an $\hat{n} \times \hat{n}$ matrix such that

$$A = \mathcal{L}^t\;\hat{A}\;\mathcal{L} \qquad (2.7)$$

and let $\hat{B}^{(1)}$ be an $\hat{n} \times \hat{n}$ matrix such that

$$\hat{B}^{(1)} = \mathcal{L}^t\;B^{-1}\;\mathcal{L} \qquad (2.8)$$

(we may not write $\hat{B}^{-1}$ because $\hat{B}^{(1)}$ is not invertible).

It is not difficult to see that a multiplication by $\hat{A}$ is an algebraic formulation for an implementation of the matrix vector product which takes a replicated vector as input to produce a distributed output vector. Indeed, if $\hat{x} = \mathcal{L}x$,

$$Ax = \mathcal{L}^t\,\hat{A}\,\mathcal{L}\,x = \mathcal{L}^t\,(\hat{A}\,\hat{x})\;.$$

Conversely, the multiplication by $\hat{B}^{(1)}$ is an algebraic formulation for an implementation of the preconditioning step which acts on a distributed vector to produce a replicated vector : if $y = \mathcal{L}^t\,\hat{y}$,

$$\hat{B}^{(1)}\,\hat{y} = \mathcal{L}\,B^{-1}\,\mathcal{L}^t\,\hat{y} = \mathcal{L}(B^{-1}y).$$

It follows then from the considerations at the beginning of this section that, instead of the algorithm above, we may execute the same algorithm, but on the extended variable set with system matrix $\hat{A}$, inverse preconditioner $\hat{B}^{(1)}$, right hand side $\hat{b}$ such that $b = \mathcal{L}^t\,\hat{b}$ and initial approximation $\hat{u}^{(0)} = \mathcal{L}\,u^{(0)}$. The generated sequences $\{\hat{r}^{(k)}\}, \{\hat{t}^{(k)}\}, \{\hat{g}^{(k)}\}, \{\hat{u}^{(k)}\}, \{\hat{d}^{(k)}\}$ will satisfy relations (2.2-2.6) with respect to the original sequences.

For a formal proof of this result, it suffices to note that

$$\begin{aligned}
\hat r^{(0)} &\equiv \hat b - \hat A\,\hat u^{(0)} &&= \mathcal{L}^t\left(b - A\,u^{(0)}\right) &&= \mathcal{L}^t\, r^{(0)}\ ,\\
\hat g^{(0)} &\equiv \hat B^{(-1)}\,\hat r^{(0)} &&= \mathcal{L}\left(B^{-1}\, r^{(0)}\right) &&= \mathcal{L}\, g^{(0)}\ ,\\
\hat t^{(0)} &\equiv \hat A\,\hat g^{(0)} &&= \mathcal{L}^t\left(A\, g^{(0)}\right) &&= \mathcal{L}^t\, t^{(0)}\ ,
\end{aligned}$$

Hence (2.2-2.6) hold for $k = 0$. A similar reasoning together with an induction argument proves then that (2.2 - 2.6) hold for all k if the computed α_k and γ_k are equal; but this is true by virtue of the same induction argument since

$$\begin{aligned}
\left(\hat g^{(k)}, \hat r^{(k)}\right) &= \left(\mathcal{L}\, g^{(k)}, \hat r^{(k)}\right) &&= \left(g^{(k)}, \mathcal{L}^t\, \hat r^{(k)}\right) &&= \left(g^{(k)}, r^{(k)}\right)\ ,\\
\left(\hat t^{(k)}, \hat d^{(k)}\right) &= \left(\hat t^{(k)}, \mathcal{L}\, d^{(k)}\right) &&= \left(\mathcal{L}^t \hat t^{(k)}, d^{(k)}\right) &&= \left(t^{(k)}, d^{(k)}\right)\ ,
\end{aligned}$$

Relations (2.2-2.6) justify that we speak of equivalence between both processes, any vector in the original algorithm being uniquely determined from its representation on the extended system (that is from its local representations on the different processors).

Now, we have to investigate (2.7) and (2.8). (2.7) is actually an assembly formula and may be rewritten

$$a_{ij} = \sum_{\substack{\hat\imath \text{ s.t. } \mathcal{L}_{\hat\imath\, i} \neq 0 \\ \hat\jmath \text{ s.t. } \mathcal{L}_{\hat\jmath\, i} \neq 0}} \hat a_{\hat\imath\,\hat\jmath} \tag{2.9}$$

for all i, j. Hence it is not difficult at all to find a matrix $\hat A$ satisfying (2.7).

Moreover, if all nonzero entries a_{ij} in A are such that i and j belong to at least one same processor, one may still satisfy (2.9) when limiting the sum in the right hand side to nodes $\hat\imath, \hat\jmath$ belonging to a same local grid. Considering the block partitioning associated to the partitioning $\left(\hat P_p\right)_{p=1,\ldots,N}$ of $[1, \hat n]$, $\hat A$ is then block diagonal. Consequently, a parallel implementation of

$$\hat t^{(k)} = \hat A\,\hat d^{(k)}$$

requires no communication since the computation of the p^{th} block component $\hat t_p^{(k)}$ of $\hat t^{(k)}$ depends only on the p^{th} block component $\hat d_p^{(k)}$ of $\hat d^{(k)}$:

$$\hat t_p^{(k)} = \hat A_{pp}\,\hat d_p^{(k)}\ , \qquad\qquad p = 1, \ldots, N.$$

An example of rule to determine $\hat A$ is then

$$\hat a_{\hat\imath\,\hat\jmath} = \begin{cases} \frac{a_{ij}}{m_{ij}} & \text{if } \hat\imath\,,\, \hat\jmath \in \hat P_p \text{ for some } p \\ 0 & \text{otherwise}\ , \end{cases}$$

where i, j are the nodes in $[1, n]$ such that one has respectively $\mathcal{L}_{\hat\imath i} = 1$ and $\mathcal{L}_{\hat\jmath j} = 1$, and where

$$m_{ij} = \#\left\{p \in [1, N] \mid i, j \in P_p\right\}$$

is the number of processors sharing i and j.

Of course, any other rule such that (2.9) holds is also worthwhile. For instance, in application to discrete PDEs, one will often obtain directly the desired block components from discretizations executed locally on each processor.

If the matrix vector product is now purely local, it does not mean that we are solving decoupled systems. Indeed, the communication are only delayed to the preconditioning step

$$\hat{g}^{(k)} = \hat{B}^{(1)}\,\hat{r}^{(k)} = \mathcal{L}\,B^{-1}\,\mathcal{L}^t\,\hat{r}^{(k)}$$

which is at first sight very sequential.

However, without preconditioning, it reduces to

$$\hat{g}^{(k)} = \mathcal{L}\,\mathcal{L}^t\,\hat{r}^{(k)} = \Sigma\,\hat{r}^{(k)},$$

which implies only communication between neighbour processors as needed in any parallel implementation of the unpreconditioned conjugated gradient algorithm.

In this paper, we aim at investigating how the *preconditioned* conjugate gradient algorithm can be implemented using the same kind of communications.

In particular, we consider preconditioners of the form

$$B = L\,P^{-1}\,U.$$

and introduce the replicated representation $\hat{L}, \hat{P}$ and $\hat{U}$ of respectively L, P and U, which is defined as the block diagonal $\hat{n} \times \hat{n}$ matrices whose each diagonal block is the restriction to the concerned local node set of the corresponding matrix. A mathematical formulation for that is

$$\begin{aligned} \hat{L} &= \mathrm{diag}_{\mathrm{block}}(\mathcal{L}\,L\,\mathcal{L}^t), \\ \hat{P} &= \mathrm{diag}_{\mathrm{block}}(\mathcal{L}\,P\,\mathcal{L}^t), \\ \hat{U} &= \mathrm{diag}_{\mathrm{block}}(\mathcal{L}\,U\,\mathcal{L}^t), \end{aligned}$$

where $\mathrm{diag}_{\mathrm{block}}(\;.\;)$ denotes the block diagonal part with respect to the partitioning $\left(\hat{P}_p\right)_{p=1,\ldots,N}$ of $[1,\hat{n}]$. Note that if L, P and U are invertible, then $\hat{L}, \hat{P}$ and $\hat{U}$ are also invertible because their determinant is the product of principal minors from the original matrix.

For these preconditioners, we will show in the next section that if, for all $i, j,\; i \neq j$,

$$\exists p \in [1, N] \text{ with } j \in P_p \;\text{ and } i \notin P_p \quad \Rightarrow \quad \ell_{ij} = u_{ji} = p_{ij} = p_{ji} = 0\;, \tag{2.10}$$

then

$$\hat{U}^{-1}\,\hat{P}\,\Sigma\,\hat{L}^{-1} = \hat{U}^{-1}\,\Sigma\,\hat{P}\,\hat{L}^{-1} = \mathcal{L}\,U^{-1}\,P\,L^{-1}\,\mathcal{L}^t\,. \tag{2.11}$$

Assumption (2.10) first requires that nonzero entries in L, P and U only connect nodes belonging to at least one same processor, which is very natural in the context of *overlapping* decompositions. Additionally, it is not permitted to have in L, U^t, P and P^t a connection from i to j if j is an interface node and if i does not belong to all processors on which j is represented.

If L is lower triangular, U upper triangular and P diagonal, this may be managed by ordering interface nodes after the interior nodes. Entries ℓ_{ij} or u_{ji} between an interface node i and an interior node j are then indeed anyway zero. Note however that no permutation is actually required and that, alternatively, one may discard forbidden entries from the preconditioner.

Note also that it is not required to have L and U triangular and P diagonal. Hence, our results may be useful to parallelize block incomplete factorization preconditioner as well, for instance.

Now, our analysis covers more general situations, and if, instead of (2.10) one has, for all $i, j,\ i \neq j$,

$$\exists p \in [1, N] \text{ with } i \in P_p \text{ and } j \notin P_p \quad \Rightarrow \quad \ell_{ij} = u_{ji} = p_{ij} = p_{ji} = 0 \ , \tag{2.12}$$

then, the results in the next section prove that

$$\Sigma\, \hat{U}^{-1}\, \hat{P}\, \Delta \hat{L}^{-1}\, \Sigma = \Sigma\, \hat{U}^{-1}\, \Delta\, \hat{P}\, \hat{L}^{-1}\, \Sigma = \mathcal{L}\, U^{-1}\, P\, L^{-1}\, \mathcal{L}^t \ , \tag{2.13}$$

where Δ is the $\hat{n} \times \hat{n}$ diagonal matrix such that

$$\Delta^{-1}\, e = \Sigma\, e$$

with $e = (1 \dots 1)^t$.

This result is particularly useful when interface nodes are odered first.

Now, considering for instance the ordering strategy proposed in [8], it is interesting to address the case where part of the interface nodes are ordered first and the remaining lastly, i.e. a mix between the situation covered by (2.10) and that covered by (2.12). This requires to introduce some more notation to be able to distinguish between both type of boundaries.

In this view, each pair of processors sharing some node will be labelled either $'f'$, $'\ell'$ or $'o\,'$:

$$P_p \cap P_q \neq \emptyset \Rightarrow label(p,q) = 'f\,' \text{ or } label(p,q) = '\ell\,' \text{ or } label(p,q) = '\,o\,' \ ,$$

with, by assumption

$$label(p,q) = label(q,p) \qquad \text{for all } p,\ q.$$

There is some freedom when choosing the labels, and, roughly speaking, one will give the label $'f\,'$ whenever the corresponding interface is to be ordered first, and the label $'\ell\,'$ when it is to be ordered lastly. Labels of type $'o'$ will concern only non principal interfaces (that is those limited to a corner or a ridge in 3D), and are introduced for technical reasons, in connection with the need to satisfy the two following properties. These form the necessary and sufficient conditions for the chosen labelling to be valid.

Property 1

For all $p, q, r \in [1, N]$ such that $P_p \cap P_q \cap P_r \neq \emptyset$:

$$\left.\begin{array}{l} label(p,q) = 'f\,' \\ label(q,r) = 'f\,' \end{array}\right\} \Rightarrow label(p,r) = 'f\,'$$

and

$$\left.\begin{array}{l} label(p,q) = '\ell\,' \\ label(q,r) = '\ell\,' \end{array}\right\} \Rightarrow label(p,r) = '\ell\,'$$

Property 2

For all $p, q \in [1, N]$ such that $label(p,q) = 'o'$, and for all $i \in P_p \cap P_q$, there exists some $r, s \in [1, N]$ such that $i \in P_r \cap P_s$ with

$$label(p,r) = 'f' \ , \quad label(r,q) = '\ell'$$

and

$$label(p,s) = '\ell' \ , \quad label(s,q) = 'f' \ .$$

A common policy to satisfy these properties is to choose the label (either $'f'$ or $'\ell'$) of pairs of processors sharing a principal boundary (that is a line of node in 2D and a plane of node in 3D), and then let the remaining of the labels be determined by the need to satisfy Property 1, which also implies

$$\left.\begin{array}{l} label(p,q) = 'f' \\ label(q,r) = '\ell' \end{array}\right\} \Rightarrow label(p,r) = 'o'$$

for any p, q, r such that $P_p \cap P_p \cap P_r \neq \emptyset$ (any other choice for $label(p,r)$ would imply a contradiction with Property 1). If no contradiction is met, it remains only to check Property 2, which should raise no difficulty for regular processor grids. We refer to [7] for an example in two and three dimensions.

Thanks to Properties 1 and 2, it is proved in the next section that

$$\Sigma = \Sigma_\ell \, \Sigma_f$$

where

$$(\Sigma_f)_{\hat{\imath}\hat{\jmath}} = \begin{cases} \Sigma_{\hat{\imath}\hat{\jmath}} & \text{if } \hat{\imath} \in P_p \, , \, \hat{\jmath} \in P_q \text{ with } p = q \\ & \quad \text{or } P_p \cap P_q \neq \emptyset \text{ and } label(p,q) = 'f' \\ 0 & \text{otherwise} \end{cases}$$

and

$$(\Sigma_\ell)_{\hat{\imath}\hat{\jmath}} = \begin{cases} \Sigma_{\hat{\imath}\hat{\jmath}} & \text{if } \hat{\imath} \in P_p \, , \, \hat{\jmath} \in P_q \text{ with } p = q \\ & \quad \text{or } P_p \cap P_q \neq \emptyset \text{ and } label(p,q) = '\ell' \\ 0 & \text{otherwise .} \end{cases}$$

Thus, Σ_f and Σ_ℓ are the matrices deduced from Σ by setting to zero the offdiagonal blocks corresponding to the pairs of processors whose label is not equal to respectively $'f'$ or $'\ell'$.

With this formalism, the above results extend as follows: if, for all $i, j, \ i \neq j$ such that $\ell_{ij} \neq 0$ or $u_{ji} \neq 0$ or $p_{ij} \neq 0$ or $p_{ji} \neq 0$, one has

$$\begin{array}{ll} \text{(a)} & i, j \in P_p \text{ for some } p \in [1, N] \\ \text{(b)} & \left.\begin{array}{l} j \in P_p \cap P_q \text{ for some } p, q \in [1, N] \text{ with } p \neq q \\ label(p,q) = '\ell' \end{array}\right\} \Rightarrow i \in P_p \cap P_q \\ \text{(c)} & \left.\begin{array}{l} i \in P_p \cap P_q \text{ for some } p, q \in [1, N] \text{ with } p \neq q \\ label(p,q) = 'f' \end{array}\right\} \Rightarrow j \in P_p \cap P_q \end{array}$$

then

$$\Sigma_f \, \hat{U}^{-1} \, \hat{P} \, \Delta_f \Sigma_\ell \, \hat{L}^{-1} \, \Sigma_f = \Sigma_f \, \hat{U}^{-1} \, \Delta_f \, \Sigma_\ell \, \hat{P} \, \hat{L}^{-1} \, \Sigma_f = \mathcal{L} \, \hat{U}^{-1} \, P \, L^{-1} \, \mathcal{L}^t \ , \tag{2.14}$$

where Δ_f is the diagonal matrix such that

$$\Delta_f^{-1}\, e = \Sigma_f\, e \ .$$

This will be proved in the next Section. Note that the two results given above are particular cases of (2.14), the first one corresponding to the case where $label(\,p, q) = {}'\ell'$ for all p, q and the second one to the case where $label(\,p, q) = {}'f\,'$ for all p, q.

Remark 1

For preconditioners $B = U^t\, P^{-1}\, U$, since

$$\alpha_k = (r_k, g_k) = (r_k, B^{-1} r_k) = \Big((U^{-t}\, r_k),\ P(U^{-t}\, r_k)\Big) \ ,$$

it is proposed in [2] to overlap the communication needed to update α_k with the computation of $U^{-1}\, y_k$ where $y_k = P\, U^{-t}\, r_k$.

This trick may be applied within the present framework since

$$\begin{aligned}(\hat r_k, \hat g_k) &= (\hat r_k\,,\ \Sigma_f\, \hat U^{-1}\, \hat P\, \Delta_f\, \Sigma_\ell\, \hat U^{-t}\, \Sigma_f\, \hat r_k) \\ &= \Big((\hat U^{-t}\, \Sigma_f\, \hat r_k)\,,\ \hat P\, \Delta_f\, \Sigma_\ell\, \Big(\hat U^{-t}\, \Sigma_f\, \hat r_k\Big)\Big) \ .\end{aligned}$$

Remark 2

Considering incomplete factorization preconditioners with parallel orderings, it is shown in [8] that the scalability of the algorithm is improved when adding a coarse grid correction to the preconditioner, that is when using

$$B^{-1} = U^{-1}\, P\, L^{-1} \ + \ V\,(\,V^t\, A\, V)^{-1}\, \hat V^t \ ,$$

where $V = [v_1, \ldots, v_m]$ is a collection of a few vectors.

This is easily implemented in the present framework since, if (2.14) holds, then letting $\hat V = \mathcal{L}\, V = [\mathcal{L}\, v_1, \ldots, \mathcal{L}\, v_m]$ be the corresponding collection of replicated vectors, one has

$$\Sigma_f\, \hat U^{-1}\, \hat P\, \Delta_f\, \Sigma_f\, \hat U^{-t}\, \Sigma_f \ + \ \hat V\ \Big(\hat V^t\, \hat A\, \hat V\Big)^{-1}\ \hat V^t = \mathcal{L}\, B^{-1}\, \mathcal{L}^t \ ,$$

because

$$\hat V\,(\hat V^t\, \hat A\, \hat V)^{-1}\, \hat V^t = \mathcal{L}\, V\, \Big(V^t\, \mathcal{L}^t\, \hat A\, \mathcal{L}\, V\Big)^{-1}\ V^t\, \mathcal{L}^t = \mathcal{L}\, V (V^t\, A\, V)^{-1}\, V^t\, \mathcal{L}^t$$

The additional term requires global communication but, if one uses the technique recalled in Remark 1, this communication may be grouped with that needed for the computation of α_k and overlapped with the computations related to $\hat U^{-1}$.

3 Theoretical results

In general, most interface nodes i will be such that the label of any pair of processor sharing them is either $'f'$ or $'\ell'$. However, except in the cases where all labels are set to either $'f'$ or $'\ell'$ and in some special situations like strip partitioning, there will be some nodes which belong simultaneously to both types of boundaries.

The following lemma, as well as its corollary, are useful to better understand the consequences of our assumptions (Properties 1 and 2) for these nodes.

Lemma 1
For any $i \in [1,n]$, the relation whose set of couples are

$$R_i^{(f)} = \{(p,q) \in [1,N] \times [1,N] \mid i \in P_p \cap P_q \text{ and } label(p,q) = 'f'\}$$

induces a partitioning of the subset of processors to which i belongs, which is such that any class of the partitioning contains the same number of elements. Similarly, the relation whose set of couples are

$$R_i^{(\ell)} = \{(p,q) \in [1,N] \times [1,N] \mid i \in P_p \cap P_q \text{ and } label(p,q) = '\ell'\}$$

induces another partitioning of the same subset whose each class has the same number of elements. In addition, the number of elements in a class of the partitioning induced by $R_i^{(f)}$ (resp. $R_i^{(\ell)}$) is equal to the number of classes in the partitioning induced by $R_i^{(\ell)}$ (resp. $R_i^{(f)}$).

Proof. Consider two classes (C_1, C_2) of the partitioning induced by $R_i^{(f)}$. For all $p \in C_1, q \in C_2$, $label(p,q)$ is either $'\ell'$ or $'o'$. By Property 2, there is at least one $q_1 \in C_2$ such that $label(p,q_1) = '\ell'$. This q_1 is unique because $label(p,q_1) = '\ell' = label(p,q_2)$ with $q_1 \neq q_2$ would imply (Property 1) $label(q_1,q_2) = '\ell'$, while q_1, $q_2 \in C_2$ implies $label(q_1,q_2) = 'f'$.

Conversely, for all $q \in C_2$ there is a unique $p_1 \in C_1$ such that $label(p_1,q) = '\ell'$. Hence, there is a bijection from C_1 to C_2, showing that $\#C_1 = \#C_2$. The proof that the classes of the partitioning induced by $R_i^{(\ell)}$ have same number of element is similar.

If we now let $C_1, \ldots, C_m$ be the classes of the partitioning induced by $R_i^{(f)}$, for all $p_1 \in C_1$, there is exactly one p_i in each C_i such that $label(p_1,p_i) = '\ell'$, showing that m is the number of element in a class of the partitioning induced by $R_i^{(\ell)}$. □

Corollary 1
For all $i \in [1,n]$, let m_i be the number of processors to which i belongs and $m_i^{(f)}$, $m_i^{(\ell)}$ the number of elements in a class of the partitioning induced by respectively $R_i^{(f)}$ and $R_i^{(\ell)}$.

One has

$$m_i = m_i^{(f)} \ . \ m_i^{(\ell)}$$

and, for all $\hat{\imath} \in [1, \hat{n}]$ *such that* $\mathcal{L}_{\hat{\imath}\, i} = 1$

$$\begin{array}{lllll}
(\Sigma\, e)_{\hat{\imath}} & = & m_i & = & 1 \;+\; \#\{q \neq p \mid \exists \hat{k} \in \hat{P}_q \;\, s.t.\;\, \Sigma_{\hat{\imath}\,\hat{k}} = 1 \;\} \\
(\Sigma_f\, e)_{\hat{\imath}} & = & m_i^{(f)} & = & 1 \;+\; \#\left\{q \neq p \mid \exists \hat{k} \in \hat{P}_q \;\, s.t.\;\, \Sigma_{\hat{\imath}\,\hat{k}} = 1 \;\, with\ label\ \, (p,q) = {}'f\,' \right\} \\
(\Sigma_\ell\, e)_{\hat{\imath}} & = & m_i^{(\ell)} & = & 1 \;+\; \#\left\{q \neq p \mid \exists \hat{k} \in \hat{P}_q \;\, s.t.\;\, \Sigma_{\hat{\imath}\,\hat{k}} = 1 \;\, with\ label\ \, (p,q) = {}'\ell\,' \right\}
\end{array}$$

Proof. Straightforward. $\square$

The remaining of this section is organized as follows. First we prove (Theorem 1) that, when an $n \times n$ matrix C is such that requirements (a), (b), (c) in Section 2 are satisfied for all i, j , $i \neq j$ such that $c_{i\,j} \neq 0$, then, the $\hat{n} \times \hat{n}$ matrix $\hat{C} = \mathrm{diag}_{\mathrm{block}}(\mathcal{L} C \mathcal{L}^t)$ satisfies

$$\Sigma_f\, \Delta_f\, \hat{C}\, \Sigma_f = \hat{C}\, \Sigma_f \tag{3.15}$$

and

$$\Sigma_\ell\, \hat{C}\, \Sigma_f = \mathcal{L} C \mathcal{L}^t \ . \tag{3.16}$$

With $C = I$, this shows in particular that

$$\Sigma_f\, \Delta_f\, \Sigma_f = \Sigma_f \tag{3.17}$$

and

$$\Sigma_\ell\, \Sigma_f = \mathcal{L} \mathcal{L}^t \ . \tag{3.18}$$

Next, we prove (Theorem 2) that, if $(\hat{C}, C)$ is any couple of invertible matrices satisfying (3.15), (3.16), then the couple of their respective inverse $\left(\hat{C}^{-1}, C^{-1}\right)$ also satisfies (3.15) and (3.16). Furthermore, we show (Theorem3) that if $(\hat{C}_1, C_1)$ and $(\hat{C}_2, C_2)$ both satisfy (3.15), (3.16), then their products $\left(\hat{C} = \hat{C}_1\, \hat{C}_2\, ,\; C = C_1\, C_2\right)$ still satisfies (3.15) and (3.16).

Finally, we prove (Theorem 4) that if $(\hat{C}_1, C_1)$ and $(\hat{C}_2, C_2)$ both satisfies (3.15), (3.16), then

$$\Sigma_f \hat{C}_1^t \Delta_f\, \Sigma_\ell \hat{C}_2^t\, \Sigma_f = \mathcal{L}\, C_1^t\, C_2\, \mathcal{L}^t \ .$$

It is trivial to check that the combined application of these four theorems proves in particular the results stated in the preceding section.

In theorem 1, any notation not explicitly referred to is assumed to be defined as in Section 2 and satisfying all basic requirements stated there. Theorems $2, 3$ and 4 hold under the weaker assumption that $\mathcal{L}$ is an $\hat{n} \times n$ matrix of rank n and that Σ_f, Δ_f and Σ_ℓ are symmetric $\hat{n} \times \hat{n}$ matrices satisfying (3.17), (3.18).

Theorem 1

Let C *be an* $n \times n$ *matrix and* $\hat{C} = diag_{\mathrm{block}}(\mathcal{L}\, C\, \mathcal{L}^t)$. *If, for all* i, j *such that* $c_{i\,j} \neq 0$,

$$\begin{array}{lll} (a) & \exists\, p \in [1,N] \textit{ such that } i,\, j \in P_p \ , & \\ (b) & \left.\begin{array}{l} j \in P_p \cap P_q \textit{ for some } p,q \in [1,N] \textit{ with } p \neq q \\ \textit{label}\,(p,q) = {}'\ell{}' \end{array}\right\} & \Rightarrow i \in P_p \cap P_q \ , \\ (c) & \left.\begin{array}{l} i \in P_p \cap P_q \textit{ for some } p,q \in [1,N] \textit{ with } p \neq q \\ \textit{label}\,(p,q) = {}'f{}' \end{array}\right\} & \Rightarrow j \in P_p \cap P_q \ , \end{array}$$

then

$$(1) \quad \Sigma_f\, \Delta_f\, \hat{C}\, \Sigma_f = \hat{C}\, \Sigma_f$$

and

$$(2) \quad \Sigma_\ell\, \hat{C}\, \Sigma_f = \mathcal{L}\, C\, \mathcal{L}^t \quad .$$

Proof. Here, we shall refer block entries of $\hat{n} \times \hat{n}$ matrix, i.e. for any $\hat{n} \times \hat{n}$ matrice G, $(G)_{p\,q}$ denotes the block component connecting the nodes in $\hat{P}_p$ to the nodes in $\hat{P}_q$. When a scalar entry of such a block component is needed, it will be noted $(G_{p\,q})_{\hat{\imath}\,\hat{\jmath}}$, notation which implies $\hat{\imath} \in \hat{P}_q$ and $\hat{\jmath} \in \hat{P}_q$. For any $\hat{\imath} \in [1, \hat{n}]$, we shall also use i *without any further comment* to denote the corresponding node in $[1, n]$, i.e. the node such that $\mathcal{L}_{\hat{\imath}\, i} = 1$.

To prove (1), we first note that

$$\Sigma_f\, \Delta_f = \Delta_f\, \Sigma_f \ ,$$

because, with Corollary 1,

$$(\Sigma_f\, \Delta_f)_{\hat{\imath}\,\hat{\jmath}} = \frac{(\Sigma_f)_{\hat{\imath}\,\hat{\jmath}}}{m_j^{(f)}}$$

and

$$(\Delta_f\, \Sigma_f)_{\hat{\imath}\,\hat{\jmath}} = \frac{(\Sigma_f)_{\hat{\imath}\,\hat{\jmath}}}{m_i^{(f)}}$$

whereas $\Sigma_{\hat{\imath}\,\hat{\jmath}} \neq 0$ implies $i = j$ and therefore $m_i^{(f)} = m_j^{(f)}$.

On the other hand, since $\hat{C}$ is block diagonal,

$$\left(\hat{C}\, \Sigma_f\right)_{pq} = C_{p\,p}\ (\Sigma_f)_{p\,q} \quad .$$

Hence, with $(C_{pp})_{\hat{\imath}\,\hat{\jmath}} = c_{i\,j}$ and

$$\left((\Sigma_f)_{pq}\right)_{\hat{k}\,\hat{\jmath}} = \begin{cases} 1 & \text{if } k = j \text{ and either } p = q \ \text{ or } \ \textit{label}(p,q) = {}'f{}' \\ 0 & \text{otherwise ,} \end{cases}$$

one obtains

$$\begin{aligned} \left(\left(\hat{C}\, \Sigma_f\right)_{pq}\right)_{\hat{\imath}\,\hat{\jmath}} &= \sum_{\hat{k} \in \hat{P}_p} (C_{pp})_{\hat{\imath}\,\hat{k}}\ \left((\Sigma_f)_{pq}\right)_{\hat{k}\,\hat{\jmath}} \\ &= \begin{cases} c_{ij} & \text{if } p = q \text{ or } p \neq q \\ & \text{with } \ \textit{label}(p,q) = {}'f{}' \text{ and } j \in P_p \cap P_q \\ 0 & \text{otherwise ,} \end{cases} \end{aligned}$$

(only one term in $\sum_{\hat{k}\in\hat{P}_p}$ may be nonzero, and there is such a term only if $j \in P_p$ since $\left((\Sigma_f)_{pq}\right)_{\hat{k}\,\hat{\jmath}} \neq 0$ requires $k = j$).

Next, using a similar reasoning

$$\begin{aligned}
\left(\left(\Sigma_f\,\hat{C}\,\Sigma_f\right)_{pp}\right)_{\hat{\imath}\,\hat{\jmath}} &= \left(\left(\hat{C}\,\Sigma_f\right)_{pp}\right)_{\hat{\imath}\,\hat{\jmath}} + \sum_{\substack{q\neq p \\ label(p,q)='f'}} \ \sum_{\hat{k}\in\hat{P}_q} \left((\Sigma_f)_{pq}\right)_{\hat{\imath}\,\hat{k}} \left(\left(\hat{C}\,\Sigma_f\right)_{pq}\right)_{\hat{k}\,\hat{\jmath}} \\
&= c_{i\,j}\ .\ \gamma_{\hat{\imath}\,\hat{\jmath}}
\end{aligned}$$

where

$$\gamma_{\hat{\imath}\,\hat{\jmath}} = 1 + \#\{q \neq p \mid label(p,q) = 'f' \text{ and } i,j \in P_p \cap P_q\}\ .$$

But, by assumption (b), $c_{ij} \neq 0$ implies $j \in P_p \cap P_q$ for any q such that $label(p,q) = 'f'$ and $i \in P_q$. Therefore, $\gamma_{\hat{\imath}\,\hat{\jmath}} = m_i^{(f)}$, whence

$$\begin{aligned}
\left(\left(\Delta_f\Sigma_f\,\hat{C}\,\Sigma_f\right)_{pp}\right)_{\hat{\imath}\,\hat{\jmath}} &= \left(\left(\Sigma_f\,\Delta_f\hat{C}\,\Sigma_f\right)_{pq}\right)_{\hat{\imath}\,\hat{\jmath}} \\
&= c_{ij} = \left(\left(\hat{C}\,\Sigma_f\right)_{pq}\right)_{\hat{\imath}\,\hat{\jmath}}\ .
\end{aligned}$$

Similarly, we find, for $p \neq q$

$$\begin{aligned}
\left(\left(\Sigma_f\,\hat{C}\,\Sigma_f\right)_{pq}\right)_{\hat{\imath}\,\hat{\jmath}} &= \left(\left(\hat{C}\,\Sigma_f\right)_{pq}\right)_{\hat{\imath}\,\hat{\jmath}} \\
&+ \left(\left(\Sigma_f\,\hat{C}\right)_{pq}\right)_{\hat{\imath}\,\hat{\jmath}} \\
&+ \sum_{\substack{r \\ r\neq p \\ r\neq q}} \ \sum_{\hat{k}\in\hat{P}_r} \left((\Sigma_f)_{pr}\right)_{\hat{\imath}\,\hat{k}} \left(\left(\hat{C}\,\Sigma_f\right)_{rq}\right)_{\hat{k}\,\hat{\jmath}} \\
&= c_{ij}\ \left(\gamma^{(1)}_{\hat{\imath}\,\hat{\jmath}} + \gamma^{(2)}_{\hat{\imath}\,\hat{\jmath}} + \gamma^{(3)}_{\hat{\imath}\,\hat{\jmath}}\right)
\end{aligned}$$

where

$$\gamma^{(1)}_{\hat{\imath}\,\hat{\jmath}} = \begin{cases} 1 & \text{if } j \in P_p \cap P_q \text{ and } label(p,q) = 'f' \\ 0 & \text{otherwise} \end{cases}$$

$$\gamma^{(2)}_{\hat{\imath}\,\hat{\jmath}} = \begin{cases} 1 & \text{if } i \in P_p \cap P_q \text{ and } label(p,q) = 'f' \\ 0 & \text{otherwise} \end{cases}$$

$$\gamma^{(3)}_{\hat{\imath}\,\hat{\jmath}} = \#\{r \neq p,q \mid label(p,r) = label(r,q) = 'f' \text{ and } i,\ j \in P_r\}\ .$$

But, for all $r \neq p$, $i \in P_p \cap P_r$, $label(p,r) = 'f'$ and $c_{ij} \neq 0$ imply $j \in P_p \cap P_r$, whence

$$\begin{aligned}
\gamma^{(2)}_{\hat{\imath}\,\hat{\jmath}} + \gamma^{(3)}_{\hat{\imath}\,\hat{\jmath}} &= \#\{r \neq p \mid label(p,r) = 'f' \text{ and } i \in P_p \cap P_r\} \\
&= m_i^{(f)} - 1\ ,
\end{aligned}$$

Therefore, if $j \in P_p \cap P_q$ and $label(p,q) = 'f'$, $\gamma^{(1)}_{\hat\imath\hat\jmath} + \gamma^{(2)}_{\hat\imath\hat\jmath} + \gamma^{(3)}_{\hat\imath\hat\jmath} = m_i^{(f)}$, whence

$$\left(\left(\Delta_f\, \Sigma_f\, \hat C\, \Sigma_f\right)_{pq}\right)_{\hat\imath\hat\jmath} = c_{ij} = \left(\left(\hat C\, \Sigma_f\right)_{pq}\right)_{\hat\imath\hat\jmath}$$

On the other hand, $\left(\left(\hat C\, \Sigma_f\right)_{pq}\right)_{\hat\imath\hat\jmath}$ is zero if $j \notin P_p$ or $label\ (p,q) \neq 'f'$. But then, $\gamma^{(1)}_{\hat\imath\hat\jmath}$ is obviously zero while $c_{ij}\gamma^{(2)}_{\hat\imath\hat\jmath} = 0$ because $i \in P_p \cap P_q$, $label(p,q) = 'f'$ and $c_{ij} \neq 0$ would also imply $j \in P_p \cap P_q$ (assumption (a)). Finally, the proof of (1) is concluded by noting that $c_{ij}\,\gamma^{(3)}_{\hat\imath\hat\jmath} \neq 0$ implies $j \in P_p \cap P_r$ for some $r \neq p,q$ (assumption (a)), i.e. $j \in P_p \cap P_r \cap P_q$; it follows then from Property 1 that $label(p,q) = 'f'$ (because $label(p,r)$ $= label(r,q) = 'f'$), i.e. $c_{ij}\,\gamma^{(3)}_{\hat\imath\hat\jmath} \neq 0$ is not compatible with $j \notin P_p$ and $label(p,q) \neq 'f'$.

To prove (2), we first note that

$$\left(\mathcal{L}\, C\, \mathcal{L}^t\right)_{\hat\imath\hat\jmath} = c_{ij}$$

for all $\hat\imath, \hat\jmath$, while

$$\begin{aligned}\left(\left(\Sigma_\ell\, \hat C\, \Sigma_f\right)_{pp}\right)_{\hat\imath\hat\jmath} &= c_{ij} + \sum_{q\neq p} (\Sigma_\ell)_{pq}\ \left(\hat C\right)_{qq}\ (\Sigma_f)_{qp} \\ &= c_{ij}\end{aligned}$$

because one may not have simultaneously $(\Sigma_\ell)_{pq} \neq 0$ and $(\Sigma_f)_{pq} \neq 0$.

Next, we have, for all $p \neq q$,

$$\begin{aligned}\left(\left(\Sigma_\ell\, \hat C\, \Sigma_f\right)_{pq}\right)_{\hat\imath\hat\jmath} &= \left(\left(\hat C\, \Sigma_f\right)_{pq}\right)_{\hat\imath\hat\jmath} + \left(\left(\Sigma_\ell\, \hat C\right)_{pq}\right)_{\hat\imath\hat\jmath} \\ &\quad + \sum_{r\neq p,q}\ \sum_{\hat k\in\hat P_r} \left((\Sigma_\ell)_{pr}\right)_{\hat\imath k}\ \left(\left(\hat C\, \Sigma_f\right)_q\right)_{\hat k\hat\jmath} \\ &= c_{ij}\ \left(\delta^{(1)}_{\hat\imath\hat\jmath} + \delta^{(2)}_{\hat\imath\hat\jmath} + \delta^{(3)}_{\hat\imath\hat\jmath}\right)\end{aligned}$$

where

$$\delta^{(1)}_{\hat\imath\hat\jmath} = \begin{cases} 1 & \text{if } j \in P_p \cap P_q \text{ and } label(p,q) = 'f' \\ 0 & \text{otherwise} \end{cases}$$

$$\delta^{(2)}_{\hat\imath\hat\jmath} = \begin{cases} 1 & \text{if } i \in P_p \cap P_q \text{ and } label(p,q) = '\ell' \\ 0 & \text{otherwise} \end{cases}$$

$$\delta^{(3)}_{\hat\imath\hat\jmath} = \#\left\{r \neq p,q \mid label(p,r) = '\ell',\ label(r,q) = 'f' \text{ and } i,\ j \in P_r\right\}$$

We shall first show that $\delta^{(1)}_{\hat\imath\hat\jmath} + \delta^{(2)}_{\hat\imath\hat\jmath} + \delta^{(3)}_{\hat\imath\hat\jmath} \leq 1$. Obviously, one may not have simultaneously $\delta^{(1)}_{\hat\imath\hat\jmath} = 1$ and $\delta^{(2)}_{\hat\imath\hat\jmath} = 1$.

Next, for any $r \neq p,q$, $j \in P_p \cap P_r \cap P_p$, $label(p,q) = 'f'$, $label(p,r) = 'f'$ and $label(q,r) = '\ell'$ is in contradiction with Property 1, i.e. one may not have simultaneously, $\delta^{(3)}_{\hat\imath\hat\jmath} \geq 1$ and $\delta^{(1)}_{\hat\imath\hat\jmath} = 1$. Similarly, $\delta^{(3)}_{\hat\imath\hat\jmath} \geq 1$ is not compatible with $\delta^{(2)}_{\hat\imath\hat\jmath} = 1$, and, finally, one easily checks that $\delta^{(3)}_{\hat\imath\hat\jmath} \leq 1$ because for any $r_1, r_2 \neq p,q$ with $r_1 \neq r_2$, $j \in P_q \cap P_{r_1} \cap P_{r_2}$,

$label(p, r_1) = label(p, r_2) = 'f'$ imply $\Rightarrow label(r_1, r_2) = 'f'$, whereas $i \in P_p \cap P_{r_1} \cap P_{r_2}$, $label(p, r_1) = label(p, r_2) = '\ell'$ imply $\Rightarrow label(r_1, r_2) = '\ell'$.

It remains thus to prove that $c_{ij} \neq 0 \Rightarrow \gamma^{(1)}_{\hat{\imath}\,\hat{\jmath}} + \gamma^{(2)}_{\hat{\imath}\,\hat{\jmath}} + \gamma^{(3)}_{\hat{\imath}\,\hat{\jmath}} \geq 1$. As $c_{ij} \neq 0 \Rightarrow i, j \in P_r$ for at least one $r \in [1, N]$, we distinguish the following cases :

- $i, j \in P_p \cap P_q$: then, $\gamma^{(1)}_{\hat{\imath}\,\hat{\jmath}} = 1$ if $label(p, q) = 'f'$; $\gamma^{(2)}_{\hat{\imath}\,\hat{\jmath}} = 1$ if $label(p, q) = '\ell'$ and $\gamma^{(3)}_{\hat{\imath}\,\hat{\jmath}} \geq 1$ if $label(p, q) = 'o'$ because, by Property 2, there is then some r such that $i \in P_p \cap P_r \cap P_q$ with $label(p, r) = 'f'$ and $label(r, q) = '\ell'$ while assumption (b) further implies $j \in P_r$.

- $i \in P_q$, $j \notin P_p$: then, $label(p, q) = 'f'$ would be in contradiction with assumption (b); hence, either $label(r, q) = '\ell'$ in which case $\gamma^{(2)}_{\hat{\imath}\,\hat{\jmath}} = 1$ or $label(r, q) = 'o'$, in which case the same argument as above shows $\gamma^{(3)}_{\hat{\imath}\,\hat{\jmath}} \geq 1$

- $i \notin P_q$, $j \in P_p$: here, $label(p, q) = '\ell'$ would be in contradiction with assumption (c); hence, either $\gamma^{(1)}_{\hat{\imath}\,\hat{\jmath}} = 1$, or $label(p, q) = 'o'$, in which case there is some r such that $j \in P_q \cap P_q \cap P_r$ with $label(p, r) = 'f'$ and $label(q, r) = '\ell'$; $\gamma^{(3)}_{\hat{\imath}\,\hat{\jmath}} \geq 1$ follows then from assumption (b) which implies $i \in P_r$.

- $i \notin P_q$, $j \notin P_p$: then, $\exists r \neq p, q$ such that $i, j \in P_r$; as $label(p, r) = 'f'$ would be in contradiction with assumption (b) and $label(r, q) = '\ell'$ in contradiction with assumption (c) $\gamma^{(3)}_{\hat{\imath}\,\hat{\jmath}} \geq 1$ except if $label(p, r) = 'o'$ or $label(q, r) = 'o'$; but $label(p, r) = 'o'$ implies that $i \in P_q \cap P_r \cap P_s$ for some s such that $label(p, s) = 'f'$ and $label(r, s) = '\ell'$, while $label(p, s) = 'f'$ implies together with assumption (b) that $j \in P_p$, case which is excluded here; similarly, one may not have $label(q, r) = 'o'$ and thus $\gamma^{(3)}_{\hat{\imath}\,\hat{\jmath}} \geq 1$.

□

THEOREM 2
Let $\hat{C}$ be an $\hat{n} \times \hat{n}$ matrix and C an $n \times n$ matrix such that

$$\Sigma_f \, \Delta_f \, \hat{C} \, \Sigma_f = \hat{C} \, \Sigma_f$$

and

$$\Sigma_\ell \, \hat{C} \, \Sigma_f = \mathcal{L} \, C \, \mathcal{L}^t \ ,$$

where $\mathcal{L}$ is a $\hat{n} \times n$ matrix of rank n and $\Sigma_f, \Delta_f, \Sigma_\ell$ are symmetric $\hat{n} \times \hat{n}$ matrices satisfying (3.17), (3.18).

If $\hat{C}$ and C are invertible, one has

$$\Sigma_f \, \Delta_f \, \hat{C}^{-1} \, \Sigma_f = \hat{C}^{-1} \, \Sigma_f$$

and

$$\Sigma_\ell \, \hat{C}^{-1} \, \Sigma_f = \mathcal{L} \, C^{-1} \, \mathcal{L}^t \ .$$

Proof. First, because $\mathcal{L}$ is an $\hat{n} \times n$ matrix of rank n, its Moore-Penrose generalized inverse $\mathcal{L}^+$ is an $n \times \hat{n}$ matrix satisfying

$$\mathcal{L}^+ \mathcal{L} = I_n \tag{3.19}$$

(see [3]). On the other hand, letting $\tilde{n}$ be the rank of Σ_f, $\tilde{n} \geq n$ because of (3.18). Therefore, see [3, p. 22] , there exists an $\hat{n} \times \tilde{n}$ matrix $\mathcal{L}_f$ of rank $\tilde{n}$ such that

$$\Sigma_f = \mathcal{L}_f \mathcal{L}_f^t \ . \tag{3.20}$$

Similarly to (3.19), one has

$$\mathcal{L}_f^+ \mathcal{L}_f = I_{\tilde{n}} \ , \tag{3.21}$$

and the multiplication of (3.17) to the left by $\mathcal{L}_f^+$ and to the right by $(\mathcal{L}_f^+)^t$ then gives

$$\mathcal{L}_f^t \Delta_f \mathcal{L}_f = I_{\tilde{n}} \ . \tag{3.22}$$

We next define

$$\mathcal{L}_\ell = \mathcal{L}_f^t \Delta_f \mathcal{L} \tag{3.23}$$

and check that

$$\mathcal{L} = \mathcal{L}_f \mathcal{L}_\ell \tag{3.24}$$

(because $\mathcal{L}^t = \mathcal{L}^+ \Sigma_\ell \Sigma_f = \mathcal{L}^+ \Sigma_\ell \Sigma_f \Delta_f \Sigma_f = \mathcal{L}^t \Delta_f \ \mathcal{L}_f \mathcal{L}_f^t = \mathcal{L}_\ell^t \mathcal{L}_f^t$).

With these relations, it is then not difficult to see that, for any $\hat{n} \times \hat{n}$ matrix $\hat{G}$ and $n \times n$ matrix G, letting

$$\tilde{G} = \mathcal{L}_f^t \Delta_f \hat{G} \mathcal{L}_f \ ,$$

one has

$$\Sigma_f \Delta_f \hat{G} \Sigma_f = \hat{G} \Sigma_f \ \Leftrightarrow \ \mathcal{L}_f \tilde{G} = \hat{G} \mathcal{L}_f \tag{3.25}$$

($\Rightarrow$ multiplying both sides to the right by $(\mathcal{L}_f^+)^t$; $\Leftarrow$ multiplying both sides to the right by $\mathcal{L}_f^t$) and

$$\Sigma_\ell \Sigma_f \Delta_f \hat{G} \Sigma_f = \mathcal{L} G \mathcal{L}^t \ \Leftrightarrow \ \mathcal{L}_\ell^t \tilde{G} = G \mathcal{L}_\ell^t \tag{3.26}$$

($\Rightarrow$ multiplying both sides to the left by $\mathcal{L}^+$ and to the right by $(\mathcal{L}_f^+)^t$; $\Leftarrow$ multiplying both sides to the left by $\mathcal{L}$ and to the right by $\mathcal{L}_f^t$).

One the other hand, if $\hat{G}$ is invertible, $\tilde{G}$ is also invertible (because, by (3.22), Δ_f and $\mathcal{L}_f$ are of rank $\geq \tilde{n}$, and

$$\mathcal{L}_f \tilde{G} = \tilde{G} \mathcal{L}_f \qquad \Leftrightarrow \qquad \hat{G}^{-1} \mathcal{L}_f = \mathcal{L}_f \tilde{G}^{-1} \tag{3.27}$$

whereas, if G is also invertible,

$$\mathcal{L}_\ell^t \tilde{G} = \tilde{G} \mathcal{L}_\ell^t \qquad \Leftrightarrow \qquad \hat{G}^{-1} \mathcal{L}_\ell^t = \mathcal{L}_\ell^t \tilde{G}^{-1} \ . \tag{3.28}$$

This is for any $\hat{G}, G$. Now, letting $\tilde{C} = \mathcal{L}_f^t \Delta_f \hat{C} \mathcal{L}_f$, one has $\tilde{C}^{-1} = \mathcal{L}_f^t \Delta_f \hat{C}^{-1} \mathcal{L}_f$ because, since $\Sigma_f \Delta_f \hat{C} \Sigma_f = \hat{C} \Sigma_f$ by assumption,

$$\begin{aligned}
\mathcal{L}_f^t \Delta_f \hat{C}^{-1} \mathcal{L}_f \mathcal{L}_f^t \Delta_f \hat{C} \mathcal{L}_f &= \mathcal{L}_f^t \Delta_f \hat{C}^{-1} \Sigma_f \Delta_f \hat{C} \Sigma_f \left(\mathcal{L}_f^+\right)^t \\
&= \mathcal{L}_f^t \Delta_f \hat{C}^{-1} \hat{C} \Sigma_f \left(\mathcal{L}_f^+\right)^t \\
&= \mathcal{L}_f^t \Delta_f \mathcal{L}_f \\
&= I_{\tilde{n}} \ .
\end{aligned}$$

Then, with (3.25), (3.27) :

$$\begin{aligned}
\Sigma_f\, \Delta_f\, \hat{C}\, \Sigma_f = \hat{C}\, \Sigma_f \quad \Leftrightarrow \quad & \mathcal{L}_f\, \left(\mathcal{L}_f^t\, \Delta_f\, \hat{C}\, \mathcal{L}_f\right) && = \hat{C}\, \mathcal{L}_f \\
\Leftrightarrow \quad & \mathcal{L}_f\, \left(\mathcal{L}_f^t\, \Delta_f\, \hat{C}\, \mathcal{L}_f\right)^{-1} && = \hat{C}^{-1}\, \mathcal{L}_f \\
\Leftrightarrow \quad & \mathcal{L}_f\, \left(\mathcal{L}_f^t\, \Delta_f\, \hat{C}^{-1}\, \mathcal{L}_f\right) && = \hat{C}^{-1}\, \mathcal{L}_f \\
\Leftrightarrow \quad & \Sigma_f\, \Delta_f\, \hat{C}^{-1}\, \Sigma_f && = \hat{C}^{-1}\, \Sigma_f
\end{aligned}$$

Similarly, $\Sigma_\ell\, \hat{C}\, \Sigma_f = \mathcal{L}\, C\, \mathcal{L}^t \Leftrightarrow \Sigma_\ell\, \hat{C}^{-1}\, \Sigma_f = \mathcal{L}\, \hat{C}^{-1}\, \mathcal{L}^t$ by virtue of (3.26), (3.28). $\square$

Theorem 3

Let $\hat{C}_1, \hat{C}_2$ be $\hat{n} \times \hat{n}$ matrices and C_1, C_2 $n \times n$ matrices such that

$$\Sigma_f\, \Delta_f\, \hat{C}_1\, \Sigma_f = \hat{C}_1\, \Sigma_f \qquad , \qquad \Sigma_f\, \Delta_f\, \hat{C}_2\, \Sigma_f = \hat{C}_2\, \Sigma_f$$

and

$$\Sigma_\ell\, \hat{C}_1\, \Sigma_f = \mathcal{L}\, C_1\, \mathcal{L}^t \qquad , \qquad \Sigma_\ell\, \hat{C}_2\, \Sigma_f = \mathcal{L}\, C_2\, \mathcal{L}^t \ ,$$

where $\mathcal{L}$ is a $\hat{n} \times n$ matrix of rank n and $\Sigma_f, \Delta_f, \Sigma_\ell$ are symmetric $\hat{n} \times \hat{n}$ matrices satisfying (3.17), (3.18).

One has

$$\Sigma_f\, \Delta_f\, \hat{C}_1\, \hat{C}_2\, \Sigma_f = \hat{C}_1\, \hat{C}_2\, \Sigma_f$$

and

$$\Sigma_\ell\, \hat{C}_1\, \hat{C}_2 \Sigma_f = \mathcal{L}\, C_1\, C_2\, \mathcal{L}^t \ .$$

Proof. First, we have

$$\begin{aligned}
\Sigma_f\, \Delta_f\, \hat{C}_1\, \hat{C}_2 \Sigma_f &= \Sigma_f\, \Delta_f\, \hat{C}_1\, \Sigma_f\, \Delta_f\, \hat{C}_2\, \Sigma_f \\
&= \hat{C}_1\, \Sigma_f\, \Delta_f\, \hat{C}_2\, \Sigma_f \\
&= \hat{C}_1\, \hat{C}_2 \Sigma_f
\end{aligned}$$

Next, we note, as in the proof of Theorem 2, that the Moore-Penrose inverse $\mathcal{L}^+$ of $\mathcal{L}$ satisfies $\mathcal{L}^+\, \mathcal{L} = I_n$, relation which implies with (3.18) that $\mathcal{L}^t = \mathcal{L}^+\, \Sigma_\ell\, \Sigma_f$. Therefore :

$$\begin{aligned}
\Sigma_\ell\, \hat{C}_1\, \hat{C}_2 \Sigma_f &= \Sigma_\ell\, \hat{C}_1\, \Sigma_f\, \Delta_f\, \hat{C}_2\, \Sigma_f \\
&= \mathcal{L}\, C_1\, \mathcal{L}^t\, \Delta_f\, \hat{C}_2\, \Sigma_f \\
&= \mathcal{L}\, C_1\, \mathcal{L}^+\, \Sigma_\ell\, \Sigma_f \Delta_f\, \hat{C}_2\, \Sigma_f \\
&= \mathcal{L}\, C_1\, \mathcal{L}^+\, \Sigma_\ell\, \hat{C}_2\, \Sigma_f \\
&= \mathcal{L}\, C_1\, \mathcal{L}^+ \mathcal{L}\, C_2\, \mathcal{L}^t \\
&= \mathcal{L}\, C_1\, C_2\, \mathcal{L}^t
\end{aligned}$$

$\square$

THEOREM 4
Let $\hat{C}_1, \hat{C}_2$ be $\hat{n} \times \hat{n}$ matrices and C_1, C_2 $n \times n$ matrices such that

$$\Sigma_f\,\Delta_f\,\hat{C}_1\,\Sigma_f = \hat{C}_1\,\Sigma_f \qquad , \qquad \Sigma_f\,\Delta_f\,\hat{C}_2\,\Sigma_f = \hat{C}_2\,\Sigma_f$$

and

$$\Sigma_\ell\,\hat{C}_1\,\Sigma_f = \mathcal{L}\,C_1\,\mathcal{L}^t \qquad , \qquad \Sigma_\ell\,\hat{C}_2\,\Sigma_f = \mathcal{L}\,C_2\,\mathcal{L}^t\ ,$$

where $\mathcal{L}$ is a $\hat{n} \times n$ matrix of rank n and $\Sigma_f, \Delta_f, \Sigma_\ell$ are symmetric $\hat{n} \times \hat{n}$ matrices satisfying (3.17), (3.18).

One has

$$\Sigma_f\,\hat{C}_1^t\,\Delta_f\,\Sigma_\ell\,\hat{C}_2\,\Sigma_f = \mathcal{L}\,C_1^t\,C_2\,\mathcal{L}^t\ .$$

and

$$\Sigma_f\,\hat{C}_1^t\,\Sigma_\ell\,\Delta_f\,\hat{C}_2\,\Sigma_f = \mathcal{L}\,C_1^t\,C_2\,\mathcal{L}^t\ .$$

Proof. Taking into account $\mathcal{L}^+\,\mathcal{L} = I_n$ and $\mathcal{L}^t = \mathcal{L}^+\,\Sigma_\ell\,\Sigma_f$ (see proof of Theorems 2 and 3), where $\mathcal{L}^+$ is the Moore-Penrose inverse of $\mathcal{L}$, the proof is straightforward :

$$\begin{aligned}
\Sigma_f\,\hat{C}_1^t\,\Delta_f\,\Sigma_\ell\,\hat{C}_2\,\Sigma_f &= \Sigma_f\,\hat{C}_1^t\,\Delta_f\,\mathcal{L}\,C_2\mathcal{L}^t \\
&= \Sigma_f\,\hat{C}_1^t\,\Delta_f\,\Sigma_f\Sigma_\ell\,\left(\mathcal{L}^+\right)^t\,C_2\,\mathcal{L}^t \\
&= \Sigma_f\,\hat{C}_1^t\,\Sigma_\ell\,\left(\mathcal{L}^+\right)^t\,C_2\,\mathcal{L}^t \\
&= \mathcal{L}\,C_1^t\,\mathcal{L}^t\,\left(\mathcal{L}^+\right)^t\,C_2\,\mathcal{L}^t \\
&= \mathcal{L}\,C_1^t\,C_2\,\mathcal{L}^t\ .
\end{aligned}$$

This also proves

$$\left(\Sigma_f\,\hat{C}_2^t\,\Delta_f\,\Sigma_\ell\,\hat{C}_1\,\Sigma_f\right)^t = \left(\mathcal{L}\,C_2^t\,C_1\mathcal{L}^t\right)^t\ ,$$

i.e.

$$\Sigma_f\,\hat{C}_1^t\,\Sigma_\ell\,\Delta_f\,\hat{C}_2\,\Sigma_f = \mathcal{L}\,C_1^t\,C_2\mathcal{L}^t\ .$$

□

4 Applications

The results in this paper allowed the design of the parallel algorithms in [7] and in [10, 11], which were quite successful for solving two and three dimensional discrete second order elliptic PDEs on respectively MIMD and SIMD computers.

The present paper is more with theoretical concern, and we have unfortunately no room to present these applications here. Let us only mention that it is addressed there the case of regular grids and that, thanks to an appropriate choice of the labels, the parallel ordering strategy proposed in [8] turns out to be fully compatible with conditions (a), (b), (c), i.e. (2.14) holds without having to discard any entry from the triangular factors (for the 5 (2D) or the 7 (3D) point stencil and factorizations without fill).

Another interesting feature is that this global ordering strategy is implemented by means of local lexicographic orderings whose initial node is shifted according the position of the subdomain. Hence, the matrices $\hat{A}$, $\hat{L}$, $\hat{U}$, Σ_f and Σ_ℓ on the augmented system keep a nice regular structure, which explains the efficiency on SIMD computers of an algorithm first designed for MIMD architectures.

Besides this technical detail, the success of this ordering strategy, used together with the implementation scheme derived in this paper, seems lie in a reasonable compromise between an implementation of the preconditioner as is (that is without reordering), which may lead to very poor efficiencies, see e.g. [4], and, on the other hand, an empirical troncation of the preconditioner, which may have a dramatic effect on the convergence rate, see e.g. [12].

Our results are on the other hand not confined to regular grids. They may be applied to parallelize preconditioners on unstructured meshes as well, although , if the processor grid is irregular too, it will generally be difficult to make a sophisticated choice for the labels, i.e. one will use either $label(p,q) = '\ell'$ for all p, q or $label(p,q) = 'f'$ for all p, q. The former choice seems more advantageous as it requires twice less communication than the latter.

Note that, on such grids, one will have generally to discard from the triangular factors some entries originally present in the system matrix. Indeed, even if one orders all interface nodes lastly of first, according the choice of the labels, this cannot help for the entries connecting nodes on different boundaries (i.e. nodes i, j such that there is some p, q with $i \in P_p$, $j \notin P_p$ and $j \in P_q$, $i \notin P_q$). This is however not a shortroming of our analysis. Indeed, if such nonzero entries are present, one cannot, whatever the implementation features, perform the forward and backward solves along the concerned boundaries independently; hence, one should either treat these computations sequentially or foresee some additional communication/synchronization points.

The results in this paper, before allowing to design a parallel implementation, may thus also be seen as a tool to systematically analyse what is to be handled in the preconditioner to allow an efficient parallelization without sequential bottleneck or excessive number of communication steps. Since the equivalence with a well defined sequential preconditioner is kept, one may guess that this handling will have only limited impact on the convergence rate, compared with the dramatic effects which may result from empirical troncations. (For instance, it is know that, instead of the system matrix, one may perform an incomplete factorization of a sparser approximation, and still get similar convergence properties; techniques to derive such approximations are presented in e.g. [1, 5]; they were first designed to enforce the matrix which one factorizes be a Stieltjes matrix, but they may be usefull in the present context as well).

Another application of the results in this paper is the comparison of parallel incomplete factorization with additive Schwarz methods. It is indeed not difficult to see that the latter may be implemented in the framework defined in section 2, using for the preconditioning step

$$\hat{g}^{(k)} = \Sigma\, \hat{C}\, \Sigma\, \hat{r}^{(k)} \tag{4.29}$$

where $\hat{C}$ is some block diagonal matrix (the first multiplication by Σ brings the trace of the gradient on each processor and corresponds to the restriction operator; the second one sums the correction on nodes in the overlapping parts and correspond to the prolongation operator).

If exact subdomain solvers are used then

$$\hat{C} = \left(\mathrm{diag}_{\mathrm{block}}\left(\mathcal{L}\,A\,\mathcal{L}^t\right)\right)^{-1} \;;$$

in many cases however, it does not pay off to make exact inversion and approximate inverses are used instead.

Comparing (4.29) with (2.13), for instance, it is seen that parallel incomplete factorization and additive Schwarz methods are more closely related than one would have thought at first sight, and that our results should allow a better understanding of both family of methods. On the one hand, approximate factorization preconditioners can be improved by integrating some ingredients of the Schwarz methods like overlapping and coarse grid correction. The results in [8] (see Remark 2 in Section 2) may be seen as a very first step in this direction.

On the other hand, the Schwarz algorithms which use minimal overlap and incomplete factorization as approximate inverse on subdomains are not much different from parallel incomplete factorizations, and we hope that our analysis can help to improve them. For instance, if many subdomains are used, the standard coarse grid correction for Schwarz methods may become costly, whereas the results in [8] show that, concerning its counterpart in the incomplete factorization context, scalability is obtained without increasing the coarse mesh together with the number of processors.

5 Parallel computation of the approximate factors

The practical application of the results in Sections 2 and 3 raises the question of the computation of the preconditioner. Indeed besides Σ_ℓ, Σ_f and Δ_f, the factors involved in our parallel preconditioner are defined on each processor as the restriction to the local grid of the corresponding factor in the sequential preconditioner. Hence, at first sight, one has to compute it sequentially before entering the parallel solution algorithm. This may be impractical and represents anyway a serious bottleneck.

In this section we show how this problem can be solved in the case of a symmetric approximate factorization preconditioner

$$B = U^t\,P^{-1}\,U$$

parallelized by means of (2.14). Our analysis is based on the results in the following theorem.

THEOREM 5
Let $i, j \in [1, N]$ be such that

(a) $\quad i, j \in P_p$ *for some* $p \in [1, N]$,

(b) $\quad \left.\begin{array}{l} j \in P_p \cap P_q \textit{ for some } p, q \in [1, N] \textit{ with } p \neq q \\ \textit{label}(p,q) = {'\ell'} \end{array}\right\} \Rightarrow i \in P_p \cap P_q$,

(c) $\quad \left.\begin{array}{l} i \in P_p \cap P_q \textit{ for some } p, q \in [1, N] \textit{ with } p \neq q \\ \textit{label}(p,q) = {'f'} \end{array}\right\} \Rightarrow j \in P_p \cap P_q$,

Then :

(1) $m_j^{(\ell)} \le m_i^{(\ell)}$;

(2) $m_i^{(\ell)} = m_j^{(\ell)} \Rightarrow j \in P_p$ *for all* p *such that* $i \in P_p$;

(3) $m_{ij} = m_i^{(f)} m_j^{(\ell)}$,

where $m_i^{(f)}$ *and* $m_j^{(\ell)}$ *are defined as in Corollary 1 and where*

$$m_{ij} = \#\{p \in [1, N] \mid i, j \in P_p\}$$

Proof. Let p be such that $i, j \in P_p$. Since

$$\begin{aligned} m_i^{(\ell)} &= 1 + \#\{q \neq p \mid i \in P_p \cap P_q \ , \ label(p,q) = '\ell'\} \ , \\ m_j^{(\ell)} &= 1 + \#\{q \neq p \mid j \in P_p \cap P_q \ , \ label(p,q) = '\ell'\} \ , \end{aligned}$$

(1) is obvious from (b). Next, assume $m_i^{(\ell)} = m_j^{(\ell)}$ and that there is some q such that $i \in P_q$, $j \notin P_q$; $label(p,q) = 'f'$ is not possible because of (c); $label(p,q) = '\ell'$ implies on the other hand either $m_i^{(\ell)} > m_j^{(\ell)}$ or that there exists some $r \neq p$ such that $j \in P_r$ with $label(p,r) = '\ell'$ and $i \notin P_r$, but this is in contradiction with (b); finally, $label(p,q) = 'o'$ is also not possible because there is then (Property 2) some r such that $i \in P_p \cap P_q \cap P_r$ with $label(p,q) = 'f'$ and $label(r,q) = '\ell'$; this would imply $j \in P_r$ by (c), and we have already seen that $i, j \in P_r$, $i \in P_q$, $j \notin P_q$ and $label(q,r) = '\ell'$ is in contradition with $m_i^{(\ell)} = m_j^{(\ell)}$.

To check (3), let C be the $n \times n$ matrix such that $c_{k_1 k_2} = \delta_{k_1 i} \delta_{k_2 j}$. Theorem 1 shows that

$$\Sigma_\ell \, \hat{C} \, \Sigma_f = \mathcal{L} \, C \, \mathcal{L}^t$$

where $\hat{C} = \mathrm{diag}_{\mathbf{block}}(C)$.

Now, $(\Sigma_f e)_{\hat{k}_2} = m_{k_2}^{(f)}$ and $(\Sigma_\ell e)_{\hat{k}_1} = m_{k_1}^{(\ell)}$, where k_1, k_2 are such that $\mathcal{L}_{\hat{k}_2 k_1} = \mathcal{L}_{\hat{k}_2 k_2} = 1$. Hence, using the same notation for block components as in the proof of Theorem 1,

$$e^t \, \Sigma_\ell \, \hat{C} \, \Sigma_f \, e = \sum_p \sum_{\hat{k}_1, \hat{k}_2 \in \hat{P}_p} m_{k_1}^{(\ell)} \, m_{k_2}^{(f)} \, \left(\hat{C}_{pp}\right)_{\hat{k}_1 \hat{k}_2}$$

but $\left(\hat{C}_{pp}\right)_{\hat{k}_1 \hat{k}_2}$ is nonzero if and only if $k_1 = i$, $k_2 = j$ and i , $j \in P_p$. Thus,

$$\begin{aligned} e^t \, \Sigma_\ell \, \hat{C} \, \Sigma_f \, e &= \sum_{\substack{p \\ i, j \in P_p}} m_i^{(\ell)} \, m_j^{(f)} \\ &= m_{ij} \, m_i^{(\ell)} \, m_j^{(f)} \ . \end{aligned}$$

On the other hand,

$$\left(\mathcal{L}^t \, e\right)_k = \#\{p \in [1, N] \mid k \in P_p\} = m_k$$

and therefore

$$
\begin{aligned}
e^t \mathcal{L}\, C\, \mathcal{L}^t e &= \sum_{k_1\, k_2} m_{k_1}\, m_{k_2}\, c_{k_1\, k_2} \\
&= m_i\, m_j
\end{aligned}
$$

The required results then readily follows from Corollary 1 witch proves $m_i = m_i^{(\ell)}\, m_i^{(f)}$ for all i. □

Consider now the following version of the symmetric incomplete factorization algorithm.

initialize :
$u_{ij} \;=\; a_{ij}$ *for all* $1 \le i \le j \le n$
$s_i \;=\; \sum_{j>i}$ *for all* $1 \le i \le n$
for $k = 1, \ldots, n$:
for all $i > k$: $u_{ki} \neq 0$:
$u_{ii} := u_{ii} - \frac{u_{ki}^2}{u_{kk}} - \frac{w_k\, u_{ki}}{u_{kk}}\, (s_k - u_{ki})$
for all $j > i$: $u_{kj} \neq 0$ *and if fill-in is permitted in position* (i, j) :
$u_{ij} := u_{ij} - \frac{u_{ki}\, u_{kj}}{u_{kk}}$
$s_i \;:= s_i - \frac{u_{ki}\, u_{kj}}{u_{kk}}$
$u_{ii} := u_{ii} - w_k\, \frac{u_{ki}\, u_{kj}}{u_{kk}}$
$u_{jj} := u_{jj} - w_k\, \frac{u_{ki}\, u_{ki}}{u_{kk}}$

If all fills are discarded, this algorithm is standard, the term $-\frac{w_k\, u_{ki}}{u_{kk}}(s_k - u_{ki})$ meaning that these discarded fills are added to the diagonal with a relaxation factor w_k which may depend on k ($w_k \equiv 0$ correspond to IC and $w_k \equiv 1$ to MIC, but several other choices are possible and even recommanded, see [6] and the references therein, for instance).

In cases where some fill entries are accepted, one sees that the algorithm above (which is on that point non standard) first adds them to the diagonal together with the discarded fills, and next substracts them once it is checked that fill-in was actually permitted.

Let now discuss the parallelization of the algorithm above. Of course, we assume that all possible nonzero u_{ji} in U are compatible with (a), (b), (c).

Theorem 5 (1) implies then that one may reorder the computation, and first eliminate the nodes k with $m_k^{(\ell)} = 1$, next those with $m_k^{(\ell)} = 2$, etc, without affecting the resulting triangular factor. Next, Theorem 5 (2) shows that, for any node i present on some processor p, all the nodes k such that $u_{ki} \neq 0$ and $m_k^{(\ell)} = m_i^{(\ell)}$ are also present on that processor. Hence, at least for factorization without fill, one can manage to eliminates the nodes k with same value of $m_k^{(\ell)}$ independently of all processors.

When some fill-in is allowed, the situation is more tricky because it may happen that, when eliminating k, one has to update some entry u_{ij} where i belongs to some processor p on which j is not present. Then, the record of the update of u_{ii} and s_i on processor p requires an additional communication if $m_i^{(\ell)} = m_k^{(\ell)}$. Note that j belonging to some processor on which i is not present raises no similar difficulties since then we have anyway $m_j^{(\ell)} > m_i^{(\ell)} \ge m_k^{(\ell)}$.

In the algorithm below, we assume that it is forbidden to update some entry u_{ij} if $i \in P_p$ for some p such that $j \notin P_p$. We guess that this will be sufficient for most applications.

This algorithm directly computes the diagonal blocs $\left(\hat{u}_{\hat{\imath}\hat{\jmath}}^{(p)}\right)$, $p = 1, \ldots, N$, of the upper triangular factor $\hat{U}$ on the augmented system. Similarly, the diagonal blocks of $\hat{A}$ are noted $\left(\hat{a}_{\hat{\imath}\hat{\jmath}}^{(p)}\right)$. We assume $\hat{A}$ block diagonal, which means (see Section 2) that the true system matrix A (the one which one has to factorize) corresponds to the assembly of these diagonals blocks.

The basic principle is to consider successively the different classes of nodes with same value of $m_k^{(\ell)}$, and, for each class, first assemble the corresponding rows in U and next eliminate the nodes as in the sequential algorithm above. When an entry is updated in a not yet assembled row, the correction is divided by the number of processor on which the same update is simultaneously performed.

To evaluate this number is easy since, by Theorem 5, the number of processors which share both k and i is equal to $m_k^{(\ell)} m_i^{(f)}$ when $u_{ki} \neq 0$. Moreover, by our additional restriction on the fill entries, the latter number is also equal to the number of processors sharing k, i and j when $u_{ki} u_{kj} \neq 0$ and when the update of u_{ij} is permitted.

We make use of the following notation : the first node in block p is noted $\hat{k}_p^{(f)}$ and the last one $\hat{k}_p^{(\ell)}$ (i.e. $\hat{k}_p^{(f)} = \sum_{q=1}^{p-1} n_q + 1$ and $\hat{k}_p^{(\ell)} = \sum_{q=1}^{p} n_p$) ; for all $\hat{\imath} \in [1, \hat{n}]$, we use $m_{\hat{\imath}}^{(f)}$ and $m_{\hat{\imath}}^{(\ell)}$ to note the value of respectively $m_i^{(f)}$ and $m_i^{(\ell)}$ for the node i such that $\mathcal{L}_{\hat{\imath}\, i} = 1$; by Corollary 1, $m_{\hat{\imath}}^{(f)} = (\Sigma_f e)_{\hat{\imath}}$ and $m_{\hat{\imath}}^{(\ell)} = (\Sigma_\ell e)_{\hat{\imath}}$, i.e. computing these quantities is easy; we also let

$$m_{\hat{k}\,\hat{\imath}} \;=\; \begin{cases} 1 & \text{if } m_{\hat{k}}^{(\ell)} = m_{\hat{\imath}}^{(\ell)} \\ m_{\hat{k}}^{(\ell)}\; m_{\hat{\imath}}^{(f)} & \text{otherwise .} \end{cases}$$

FOR ALL p :

initialize :

$$\hat{u}^{(p)}_{\hat{\imath}\,\hat{\jmath}} = \hat{a}^{(p)}_{\hat{\imath}\,\hat{\jmath}} \qquad \textit{for all } \hat{k}^{(f)}_p \le \hat{\imath} \le \hat{\jmath} \le \hat{k}^{(\ell)}_p$$

$$\hat{s}^{(p)}_{\hat{\imath}} = \textstyle\sum_{\hat{\jmath}>\hat{\imath}} u^{(p)}_{\hat{\imath}\,\hat{\jmath}} \qquad \textit{for all } \hat{k}^{(f)}_p \le \hat{\imath} \le \hat{k}^{(\ell)}_p$$

for $m = 1, \ldots, m_{\max}$:

for all $\hat{k}^{(f)}_p \le \hat{\imath} \le \hat{k}^{(\ell)}_p$ *such that* $m^{(\ell)}_{\hat{\imath}} = m$:

$$\hat{u}^{(p)}_{\hat{\imath}\,\hat{\jmath}} := \hat{u}^{(p)}_{\hat{\imath}\,\hat{\jmath}} + \sum_{q \ne p} \sum_{\substack{\bar{\imath},\bar{\jmath} \in \hat{P}_q \\ \Sigma_{\hat{\imath}\,\bar{\imath}} = \Sigma_{\hat{\jmath}\,\bar{\imath}} = 1}} u^{(q)}_{\bar{\jmath}\,\bar{\jmath}}$$

$$\hat{s}^{(p)}_{\hat{\imath}} := \hat{s}^{(p)}_{\hat{\imath}} + \sum_{q \ne p} \sum_{\substack{\bar{\imath} \in \hat{P}_q \\ \Sigma_{\hat{\imath}\,\bar{\imath}} = 1}} s^{(q)}_{\bar{\imath}}$$

for $\hat{k} = \hat{k}^{(f)}_p, \ldots, \hat{k}^{(\ell)}_P$, *and if* $m^{(\ell)}_{\hat{k}} = m$:

for all $\hat{\imath} > \hat{k}$: $\hat{u}^{(p)}_{\hat{k}\,\hat{\imath}} \ne 0$

$$\hat{u}^{(p)}_{\hat{\imath}\,\hat{\imath}} := \hat{u}^{(p)}_{\hat{\imath}\,\hat{\imath}} - \frac{1}{m_{\hat{k}\,\hat{\imath}}} \frac{\hat{u}^{(p)}_{\hat{k}\,\hat{\imath}}}{\hat{u}_{\hat{k}\,\hat{k}}} \left(\hat{u}^{(p)}_{\hat{k}\,\hat{\imath}} + w_{\hat{k}} \left(\hat{s}^{(p)}_{\hat{k}} - \hat{u}^{(p)}_{\hat{k}\,\hat{\imath}} \right) \right)$$

for all for all $\hat{\jmath} > \hat{\imath}$: $\hat{u}^{(p)}_{\hat{k}\,\hat{\imath}} \ne 0$ *and if fill-in permitted in position* $(\hat{\imath}, \hat{\jmath})$:

$$\hat{u}^{(p)}_{\hat{\imath}\,\hat{\jmath}} := \hat{u}^{(p)}_{\hat{\imath}\,\hat{\jmath}} - \frac{1}{m_{\hat{k}\,\hat{\imath}}} \frac{\hat{u}^{(p)}_{\hat{k}\,\hat{\imath}} \; \hat{u}^{(p)}_{\hat{k}\,\hat{\jmath}}}{\hat{u}_{\hat{k}\,\hat{k}}}$$

$$\hat{s}^{(p)}_{\hat{\imath}} := \hat{s}_{\hat{\imath}} - \frac{1}{m_{\hat{k}\,\hat{\imath}}} \frac{\hat{u}^{(p)}_{\hat{k}\,\hat{\imath}} \; \hat{u}^{(p)}_{\hat{k}\,\hat{\jmath}}}{\hat{u}_{\hat{k}\,\hat{k}}}$$

$$\hat{u}^{(p)}_{\hat{\imath}\,\hat{\imath}} := \hat{u}^{(p)}_{\hat{\imath}\,\hat{\imath}} + \frac{w_{\hat{k}}}{m_{\hat{k}\,\hat{\imath}}} \frac{\hat{u}^{(p)}_{\hat{k}\,\hat{\imath}} \; \hat{u}^{(p)}_{\hat{k}\,\hat{\jmath}}}{\hat{u}_{\hat{k}\,\hat{k}}}$$

$$\hat{u}^{(p)}_{\hat{\jmath}\,\hat{\jmath}} := \begin{cases} \hat{u}^{(p)}_{\hat{\jmath}\,\hat{\jmath}} + w_{\hat{k}} \dfrac{\hat{u}^{(p)}_{\hat{k}\,\hat{\imath}} \; \hat{u}^{(p)}_{\hat{k}\,\hat{\jmath}}}{\hat{u}_{\hat{k}\,\hat{k}}} & \text{if } m^{(\ell)}_{\hat{\jmath}} = m \\ \hat{u}^{(p)}_{\hat{\jmath}\,\hat{\jmath}} + \dfrac{w_{\hat{k}}}{m^{(\ell)}_{\hat{k}}\, m^{(f)}_{\hat{\imath}}} \dfrac{\hat{u}^{(p)}_{\hat{k}\,\hat{\imath}} \; \hat{u}^{(p)}_{\hat{k}\,\hat{\jmath}}}{\hat{u}_{\hat{k}\,\hat{k}}} & \text{otherwise} \end{cases}$$

Acknowledgements This work was supported by IBM through a research contract between ULB and IBM.

References

[1] O. Axelsson and V. A. Barker, *Finite Element Solution of Boundary Value Problems. Theory and Computation*, Academic Press, New York, 1984.

[2] Barrett, Berry, Chan, Demmel, Donato, Dongarra, Eijkhout, Pozo, Romine, and van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM Publications, Philadelphia, 1993.

[3] A. Ben Israel and T. N. E. Greville, *Generalized Inverses : theory and applications*, J. Wiley and Sons, New York, 1974.

[4] P. Manneback and J. Qin, *Algorithmic on a distributed memory mimd computer: a case study*, in Proceedings of Transputers'92, M. B. et al., ed., Amsterdam, 1992, IOS Press, pp. 172–178.

[5] Y. Notay, *Résolution itérative de systèmes linéaires par factorisations approchées*, PhD thesis, Service de Métrologie Nucléaire, Université Libre de Bruxelles, Brussels, Belgium, 1991.

[6] Y. Notay, *DRIC : a dynamic version of the RIC method*, Journ. Num. Lin. Alg. with Appl., 1 (1994), pp. 511–532.

[7] Y. Notay, *An efficient parallel discrete PDE solver*. Parallel Computing, to appear, 1994.

[8] Y. Notay and A. Van de Velde, *Coarse-grid acceleration of parallel incomplete factorization preconditioners*. submitted for publication, 1995.

[9] J. Ortega, *Orderings for conjugate gradient preconditionings*, SIAM J. Optimization, 1 (1990), pp. 565–582.

[10] Z. Ould Amar, *An efficient preconditioning method for SIMD distributed memory-computers*. submitted to Appl. Numer. Math. for publication, 1995.

[11] Z. Ould Amar, *A parallel solver for 3D-PDEs on SIMD computers*, Tech. Rep. IT/IF/14-24, Université Libre de Bruxelles, 1995.

[12] T. Washio and K. Hayami, *Parallel block preconditioning based on SSOR and MILU*, Journ. Num. Lin. Alg. with Appl., 1 (1994), pp. 533–553.