

Asynchronous substructuring method with alternating local and global iterations

Guillaume Gbikpi-Benissan^{a,b}, Frédéric Magoulès^{b,c,*}

^a Engineering Academy, Peoples' Friendship University of Russia (RUDN University), Moscow, Russian Federation

^b Université Paris-Saclay, CentraleSupélec, Gif-sur-Yvette, France

^c Faculty of Engineering and Information Technology, University of Pécs, Pécs, Hungary

ARTICLE INFO

Article history:

Received 5 October 2020

Received in revised form 9 February 2021

Keywords:

Asynchronous methods

Substructuring methods

Domain decomposition methods

Alternating methods

Parallel computing

ABSTRACT

Until now, almost all investigations of asynchronous iterations within domain decomposition frameworks targeted methods of the parallel Schwarz type. A first, and sole, attempt to deal with a primal substructuring framework resulted in an asynchronous substructuring method where relaxation occurs simultaneously on the subdomains and on the interface between them, which therefore corresponds to a substructured relaxation scheme defined on the whole global domain. In this paper, we propose a Gauss–Seidel kind of improvement consisting of alternating between relaxation on the interface and relaxation on the subdomains, hence, always using the latest solutions in the subdomains when updating the solution on the interface, which is feasible at no additional cost. It turns out that one particular case of our general alternating relaxation scheme corresponds to an asynchronous substructuring method with iterations fully defined on the subdomains' interface, and where only local Schur complements are involved. Practical performance evaluation on both standard Poisson's and linear elasticity problems has been conducted using a multi-node parallel computational platform with up to 720 CPU cores.

© 2021 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

1. Introduction

We are considering sparse linear problems

$$Ax = b, \quad x \in \mathbb{R}^n, \quad n \in \mathbb{N}, \quad (1)$$

arising, e.g. as discrete analogues of partial differential equations defined on domains Ω . With a partitioning of Ω into p parts, p discrete subsystems

$$A^{(i)}x^{(i)} = b^{(i)}, \quad x^{(i)} \in \mathbb{R}^{n^{(i)}}, \quad n^{(i)} \in \mathbb{N}, \quad i \in \{1, \dots, p\},$$

can be obtained on subdomains $\Omega^{(i)}$ sharing, however, a subset of unknowns lying on an interface Γ joining the partitions. Each subdomain i interacts on only a portion of Γ which will be denoted by $\Gamma^{(i)}$. Non-interface unknowns will be said to

* Corresponding author at: Université Paris-Saclay, CentraleSupélec, Gif-sur-Yvette, France.

E-mail addresses: guibenissan@gmail.com (G. Gbikpi-Benissan), frederic.magoules@hotmail.com (F. Magoulès).

URL: <http://www.computational-science.org/html/mag/biosketch.html> (F. Magoulès).

be “local” to subdomains, and will be designated by the subscript “L”. It yields

$$A^{(i)} := \begin{bmatrix} A_{LL}^{(i)} & A_{L\Gamma^{(i)}}^{(i)} \\ A_{\Gamma^{(i)}L}^{(i)} & A_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} \end{bmatrix}, \quad b^{(i)} := \begin{bmatrix} b_L^{(i)} \\ b_{\Gamma^{(i)}}^{(i)} \end{bmatrix} \quad (2)$$

and a global problem (1) in a form

$$A := \begin{bmatrix} A_{LL}^{(1)} & 0 & \cdots & 0 & A_{L\Gamma}^{(1)} \\ 0 & A_{LL}^{(2)} & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & A_{L\Gamma}^{(p-1)} \\ 0 & \cdots & 0 & A_{LL}^{(p)} & A_{L\Gamma}^{(p)} \\ A_{\Gamma L}^{(1)} & \cdots & A_{\Gamma L}^{(p-1)} & A_{\Gamma L}^{(p)} & A_{\Gamma\Gamma} \end{bmatrix}, \quad b := \begin{bmatrix} b_L^{(1)} \\ b_L^{(2)} \\ \vdots \\ b_L^{(p)} \\ b_{\Gamma} \end{bmatrix}, \quad (3)$$

with

$$A_{L\Gamma}^{(i)} := A_{L\Gamma^{(i)}}^{(i)} C_{\Gamma^{(i)}\Gamma}^{(i)}, \quad A_{\Gamma L}^{(i)} := C_{\Gamma\Gamma^{(i)}}^{(i)} A_{\Gamma^{(i)}L}^{(i)}, \quad C_{\Gamma\Gamma^{(i)}}^{(i)} = C_{\Gamma^{(i)}\Gamma}^{(i)\top},$$

where $C_{\Gamma^{(i)}\Gamma}^{(i)}$ is a Boolean restriction matrix allowing one to index only unknowns on Γ which are on $\Gamma^{(i)}$. It will finally be assumed that

$$A_{\Gamma\Gamma} := \sum_{i=1}^p A_{\Gamma\Gamma}^{(i)}, \quad b_{\Gamma} := \sum_{i=1}^p b_{\Gamma}^{(i)},$$

with, here as well,

$$A_{\Gamma\Gamma}^{(i)} := C_{\Gamma\Gamma^{(i)}}^{(i)} A_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} C_{\Gamma^{(i)}\Gamma}^{(i)}, \quad b_{\Gamma}^{(i)} := C_{\Gamma\Gamma^{(i)}}^{(i)} b_{\Gamma^{(i)}}^{(i)}.$$

We are interested in solving (1) using asynchronous relaxation [1] over p parallel processes. Relaxation methods feature a low convergence speed compared to Krylov approaches, however, being able to run iterations with randomized interprocess synchronization appears to be a seducing approach for massively or heterogeneous parallel scientific computing. Asynchronous relaxation therefore keeps being particularly attractive, and one can refer to, e.g., recent works [2–9] and [10], among others. Most efforts in the development of asynchronous iterations focused on overlapping subdomains approaches (see, e.g., [11–15] and references therein). To the best of our knowledge, the nonoverlapping subdomains framework (3) has been investigated only very recently in [16]. We propose, here, an improvement and further analysis of the asynchronous relaxation-based substructuring method introduced there. Note that, according to the substructuring framework (3), the proposed method does not distribute the set of interface unknowns among the p processes. Instead, if $\Gamma^{(i,j)}$, $i \neq j$, is the interface portion between the subdomains i and j , $i, j \in \{1, \dots, p\}$, then it is straightforwardly ensured, at convergence, that

$$x_{\Gamma^{(i,j)}} = x_{\Gamma^{(i,j)}}^{(i)} = x_{\Gamma^{(i,j)}}^{(j)}.$$

2. Preliminaries

2.1. Generalities

We denote by $\mathcal{A} = (a_{ij})$ a matrix \mathcal{A} with entry a_{ij} on its i th row and j th column. $x = (x_i)$ expresses the analogous for a vector x . Comparisons $\mathcal{A} < \mathcal{B}$ and $\mathcal{A} \leq \mathcal{B}$ of two matrices or vectors \mathcal{A} and \mathcal{B} are entrywise. The absolute value $|\mathcal{A}|$ of \mathcal{A} is entrywise too. The spectral radius of \mathcal{A} is denoted as $\rho(\mathcal{A})$. We recall weighted norms

$$\|\mathcal{A}\|_{\infty}^w := \max_i \frac{1}{w_i} \sum_j |a_{ij}| w_j,$$

where $w = (w_i)$ is a positive (entrywise) vector. We note $w > 0$.

An M-matrix \mathcal{A} will be considered here as a nonsingular square matrix with no positive offdiagonal entry and a nonnegative (entrywise) inverse, i.e., $\mathcal{A}^{-1} \geq 0$ (see, e.g., [17]). The comparison matrix $\langle \mathcal{A} \rangle$ of a square matrix \mathcal{A} is obtained by taking the absolute value of the diagonal entries and the opposite of the absolute value of the offdiagonal ones. \mathcal{A} is an H-matrix when $\langle \mathcal{A} \rangle$ is an M-matrix. Finally, we denote by $\mathcal{A} = \mathcal{M} - \mathcal{N}$ a splitting of \mathcal{A} with \mathcal{M} being nonsingular. $\mathcal{A} = \mathcal{M} - \mathcal{N}$ is an H-splitting when $\langle \mathcal{M} \rangle - |\mathcal{N}|$ is an M-matrix.

Lemma 1 (See, e.g., Corollary 6.1 in [18]).

$$\rho(|\mathcal{A}|) < 1 \iff \exists w > 0 : \|\mathcal{A}\|_{\infty}^w < 1.$$

Lemma 2. If $A = \mathcal{M} - \mathcal{N}$ is an H-splitting, then

$$\rho(|\mathcal{I} - \mathcal{M}^{-1}\mathcal{A}|) < 1$$

with \mathcal{I} denoting the identity matrix.

Proof. This directly follows from the proof of Theorem 3.4 (c) in [19]. \square

2.2. Classical asynchronous relaxation

Consider again the problem (1) with a splitting $A = M - N$. Classical relaxation is then given by the iterative relation

$$x^{k+1} = x^k + M^{-1}(b - Ax^k), \quad k \in \mathbb{N}, \quad (4)$$

defining a sequence of vectors, $\{x^k\}_{k \in \mathbb{N}}$, which converges to the solution of (1) if

$$\rho(I - M^{-1}A) < 1$$

with I denoting the identity matrix. The classical asynchronous relaxation model (see, e.g., [20,21]) arose in a parallel computing context where the set of updated components is distributed over the set of processes. With the substructured form (3) of the matrix A , a parallel Jacobi relaxation, for instance, is naturally given by

$$\begin{cases} x_L^{(i),k+1} = x_L^{(i),k} + D_{LL}^{(i)-1} \left(b_L^{(i)} - A_{LL}^{(i)} x_L^{(i),k} - A_{LR}^{(i)} x_R^k \right) \\ x_R^{k+1} = x_R^k + D_{RR}^{-1} \left(b_R - \sum_{i=1}^p A_{RL}^{(i)} x_L^{(i),k} - A_{RR} x_R^k \right). \end{cases} \quad \forall i \in \{1, \dots, p\}, \quad (5)$$

This thus would require a $p+1$ -th process dedicated to the unknowns on Γ . The first step toward asynchronous relaxation is the free steering model (see, e.g., [22]) where, at each iteration k , an arbitrary subset $U^{(k)} \subseteq \{1, \dots, p+1\}$ of unknowns can be updated. To completely randomize such a computing model, the asynchronous relaxation scheme additionally allows for updating unknowns, e.g., $x_L^{(i),k+1}$, using, this time, an arbitrarily outdated value $x_R^{\delta_{p+1}^i(k)}$ from some iteration number $\delta_{p+1}^i(k) \leq k$, which describes communication delay from the process $p+1$ to the process i , $i \in \{1, \dots, p\}$. The asynchronous version of such a Jacobi relaxation would therefore be given by

$$\begin{cases} x_L^{(i),k+1} = \begin{cases} x_L^{(i),\delta_i^i(k)} + D_{LL}^{(i)-1} \left(b_L^{(i)} - A_{LL}^{(i)} x_L^{(i),\delta_i^i(k)} - A_{LR}^{(i)} x_R^{\delta_{p+1}^i(k)} \right) & \forall i \in U^{(k)}, \\ x_L^{(i),k} & \forall i \notin U^{(k)}, \end{cases} \\ x_R^{k+1} = \begin{cases} x_R^{\delta_{p+1}^{p+1}(k)} + D_{RR}^{-1} \left(b_R - \sum_{i=1}^p A_{RL}^{(i)} x_L^{(i),\delta_i^{p+1}(k)} - A_{RR} x_R^{\delta_{p+1}^{p+1}(k)} \right), & p+1 \in U^{(k)}, \\ x_R^k, & p+1 \notin U^{(k)}. \end{cases} \end{cases}$$

In a general manner, the following result ensures the convergence of such classically derived asynchronous relaxation methods.

Theorem 1 (Chazan and Miranker (1969) [1]). A splitting $A = M - N$ is asynchronously convergent if

$$\rho(|I - M^{-1}A|) < 1.$$

A similar convergence condition has been established in [16] for the asynchronous substructured relaxation method introduced there, which we briefly recall in the next section.

2.3. Asynchronous substructured relaxation

The iterative model (5) is obviously unpractical due to the severely unbalanced workload and the induced star network topology. Moreover, only the incomplete interface matrices $A_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)}$, $i \in \{1, \dots, p\}$, are usually explicitly known, and

building $A_{\Gamma\Gamma}$ could require a significant preprocessing step. In [16], the first (practical) asynchronous substructuring method is introduced, based on a splitting $A = M - N$ with M being of the form

$$M := \begin{bmatrix} M_{LL}^{(1)} & 0 & \cdots & 0 & 0 \\ 0 & M_{LL}^{(2)} & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & M_{LL}^{(p)} & 0 \\ 0 & \cdots & 0 & 0 & M_{\Gamma\Gamma} \end{bmatrix}. \quad (6)$$

The main operation to be made parallel in relaxations (4) is the matrix–vector product Ax^k . According to local available data (2), take

$$x^{(i),k} := \begin{bmatrix} x_L^{(i),k} & x_{\Gamma(i)}^{(i),k} \end{bmatrix}^T, \quad x_{\Gamma(i)}^{(i),k} = C_{\Gamma(i)\Gamma}^{(i)} x_{\Gamma}^k, \quad i \in \{1, \dots, p\}.$$

A product $z = Ax^k$ can therefore be performed in a parallel substructured form given by

$$\begin{cases} y^{(i),k} &:= A^{(i)} x^{(i),k}, & i \in \{1, \dots, p\}, \\ z_L^{(i)} &= y_L^{(i),k} & \forall i \in \{1, \dots, p\}, \\ z_{\Gamma} &= \sum_{i=1}^p C_{\Gamma\Gamma(i)}^{(i)} y_{\Gamma(i)}^{(i),k}. \end{cases}$$

Consider, then, diagonal weighting matrices $I_{\Gamma(i)\Gamma(i)}^{(i)}$, $i \in \{1, \dots, p\}$, such that

$$I_{\Gamma\Gamma} = \sum_{i=1}^p I_{\Gamma\Gamma}^{(i)}, \quad I_{\Gamma\Gamma}^{(i)} := C_{\Gamma\Gamma(i)}^{(i)} I_{\Gamma(i)\Gamma(i)}^{(i)} C_{\Gamma(i)\Gamma}^{(i)},$$

where $I_{\Gamma\Gamma}$ is the identity matrix on Γ , and assume that $M_{\Gamma\Gamma}^{-1}$ is such that its product by an extended local vector is equivalent to the extension of the product of its restriction by that local vector, i.e.,

$$M_{\Gamma\Gamma}^{-1} C_{\Gamma\Gamma(i)}^{(i)} x_{\Gamma(i)}^{(i)} = C_{\Gamma\Gamma(i)}^{(i)} \left(C_{\Gamma(i)\Gamma}^{(i)} M_{\Gamma\Gamma}^{-1} C_{\Gamma(i)\Gamma}^{(i)} \right) x_{\Gamma(i)}^{(i)}.$$

This is satisfied if, for instance, $M_{\Gamma\Gamma}$ is a diagonal matrix. Define, finally,

$$I^{(i)} := \begin{bmatrix} I_{LL}^{(i)} & 0 \\ 0 & I_{\Gamma(i)\Gamma(i)}^{(i)} \end{bmatrix}, \quad M^{(i)-1} := \begin{bmatrix} M_{LL}^{(i)-1} & 0 \\ 0 & C_{\Gamma(i)\Gamma}^{(i)} M_{\Gamma\Gamma}^{-1} C_{\Gamma(i)\Gamma}^{(i)} \end{bmatrix},$$

where $I_{LL}^{(i)}$ denotes the identity matrix on the subdomain i . The asynchronous substructuring method [16] can be reformulated as

$$\begin{cases} y^{(i),k} &:= I^{(i)} x^{(i),k} + M^{(i)-1} (b^{(i)} - A^{(i)} x^{(i),k}), & i \in \{1, \dots, p\}, \\ x_L^{(i),k+1} &= \begin{cases} y_L^{(i),\delta_i^k(k)} & \forall i \in U^{(k)}, \\ x_L^{(i),k} & \forall i \notin U^{(k)}, \end{cases} \\ x_{\Gamma(i)}^{(i),k+1} &= \begin{cases} C_{\Gamma(i)\Gamma}^{(i)} \sum_{j=1}^p C_{\Gamma\Gamma(j)}^{(j)} y_{\Gamma(j)}^{(j),\delta_j^k(k)} & \forall i \in U^{(k)}, \\ x_{\Gamma(i)}^{(i),k} & \forall i \notin U^{(k)} \end{cases} \end{cases} \quad (7)$$

with, now, $U^{(k)} \subseteq \{1, \dots, p\}$. It is easy to verify that if, for instance, M is the diagonal part of A , then the synchronous instance of (7) is equivalent to the parallel Jacobi relaxation (5), considering

$$x_{\Gamma(i)}^{(i),k+1} := C_{\Gamma(i)\Gamma}^{(i)} x_{\Gamma}^{k+1}, \quad x_{\Gamma}^{k+1} = \sum_{j=1}^p C_{\Gamma\Gamma(j)}^{(j)} y_{\Gamma(j)}^{(j),k}.$$

Such an equivalence does not hold in case of arbitrary asynchronous instances, which makes Theorem 1 not directly applicable, although an almost same result has been established for such a nonoverlapping subdomains framework.

Assumption 1. Let the matrices $I_{\Gamma\Gamma}^{(i)} - M_{\Gamma\Gamma}^{-1} A_{\Gamma\Gamma}^{(i)}$, $i \in \{1, \dots, p\}$, be such that

$$\sum_{i=1}^p \left| I_{\Gamma\Gamma}^{(i)} - M_{\Gamma\Gamma}^{-1} A_{\Gamma\Gamma}^{(i)} \right| = \left| \sum_{i=1}^p I_{\Gamma\Gamma}^{(i)} - M_{\Gamma\Gamma}^{-1} A_{\Gamma\Gamma}^{(i)} \right|.$$

Theorem 2 (Magoulès and Venet (2018) [16]). Under [Assumption 1](#), the asynchronous substructuring method (7) is convergent if

$$\rho(|I - M^{-1}A|) < 1.$$

3. Asynchronous alternating substructuring method

3.1. Iterative scheme

Consider, from (7), the definition of $y^{(i),k}$ in a more explicit form

$$\begin{cases} y_L^{(i),k} &:= x_L^{(i),k} + M_{LL}^{(i)-1} \left(b_L^{(i)} - A_{LL}^{(i)} x_L^{(i),k} - A_{L\Gamma^{(i)}}^{(i)} x_{\Gamma^{(i)}}^{(i),k} \right), \\ y_{\Gamma^{(i)}}^{(i),k} &:= I_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} x_{\Gamma^{(i)}}^{(i),k} + C_{\Gamma^{(i)}\Gamma}^{(i)} M_{\Gamma\Gamma}^{-1} C_{\Gamma\Gamma^{(i)}}^{(i)} \left(b_{\Gamma^{(i)}}^{(i)} - A_{\Gamma^{(i)}L}^{(i)} x_L^{(i),k} \right. \\ &\quad \left. - A_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} x_{\Gamma^{(i)}}^{(i),k} \right). \end{cases}$$

One can thus notice that the update of the interface vector can easily be improved by using the newly computed value of the subdomain vector, i.e, replacing $x_L^{(i),k}$ by $y_L^{(i),k}$ in the definition of $y_{\Gamma^{(i)}}^{(i),k}$. It yields alternations between local iterations on the subdomains and global iterations on the interface, in contrast with the substructuring approach [16] featuring simultaneous subdomain and interface updates. Our substructured alternating relaxation model is thus given by

$$\begin{cases} y_L^{(i),k} &:= x_L^{(i),k} + M_{LL}^{(i)-1} \left(b_L^{(i)} - A_{LL}^{(i)} x_L^{(i),k} - A_{L\Gamma^{(i)}}^{(i)} x_{\Gamma^{(i)}}^{(i),k} \right), \\ y_{\Gamma^{(i)}}^{(i),k} &:= I_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} x_{\Gamma^{(i)}}^{(i),k} + C_{\Gamma^{(i)}\Gamma}^{(i)} M_{\Gamma\Gamma}^{-1} C_{\Gamma\Gamma^{(i)}}^{(i)} \left(b_{\Gamma^{(i)}}^{(i)} \right. \\ &\quad \left. - A_{\Gamma^{(i)}L}^{(i)} y_L^{(i),k} - A_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} x_{\Gamma^{(i)}}^{(i),k} \right), \\ x_L^{(i),k+1} &= \begin{cases} y_L^{(i),\delta_i^1(k)} & \forall i \in U^{(k)}, \\ x_L^{(i),k} & \forall i \notin U^{(k)}, \end{cases} \\ x_{\Gamma^{(i)}}^{(i),k+1} &= \begin{cases} C_{\Gamma^{(i)}\Gamma}^{(i)} \sum_{j=1}^p C_{\Gamma\Gamma^{(j)}}^{(j)} y_{\Gamma^{(j)}}^{(j),\delta_j^1(k)} & \forall i \in U^{(k)}, \\ x_{\Gamma^{(i)}}^{(i),k} & \forall i \notin U^{(k)}. \end{cases} \end{cases} \quad (8)$$

One particularly interesting instance of (8) consists of taking $M_{LL}^{(i)} := A_{LL}^{(i)}$, which eliminates the subdomains relaxation since we would have

$$y_L^{(i),k} := A_{LL}^{(i)-1} \left(b_L^{(i)} - A_{L\Gamma^{(i)}}^{(i)} x_{\Gamma^{(i)}}^{(i),k} \right).$$

By substituting $y_L^{(i),k}$ in the definition of $y_{\Gamma^{(i)}}^{(i),k}$, it results

$$y_{\Gamma^{(i)}}^{(i),k} := I_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} x_{\Gamma^{(i)}}^{(i),k} + C_{\Gamma^{(i)}\Gamma}^{(i)} M_{\Gamma\Gamma}^{-1} C_{\Gamma\Gamma^{(i)}}^{(i)} \left(b_{\Gamma^{(i)}}^{(i)} - A_{\Gamma^{(i)}L}^{(i)} A_{LL}^{(i)-1} b_L^{(i)} \right. \\ \left. - \left(A_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} - A_{\Gamma^{(i)}L}^{(i)} A_{LL}^{(i)-1} A_{L\Gamma^{(i)}}^{(i)} \right) x_{\Gamma^{(i)}}^{(i),k} \right).$$

This therefore exhibits a pure interface relaxation method

$$\begin{cases} y_{\Gamma^{(i)}}^{(i),k} &:= I_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} x_{\Gamma^{(i)}}^{(i),k} + C_{\Gamma^{(i)}\Gamma}^{(i)} M_{\Gamma\Gamma}^{-1} C_{\Gamma\Gamma^{(i)}}^{(i)} \left(d_{\Gamma^{(i)}}^{(i)} \right. \\ &\quad \left. - S_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} x_{\Gamma^{(i)}}^{(i),k} \right), \\ x_{\Gamma^{(i)}}^{(i),k+1} &= \begin{cases} C_{\Gamma^{(i)}\Gamma}^{(i)} \sum_{j=1}^p C_{\Gamma\Gamma^{(j)}}^{(j)} y_{\Gamma^{(j)}}^{(j),\delta_j^1(k)} & \forall i \in U^{(k)}, \\ x_{\Gamma^{(i)}}^{(i),k} & \forall i \notin U^{(k)} \end{cases} \end{cases} \quad (9)$$

with

$$d_{\Gamma^{(i)}}^{(i)} := b_{\Gamma^{(i)}}^{(i)} - A_{\Gamma^{(i)}L}^{(i)} A_{LL}^{(i)-1} b_L^{(i)}, \\ S_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} := A_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} - A_{\Gamma^{(i)}L}^{(i)} A_{LL}^{(i)-1} A_{L\Gamma^{(i)}}^{(i)},$$

where $S_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)}$ appears to be the Schur complement of $A_{LL}^{(i)}$ in $A^{(i)}$.

3.2. Convergence conditions

The weighted norms

$$\|\mathcal{A}\|_{\infty}^w := \max_i \frac{1}{w_i} \sum_j |a_{i,j}| w_j, \quad w > 0,$$

operate on square matrices $\mathcal{A} = (a_{i,j})$ using a unique positive vector $w = (w_i)$. For the analysis of our alternating substructuring method, we will additionally need a concise notation, $|\mathcal{A}|_v^w$, of more general weighted row-sum operators

$$(|\mathcal{A}|_v^w)_i := \frac{1}{v_i} \sum_j |a_{i,j}| w_j, \quad v = (v_i) > 0, \quad w = (w_j),$$

allowing us to similarly handle matrices of any shape.

Lemma 3. Let $\mathcal{A} = (a_{i,j})$ and $\mathcal{B} = (b_{j,l})$ be matrices with shapes allowing for the product \mathcal{AB} . Let $z = (z_j) > 0$, $v = (v_i) > 0$ and $w = (w_l)$ be vectors with sizes allowing for operators $|\mathcal{A}|_v^w$ and $|\mathcal{B}|_z^w$. Then, we have

$$|\mathcal{B}|_z^w < \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T \implies |\mathcal{AB}|_v^w < |\mathcal{A}|_v^z.$$

Proof. Let $(ab)_{i,l}$ denote entries of the product \mathcal{AB} . Then, we have

$$\begin{aligned} \frac{1}{v_i} \sum_l |(ab)_{i,l}| w_l &= \frac{1}{v_i} \sum_l \left| \sum_j a_{i,j} b_{j,l} \right| w_l \\ &\leq \frac{1}{v_i} \sum_l \sum_j |a_{i,j} b_{j,l}| w_l \\ &= \frac{1}{v_i} \sum_l \sum_j \frac{1}{z_j} |a_{i,j}| |b_{j,l}| z_j w_l \\ &= \frac{1}{v_i} \sum_j \left(\frac{1}{z_j} \sum_l |b_{j,l}| w_l \right) |a_{i,j}| z_j. \end{aligned}$$

If, therefore,

$$\frac{1}{z_j} \sum_l |b_{j,l}| w_l < 1 \quad \forall j,$$

then

$$\begin{aligned} \left(\frac{1}{z_j} \sum_l |b_{j,l}| w_l \right) |a_{i,j}| z_j &< |a_{i,j}| z_j \quad \forall j \forall i, \\ \frac{1}{v_i} \sum_j \left(\frac{1}{z_j} \sum_l |b_{j,l}| w_l \right) |a_{i,j}| z_j &< \frac{1}{v_i} \sum_j |a_{i,j}| z_j \quad \forall i, \\ \frac{1}{v_i} \sum_l |(ab)_{i,l}| w_l &\leq \frac{1}{v_i} \sum_j \left(\frac{1}{z_j} \sum_l |b_{j,l}| w_l \right) |a_{i,j}| z_j < \frac{1}{v_i} \sum_j |a_{i,j}| z_j \quad \forall i, \end{aligned}$$

which concludes the proof. \square

Theorem 3. Under [Assumption 1](#), the asynchronous alternating substructuring method (8) is convergent if

$$\rho(|I - M^{-1}A|) < 1.$$

Proof. Define, for any $i \in \{1, \dots, p\}$,

$$\begin{aligned} T_{LL}^{(i)} &:= I_{LL}^{(i)} - M_{LL}^{(i)-1} A_{LL}^{(i)}, & T_{LR}^{(i)} &:= -M_{LL}^{(i)-1} A_{LR}^{(i)}, \\ T_{RL}^{(i)} &:= -M_{RR}^{-1} A_{RL}^{(i)} T_{LL}^{(i)}, & T_{RR}^{(i)} &:= I_{RR}^{(i)} - M_{RR}^{-1} A_{RR}^{(i)} - M_{RR}^{-1} A_{RL}^{(i)} T_{LR}^{(i)}. \end{aligned}$$

Define, further,

$$T_{LL} := \begin{bmatrix} T_{LL}^{(1)} & 0 & \cdots & 0 \\ 0 & T_{LL}^{(2)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & T_{LL}^{(p)} \end{bmatrix}, \quad T_{L\zeta} := \begin{bmatrix} T_{LR}^{(1)} & 0 & \cdots & 0 \\ 0 & T_{LR}^{(2)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & T_{LR}^{(p)} \end{bmatrix},$$

$$T_{\zeta L} := \begin{bmatrix} T_{RL}^{(1)} & T_{RL}^{(2)} & \cdots & T_{RL}^{(p)} \\ T_{RL}^{(1)} & T_{RL}^{(2)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & T_{RL}^{(p)} \\ T_{RL}^{(1)} & \cdots & T_{RL}^{(p-1)} & T_{RL}^{(p)} \end{bmatrix}, \quad T_{\zeta\zeta} := \begin{bmatrix} T_{RR}^{(1)} & T_{RR}^{(2)} & \cdots & T_{RR}^{(p)} \\ T_{RR}^{(1)} & T_{RR}^{(2)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & T_{RR}^{(p)} \\ T_{RR}^{(1)} & \cdots & T_{RR}^{(p-1)} & T_{RR}^{(p)} \end{bmatrix},$$

where ζ expresses the augmented space Γ^p . Then, the global iteration mapping induced by (8) in the framework of Theorem 1 is given by

$$T := \begin{bmatrix} T_{LL} & T_{L\zeta} \\ T_{\zeta L} & T_{\zeta\zeta} \end{bmatrix}.$$

Note, now, that

$$I - M^{-1}A = \begin{bmatrix} T_{LL}^{(1)} & 0 & \cdots & 0 & T_{LR}^{(1)} \\ 0 & T_{LL}^{(2)} & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & T_{LR}^{(p-1)} \\ 0 & \cdots & 0 & T_{LL}^{(p)} & T_{LR}^{(p)} \\ -M_{RR}^{-1}A_{RL}^{(1)} & \cdots & -M_{RR}^{-1}A_{RL}^{(p-1)} & -M_{RR}^{-1}A_{RL}^{(p)} & I_{RR} - M_{RR}^{-1}A_{RR} \end{bmatrix}.$$

According to Lemma 1,

$$\rho(|I - M^{-1}A|) < 1 \iff \exists w > 0 : \|I - M^{-1}A\|_{\infty}^w < 1.$$

Let such a vector w be decomposed as $w = (w_L^{(1)}, \dots, w_L^{(p)}, w_R)^T$, and consider, for any $i \in \{1, \dots, p\}$,

$$T_{LL \times R}^{(i)} := [T_{LL}^{(i)} \quad T_{LR}^{(i)}], \quad w^{(i)} := [w_L^{(i)} \quad w_R]^T.$$

Then, from $\|I - M^{-1}A\|_{\infty}^w < 1$, we have both

$$\begin{cases} \left| T_{LL \times R}^{(i)} \right|_{w_L^{(i)}}^{w^{(i)}} < [1 \quad 1 \quad \cdots \quad 1]^T, \\ \sum_{i=1}^p \left| -M_{RR}^{-1}A_{RL}^{(i)} \right|_{w_R}^{w_L^{(i)}} + |I_{RR} - M_{RR}^{-1}A_{RR}|_{w_R}^{w_R} < [1 \quad 1 \quad \cdots \quad 1]^T. \end{cases} \quad (10)$$

According to Lemma 3, the first inequality of (10) implies

$$\left| -M_{RR}^{-1}A_{RL}^{(i)} T_{LL \times R}^{(i)} \right|_{w_R}^{w_L^{(i)}} < \left| -M_{RR}^{-1}A_{RL}^{(i)} \right|_{w_R}^{w_L^{(i)}},$$

which further implies, from the second inequality,

$$\sum_{i=1}^p \left| -M_{RR}^{-1}A_{RL}^{(i)} T_{LL \times R}^{(i)} \right|_{w_R}^{w_L^{(i)}} + |I_{RR} - M_{RR}^{-1}A_{RR}|_{w_R}^{w_R} < [1 \quad 1 \quad \cdots \quad 1]^T.$$

If, therefore, according to Assumption 1,

$$\sum_{i=1}^p |I_{RR}^{(i)} - M_{RR}^{-1}A_{RR}^{(i)}| = |I_{RR} - M_{RR}^{-1}A_{RR}|,$$

then we equivalently have

$$\sum_{i=1}^p \left| -M_{RR}^{-1}A_{RL}^{(i)} T_{LL \times R}^{(i)} \right|_{w_R}^{w_L^{(i)}} + \sum_{i=1}^p |I_{RR}^{(i)} - M_{RR}^{-1}A_{RR}^{(i)}|_{w_R}^{w_R} < [1 \quad 1 \quad \cdots \quad 1]^T.$$

Note, in addition, that

$$\begin{aligned} \left| -M_{\Gamma\Gamma}^{-1} A_{\Gamma L}^{(i)} T_{LL \times \Gamma}^{(i)} \right|_{w_{\Gamma}}^{w_{\Gamma}^{(i)}} &= \left| -M_{\Gamma\Gamma}^{-1} A_{\Gamma L}^{(i)} T_{LL}^{(i)} \right|_{w_{\Gamma}}^{w_{\Gamma}^{(i)}} + \left| -M_{\Gamma\Gamma}^{-1} A_{\Gamma L}^{(i)} T_{L\Gamma}^{(i)} \right|_{w_{\Gamma}}^{w_{\Gamma}^{(i)}} \\ &= \left| T_{\Gamma L}^{(i)} \right|_{w_{\Gamma}}^{w_{\Gamma}^{(i)}} + \left| -M_{\Gamma\Gamma}^{-1} A_{\Gamma L}^{(i)} T_{L\Gamma}^{(i)} \right|_{w_{\Gamma}}^{w_{\Gamma}^{(i)}}, \end{aligned}$$

which leads to

$$\sum_{i=1}^p \left| T_{\Gamma L}^{(i)} \right|_{w_{\Gamma}}^{w_{\Gamma}^{(i)}} + \left| -M_{\Gamma\Gamma}^{-1} A_{\Gamma L}^{(i)} T_{L\Gamma}^{(i)} \right|_{w_{\Gamma}}^{w_{\Gamma}^{(i)}} + \left| I_{\Gamma\Gamma}^{(i)} - M_{\Gamma\Gamma}^{-1} A_{\Gamma\Gamma}^{(i)} \right|_{w_{\Gamma}}^{w_{\Gamma}^{(i)}} < [1 \quad 1 \quad \cdots \quad 1]^{\top}.$$

Note, at last, that

$$\begin{aligned} \left| -M_{\Gamma\Gamma}^{-1} A_{\Gamma L}^{(i)} T_{L\Gamma}^{(i)} + I_{\Gamma\Gamma}^{(i)} - M_{\Gamma\Gamma}^{-1} A_{\Gamma\Gamma}^{(i)} \right|_{w_{\Gamma}}^{w_{\Gamma}^{(i)}} &\leq \left| -M_{\Gamma\Gamma}^{-1} A_{\Gamma L}^{(i)} T_{L\Gamma}^{(i)} \right|_{w_{\Gamma}}^{w_{\Gamma}^{(i)}} \\ &\quad + \left| I_{\Gamma\Gamma}^{(i)} - M_{\Gamma\Gamma}^{-1} A_{\Gamma\Gamma}^{(i)} \right|_{w_{\Gamma}}^{w_{\Gamma}^{(i)}}, \end{aligned}$$

and, therefore, we also have

$$\begin{aligned} \sum_{i=1}^p \left| T_{\Gamma L}^{(i)} \right|_{w_{\Gamma}}^{w_{\Gamma}^{(i)}} + \left| -M_{\Gamma\Gamma}^{-1} A_{\Gamma L}^{(i)} T_{L\Gamma}^{(i)} + I_{\Gamma\Gamma}^{(i)} - M_{\Gamma\Gamma}^{-1} A_{\Gamma\Gamma}^{(i)} \right|_{w_{\Gamma}}^{w_{\Gamma}^{(i)}} &< [1 \quad 1 \quad \cdots \quad 1]^{\top}, \\ \sum_{i=1}^p \left| T_{\Gamma L}^{(i)} \right|_{w_{\Gamma}}^{w_{\Gamma}^{(i)}} + \left| T_{\Gamma\Gamma}^{(i)} \right|_{w_{\Gamma}}^{w_{\Gamma}^{(i)}} &< [1 \quad 1 \quad \cdots \quad 1]^{\top}. \end{aligned}$$

It follows, with the first inequality in (10), that

$$\|T\|_{\infty}^v < 1, \quad v := [w_L^{(1)} \quad \cdots \quad w_L^{(p)} \quad w_{\Gamma} \quad \cdots \quad w_{\Gamma}]^{\top},$$

which concludes the proof, according to Lemma 1 and Theorem 1. \square

Now, consider a portion of Γ only shared by the subdomains i and j , $i, j \in \{1, \dots, p\}$. Let, on this portion, an entry $(A_{\Gamma\Gamma})_{l,t}$ in $A_{\Gamma\Gamma}$ correspond to entries $(A_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)})_{l^{(i)},t^{(i)}}$ and $(A_{\Gamma^{(j)}\Gamma^{(j)}}^{(j)})_{l^{(j)},t^{(j)}}$, i.e.,

$$\begin{cases} (C_{\Gamma\Gamma^{(i)}}^{(i)} A_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} C_{\Gamma^{(i)}\Gamma}^{(i)})_{l,t} = (A_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)})_{l^{(i)},t^{(i)}} \\ (C_{\Gamma\Gamma^{(j)}}^{(j)} A_{\Gamma^{(j)}\Gamma^{(j)}}^{(j)} C_{\Gamma^{(j)}\Gamma}^{(j)})_{l,t} = (A_{\Gamma^{(j)}\Gamma^{(j)}}^{(j)})_{l^{(j)},t^{(j)}}. \end{cases}$$

It is usual, e.g., with finite elements methods, to satisfy

$$\begin{cases} (A_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)})_{l^{(i)},t^{(i)}} = \lambda (A_{\Gamma\Gamma})_{l,t}, \\ (A_{\Gamma^{(j)}\Gamma^{(j)}}^{(j)})_{l^{(j)},t^{(j)}} = (1 - \lambda) (A_{\Gamma\Gamma})_{l,t}, \end{cases} \quad 0 \leq \lambda < 1.$$

This is expressed by the following assumption.

Assumption 2. Let $\Lambda_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} \geq 0$, $i \in \{1, \dots, p\}$, be weighting matrices such that

$$\begin{cases} A_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} = \Lambda_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} \circ (C_{\Gamma^{(i)}\Gamma}^{(i)} A_{\Gamma\Gamma} C_{\Gamma\Gamma^{(i)}}^{(i)}), \\ I_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} = \Lambda_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} \circ (C_{\Gamma^{(i)}\Gamma}^{(i)} I_{\Gamma\Gamma} C_{\Gamma\Gamma^{(i)}}^{(i)}) \end{cases}$$

with \circ indicating a Hadamard product, and

$$\sum_{i=1}^p C_{\Gamma\Gamma^{(i)}}^{(i)} \Lambda_{\Gamma^{(i)}\Gamma^{(i)}}^{(i)} C_{\Gamma^{(i)}\Gamma}^{(i)} = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}.$$

Corollary 1. Under Assumption 2, the asynchronous alternating substructuring method (8) is convergent if $M_{\Gamma\Gamma}$ is diagonal and

$$\rho(|I - M^{-1}A|) < 1.$$

Proof. Note, first, that

$$\begin{aligned} \Lambda_{r^{(i)}r^{(i)}}^{(i)} \circ \left(C_{r^{(i)}r}^{(i)} A_{rr} C_{rr^{(i)}}^{(i)} \right) &= \left(C_{r^{(i)}r}^{(i)} C_{rr^{(i)}}^{(i)} \Lambda_{r^{(i)}r^{(i)}}^{(i)} C_{r^{(i)}r}^{(i)} C_{rr^{(i)}}^{(i)} \right) \\ &\quad \circ \left(C_{r^{(i)}r}^{(i)} A_{rr} C_{rr^{(i)}}^{(i)} \right) \\ &= C_{r^{(i)}r}^{(i)} \left(\left(C_{rr^{(i)}}^{(i)} \Lambda_{r^{(i)}r^{(i)}}^{(i)} C_{r^{(i)}r}^{(i)} \right) \right. \\ &\quad \left. \circ A_{rr} \right) C_{rr^{(i)}}^{(i)}. \end{aligned}$$

Under [Assumption 2](#) then,

$$\begin{aligned} A_{r^{(i)}r^{(i)}}^{(i)} &= C_{r^{(i)}r}^{(i)} \left(\left(C_{rr^{(i)}}^{(i)} \Lambda_{r^{(i)}r^{(i)}}^{(i)} C_{r^{(i)}r}^{(i)} \right) \circ A_{rr} \right) C_{rr^{(i)}}^{(i)}, \\ A_{rr}^{(i)} &= \left(C_{rr^{(i)}}^{(i)} \Lambda_{r^{(i)}r^{(i)}}^{(i)} C_{r^{(i)}r}^{(i)} \right) \circ A_{rr}, \\ I_{rr}^{(i)} - M_{rr}^{-1} A_{rr}^{(i)} &= \left(C_{rr^{(i)}}^{(i)} \Lambda_{r^{(i)}r^{(i)}}^{(i)} C_{r^{(i)}r}^{(i)} \right) \circ I_{rr} \\ &\quad - M_{rr}^{-1} \left(\left(C_{rr^{(i)}}^{(i)} \Lambda_{r^{(i)}r^{(i)}}^{(i)} C_{r^{(i)}r}^{(i)} \right) \circ A_{rr} \right). \end{aligned}$$

If, therefore, M_{rr} is diagonal, then

$$\begin{aligned} I_{rr}^{(i)} - M_{rr}^{-1} A_{rr}^{(i)} &= \left(C_{rr^{(i)}}^{(i)} \Lambda_{r^{(i)}r^{(i)}}^{(i)} C_{r^{(i)}r}^{(i)} \right) \circ I_{rr} \\ &\quad - \left(C_{rr^{(i)}}^{(i)} \Lambda_{r^{(i)}r^{(i)}}^{(i)} C_{r^{(i)}r}^{(i)} \right) \circ (M_{rr}^{-1} A_{rr}) \\ &= \left(C_{rr^{(i)}}^{(i)} \Lambda_{r^{(i)}r^{(i)}}^{(i)} C_{r^{(i)}r}^{(i)} \right) \circ (I_{rr} - M_{rr}^{-1} A_{rr}), \\ \sum_{i=1}^p |I_{rr}^{(i)} - M_{rr}^{-1} A_{rr}^{(i)}| &= \left(\sum_{i=1}^p C_{rr^{(i)}}^{(i)} \Lambda_{r^{(i)}r^{(i)}}^{(i)} C_{r^{(i)}r}^{(i)} \right) \circ |I_{rr} - M_{rr}^{-1} A_{rr}| \\ &= |I_{rr} - M_{rr}^{-1} A_{rr}|. \end{aligned}$$

[Assumption 1](#) is so satisfied, hence, [Theorem 3](#) applies, which concludes the proof. \square

Corollary 2. Under [Assumption 1](#), the asynchronous alternating substructuring method [\(8\)](#) is convergent if A is an H-matrix and

$$\begin{cases} \langle M_{LL}^{(i)} \rangle - |M_{LL}^{(i)} - A_{LL}^{(i)}| = \langle A_{LL}^{(i)} \rangle & \forall i \in \{1, \dots, p\}, \\ \langle M_{rr} \rangle - |M_{rr} - A_{rr}| = \langle A_{rr} \rangle. \end{cases}$$

Proof. According to the substructured forms [\(3\)](#) and [\(6\)](#) of A and M , we have

$$\begin{cases} \langle M_{LL}^{(i)} \rangle - |M_{LL}^{(i)} - A_{LL}^{(i)}| = \langle A_{LL}^{(i)} \rangle \\ \langle M_{rr} \rangle - |M_{rr} - A_{rr}| = \langle A_{rr} \rangle \end{cases} \implies \langle M \rangle - |M - A| = \langle A \rangle.$$

A being an H-matrix, $\langle A \rangle$ is an M-matrix, hence, by definition, $A = M - (M - A)$ is an H-splitting. [Lemma 2](#) therefore ensures that

$$\rho(|I - M^{-1}A|) < 1,$$

hence, [Theorem 3](#) applies, which concludes the proof. \square

Corollary 3. Under [Assumption 2](#), the asynchronous alternating substructuring method [\(8\)](#) is convergent if A is an H-matrix, M_{rr} is diagonal, and

$$\begin{cases} \langle M_{LL}^{(i)} \rangle - |M_{LL}^{(i)} - A_{LL}^{(i)}| = \langle A_{LL}^{(i)} \rangle & \forall i \in \{1, \dots, p\}, \\ \langle M_{rr} \rangle - |M_{rr} - A_{rr}| = \langle A_{rr} \rangle. \end{cases}$$

Proof. This is straightforward from [Corollaries 1](#) and [2](#). \square

Let $\mathcal{D}(\mathcal{A})$ stand for the matrix obtained by setting all offdiagonal entries of a matrix \mathcal{A} to zero. In regard to [Corollaries 2](#) and [3](#), it is useful to notice, e.g., that

$$\mathcal{M} = \mathcal{D}(\mathcal{A}) \implies \langle \mathcal{M} \rangle - |\mathcal{M} - \mathcal{A}| = \langle \mathcal{A} \rangle,$$

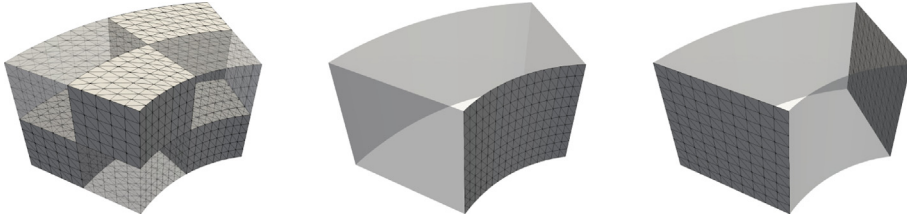


Fig. 1. Left: partitioned domain mesh for 8 subdomains. Middle: inner side. Right: start and end sides.

and, as applied in our numerical experiments,

$$\mathcal{M} = \alpha \mathcal{D}(\mathcal{A}) \geq 0 \implies \langle \mathcal{M} \rangle - |\mathcal{M} - \mathcal{A}| = \langle \mathcal{A} \rangle, \quad \alpha \geq 1.$$

Note, at last, in regard to the interface relaxation method (9), that one also obviously has

$$M_{LL}^{(i)} := A_{LL}^{(i)} \implies \langle M_{LL}^{(i)} \rangle - |M_{LL}^{(i)} - A_{LL}^{(i)}| = \langle A_{LL}^{(i)} \rangle.$$

4. Experiments

4.1. Computational environment

For numerical experiments, we got access to 30 nodes of an SGI ICE X computational platform equipped with an Infiniband network at 56 Gb/s. The nodes consist of 58 GB of memory and 2×12 cores on two Intel Haswell Xeon CPUs with a frequency of 2.30 GHz. Up to 720 processor cores could thus be used. The Message Passing Interface (MPI) standard was provided through the SGI-MPT library, and each subdomain was handled by one MPI process on one dedicated core. The different methods were programmed in the Python language, for which the MPI4Py module gives access to the MPI library (see, e.g., [23,24]). The P1 Lagrangian finite elements method was considered through the SfePy library [25] to generate the local matrices and vectors $A^{(i)}$ and $b^{(i)}$ on each subdomain, independently. Matrices and all operations were handled in the SciPy CSR format.

4.2. Numerical setting

Standard Poisson's and linear elasticity problems,

$$-\Delta u = f$$

and

$$-\operatorname{div} \sigma(u) = f,$$

with uniform volume source/load f , were considered on a partitioned portion of helicoid domain which mesh is shown in Fig. 1, left, for an example with $8 (2 \times 2 \times 2)$ subdomains. The domain is 1 meter wide, $\pi/4$ long in rotation and 1 meter thick. Its raising factor was set to 0.25. We refer to, e.g., [26] for details about parametric equations of helicoids. For the Poisson's problem, a Dirichlet boundary condition, $u = 0$, was set on the inner side of the domain (Fig. 1, middle). $71 \times 111 \times 71$ mesh points were generated (width \times length \times thickness), yielding a problem size of $n = 559551$ unknowns. For the linear elasticity equation, the Dirichlet condition was set at the start and end sides of the domain (Fig. 1, right). The Young modulus and Poisson ratio were set to 32.5 GPa and 0.18, respectively. $46 \times 72 \times 46$ mesh points were generated, yielding $n = 457056$ unknowns.

Parallel computational performance was compared between the substructuring method (7) introduced by [16], our alternating substructuring method (8) and the implied interface relaxation method (9) arising from taking $M_{LL}^{(i)} := A_{LL}^{(i)}$ $\forall i \in \{1, \dots, p\}$. We considered matrix splitting

$$A = \alpha \mathcal{D}(A) - N, \quad \alpha \geq 1,$$

or, in the pure interface relaxation case,

$$A_{\Gamma\Gamma} = \alpha \mathcal{D}(A_{\Gamma\Gamma}) - N_{\Gamma\Gamma}, \quad \alpha \geq 1.$$

Convergence was obtained from $\alpha = 1$ for the Poisson's problem, and from $\alpha = 1.4$ for the linear elasticity one. Table 1 reports the parallel processing configurations considered for the tests.

Table 1
Parallel processing configurations.

#nodes	#cores	domain partitioning (width \times length \times thickness)	#subdomains (p)
4	96	$4 \times 6 \times 4$	96
8	192	$6 \times 8 \times 4$	192
16	384	$8 \times 8 \times 6$	384
30	720	$9 \times 10 \times 8$	720

Table 2
Comparison of synchronous performances on the Poisson's problem.

p	Substructuring			Alternating substructuring		
	Time (s)	k	$ Ax - b $	Time (s)	k	$ Ax - b $
96	316	164 563	9.90E-05	303	158 218	9.90E-05
192	265	164 563	9.90E-05	251	156 199	9.90E-05
384	249	164 563	9.90E-05	236	153 746	9.90E-05
720	301	164 563	9.90E-05	275	151 547	9.90E-05

Table 3
Comparison of asynchronous performances on the Poisson's problem.

p	Substructuring			Alternating substructuring		
	Time (s)	k_{\max}	$ Ax - b $	Time (s)	k_{\max}	$ Ax - b $
96	167	322 608	9.90E-05	158	311 185	9.90E-05
192	166	396 477	9.89E-05	156	370 847	9.90E-05
384	169	504 649	9.90E-05	157	480 117	9.90E-05
720	170	548 306	9.89E-05	158	518 043	9.89E-05

Table 4
Performance of the interface relaxation method (alternating substructuring method with $M_{LL}^{(i)} := A_{LL}^{(i)}$) on the Poisson's problem.

p	Synchronous			Asynchronous		
	Time (s)	k	$ Ax - b $	Time (s)	k_{\max}	$ Ax - b $
96	546	22 370	9.90E-05	553	73 464	9.84E-05
192	265	29 511	9.90E-05	261	98 831	9.78E-05
384	149	37 802	9.90E-05	120	103 801	9.87E-05
720	115	46 136	9.90E-05	84	168 450	9.92E-05

4.3. Performance results

Table 2, first, compares synchronous executions of the substructuring and alternating substructuring methods. Execution (wall-clock) times, numbers of iterations, k , and final residual error, $\|Ax - b\|$, are presented, for a residual threshold set to $1.00\text{E}-04$. As expected, using the newly updated subdomain solutions for the update of the interface solution improved the convergence of the substructured relaxation. More importantly, we can notice that the number of iterations of the alternating method decreases while the number of subdomains increases. It is a quite interesting behavior which can be explained by the fact that the ratio of alternation increases as the subdomains sizes are reduced and more of interface appears. The alternating approach is therefore similar to gradually switching from a substructured Jacobi to a substructured Gauss-Seidel relaxation as the degree of parallelism gets high.

On the asynchronous execution side as well, **Table 3** confirms the performance gain. About asynchronous iterations, since a process $i \in \{1, \dots, p\}$ never waits for data, it performs a maximum possible number of iterations, say $k^{(i)}$. Performance can therefore be characterized using either the slowest or the fastest process. We focused here on $k_{\max} = \max_i k^{(i)}$. Besides, as shown by the final residual errors evaluated, convergence was consistently accurately detected, using no particular detection protocol though, according to the recent work [27].

Still, it is noticeable that both the substructured and the alternating substructured relaxation methods reached a scaling limit under the described computational settings, even in their asynchronous versions. Switching, then, to the interface relaxation method allowed for overcoming such a limit, as one can see in **Table 4**.

The same behavior is globally observed with the linear elasticity test case, as reported in **Tables 5–7**. Here, however, it appeared that the perturbation introduced by asynchronous iterations has a greater impact on the convergence of the three methods, which made synchronous executions more efficient. Still, the impact of workload imbalance, for instance, remains significantly lower in case of asynchronous execution, as shown by **Tables 8–10**, where one computing node was 5 times slowed down.

Table 5

Comparison of synchronous performances on the linear elasticity problem.

p	Substructuring			Alternating substructuring		
	Time (s)	k	$ Ax - b $	Time (s)	k	$ Ax - b $
96	428	147 468	9.90E-05	405	141 471	9.90E-05
192	305	147 468	9.90E-05	289	140 282	9.90E-05
384	251	147 468	9.90E-05	233	137 841	9.90E-05
720	285	147 468	9.90E-05	273	135 958	9.90E-05

Table 6

Comparison of asynchronous performances on the linear elasticity problem.

p	Substructuring			Alternating substructuring		
	Time (s)	k_{\max}	$ Ax - b $	Time (s)	k_{\max}	$ Ax - b $
96	783	1 084 289	9.90E-05	733	1 033 730	9.90E-05
192	614	1 237 247	9.90E-05	582	1 172 877	9.90E-05
384	629	1 640 060	9.88E-05	591	1 550 450	9.88E-05
720	622	1 899 685	9.89E-05	577	1 685 070	9.93E-05

Table 7Performance of the interface relaxation method (alternating substructuring method with $M_{LL}^{(i)} := A_{LL}^{(i)}$) on the linear elasticity problem.

p	Synchronous			Asynchronous		
	Time (s)	k	$ Ax - b $	Time (s)	k_{\max}	$ Ax - b $
96	694	30 083	9.90E-05	2575	287 745	9.86E-05
192	348	39 527	9.90E-05	1173	436 757	9.88E-05
384	193	50 227	9.90E-05	496	580 210	9.91E-05
720	139	59 732	9.90E-05	381	802 017	9.86E-05

Table 8

Performance of the substructuring method on the linear elasticity problem with one 5x-slower node.

p	Synchronous			Asynchronous		
	Time (s)	k	$ Ax - b $	Time (s)	k_{\max}	$ Ax - b $
96	1827	147 468	9.90E-05	1912	2 964 149	9.89E-05
192	1259	147 468	9.90E-05	1105	2 404 985	9.85E-05
384	981	147 468	9.90E-05	642	1 783 569	9.87E-05
720	730	147 468	9.90E-05	598	1 915 474	9.94E-05

Table 9

Performance of the alternating substructuring method on the linear elasticity problem with one 5x-slower node.

p	Synchronous			Asynchronous		
	Time (s)	k	$ Ax - b $	Time (s)	k_{\max}	$ Ax - b $
96	1784	141 471	9.90E-05	1804	2 840 371	9.88E-05
192	1157	140 282	9.90E-05	1054	2 294 140	9.88E-05
384	920	137 841	9.90E-05	604	1 715 769	9.89E-05
720	754	135 958	9.90E-05	566	1 803 069	9.88E-05

Table 10Performance of the interface relaxation method (alternating substructuring method with $M_{LL}^{(i)} := A_{LL}^{(i)}$) on the linear elasticity problem with one 5x-slower node.

p	Synchronous			Asynchronous		
	Time (s)	k	$ Ax - b $	Time (s)	k_{\max}	$ Ax - b $
96	2849	30 083	9.90E-05	6306	654 929	9.82E-05
192	1461	39 527	9.90E-05	1963	694 990	9.81E-05
384	793	50 227	9.90E-05	557	641 356	9.95E-05
720	449	59 732	9.90E-05	371	808 234	9.93E-05

5. Conclusion

Despite the low convergence rate of relaxation methods, their ability to run on parallel asynchronous processes is a key seducing feature for overcoming efficiency issues related to, e.g., unbalanced workload or computing node failure.

Nonoverlapping domain decomposition methods therefore provide a very suitable framework for applying asynchronous methods, as they allow for reducing problems sizes from large domains to relatively small subdomains' interfaces, hence, potentially make relaxation approaches more competitive. In this paper, we improved an asynchronous substructured relaxation method by alternating between subdomains and subdomains' interface relaxation. A particular case of our alternating substructuring iterative model consists of exactly inverting the local matrices defined on the interior of the subdomains. This results in a relaxation method defined only on the interface between the subdomains, which constitutes, to the best of our knowledge, the only asynchronous primal nonoverlapping domain decomposition method developed so far. Depending, therefore, on the local matrices splitting scheme, our substructured relaxation model acts as a nonoverlapping counterpart of the usual asynchronous two-stage overlapping Schwarz method. Experiments conducted on a parallel computational platform with up to 30 nodes of 24 processor cores (total of 720 cores) showed the practical applicability of the improved substructuring method, and confirmed the performance gain, on both Poisson's and linear elasticity problems.

Until now, the possible applications of the asynchronous iterations theory to domain decomposition frameworks seemed limited to Schwarz methods, probably due to their inherent relaxation feature. This work therefore encourages for further possibilities and new promising investigations of the asynchronous computing paradigm.

Acknowledgments

The paper has been prepared with the support of the "RUDN University Program 5-100, Russia", the French national program LEFE/INSU, the project ADOM (Méthodes de décomposition de domaine asynchrones) of the French National Research Agency (ANR), and using HPC resources from the "Mésocentre" computing center of CentraleSupélec and École Normale Supérieure Paris-Saclay supported by CNRS and Région Île-de-France.

References

- [1] D. Chazan, W. Miranker, Chaotic relaxation, *Linear Algebra Appl.* 2 (2) (1969) 199–222.
- [2] P. Nayak, T. Cojean, H. Anzt, Evaluating asynchronous Schwarz solvers on GPUs, *Int. J. High Perform. Comput. Appl.* (2020) <http://dx.doi.org/10.1177/1094342020946814>.
- [3] E. Coleman, E.J. Jensen, M. Sosonkina, Fault recovery methods for asynchronous linear solvers, *Int. J. Parallel Program.* (2020) <http://dx.doi.org/10.1007/s10766-020-00676-w>.
- [4] J. Hook, N. Dingle, Performance analysis of asynchronous parallel Jacobi, *Numer. Algorithms* 77 (2018) 831–866.
- [5] C. Glusa, P. Ramanan, E.G. Boman, E. Chow, S. Rajamanickam, Asynchronous one-level and two-level domain decomposition solvers, *CoRR abs/1808.08172* (2018) <http://arxiv.org/abs/1808.08172>.
- [6] A. Fanfakh, J.-C. Charr, R. Couturier, A. Giersch, Energy consumption reduction for asynchronous message-passing applications, *J. Supercomput.* 73 (2017) 2369–2401.
- [7] J. Wolfson-Pou, E. Chow, Reducing communication in distributed asynchronous iterative methods, *Procedia Comput. Sci.* 80 (2016) 1906–1916.
- [8] I. Bethune, J.M. Bull, N.J. Dingle, N.J. Higham, Performance analysis of asynchronous Jacobi's method implemented in MPI, SHMEM and OpenMP, *Int. J. High Perform. Comput. Appl.* 28 (1) (2014) 97–111.
- [9] H.R. Feyzmahdavian, M. Johansson, On the convergence rates of asynchronous iterations, in: 53rd IEEE Conference on Decision and Control, December 15–17, 2014, Los Angeles, California, USA, 2014, pp. 153–159.
- [10] M. Chau, T. Garcia, P. Spiteri, Asynchronous Schwarz methods applied to constrained mechanical structures in grid environment, *Adv. Eng. Softw.* 74 (2014) 1–15.
- [11] A. Frommer, H. Schwanndt, D.B. Szyld, Asynchronous weighted additive Schwarz methods, *Electron. Trans. Numer. Anal.* 5 (1997) 48–61.
- [12] M.J. Castel, V. Migallón, J. Penadés, Convergence of non-stationary parallel multisplitting methods for hermitian positive definite matrices, *Math. Comp.* 67 (221) (1998) 209–220.
- [13] P. Spiteri, J.-C. Miellou, D. El Baz, Parallel asynchronous Schwarz and multisplitting methods for a nonlinear diffusion problem, *Numer. Algorithms* 33 (1) (2003) 461–474.
- [14] E. Laitinen, A.V. Lapin, J. Pieskä, Asynchronous domain decomposition methods for continuous casting problem, *J. Comput. Appl. Math.* 154 (2) (2003) 393–413.
- [15] M. Chau, C. Tauber, P. Spiteri, Parallel Schwarz alternating methods for anisotropic diffusion of speckled medical images, *Numer. Algorithms* 51 (2009) 85–114.
- [16] F. Magoulès, C. Venet, Asynchronous iterative sub-structuring methods, *Math. Comput. Simulation* 145 (Supplement C) (2018) 34–49.
- [17] K. Fan, Note on M -matrices, *Q. J. Math.* 11 (1) (1960) 43–49.
- [18] D.P. Bertsekas, J.N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall Inc., Upper Saddle River, NJ, USA, 1989.
- [19] A. Frommer, D.B. Szyld, H -splittings and two-stage iterative methods, *Numer. Math.* 63 (1) (1992) 345–356.
- [20] D.P. Bertsekas, J.N. Tsitsiklis, Some aspects of parallel and distributed iterative algorithms – A survey, *Autom. J. IFAC* 27 (1) (1991) 3–21.
- [21] A. Frommer, D.B. Szyld, On asynchronous iterations, *J. Comput. Appl. Math.* 123 (1–2) (2000) 201–216.
- [22] S. Schechter, Relaxation methods for linear equations, *Comm. Pure Appl. Math.* 12 (2) (1959) 313–335.
- [23] L. Dalcín, R. Paz, M. Storti, MPI for Python, *J. Parallel Distrib. Comput.* 65 (9) (2005) 1108–1115.
- [24] L.D. Dalcín, R.R. Paz, P.A. Kler, A. Cosimo, Parallel distributed computing using Python, *Adv. Water Resour.* 34 (9) (2011) 1124–1139, *New Computational Methods and Software Tools*.
- [25] R. Cimrman, V. Lukeš, E. Rohan, Multiscale finite element calculations in Python using SfePy, *Adv. Comput. Math.* 45 (2019) 1897–1921.
- [26] S.N. Krivoshapko, M. Rynkovskaya, Five types of ruled helical surfaces for helical conveyers, support anchors and screws, *MATEC Web Conf.* 95 (2017) 06002.
- [27] G. Gbikpi-Benissan, F. Magoulès, Protocol-free asynchronous iterations termination, *Adv. Eng. Softw.* 146 (2020) 102827.