Engineering Analysis with Boundary Elements 37 (2013) 646–657

Contents lists available at SciVerse ScienceDirect

Engineering Analysis with Boundary Elements

journal homepage: www.elsevier.com/locate/enganabound

The derived-vector space framework and four general purposes massively parallel DDM algorithms

Ismael Herrera*, Alberto A. Rosas-Medina

Instituto de Geofísica, Universidad Nacional Autónoma de México (UNAM), Apdo. Postal 22-220, México, D.F.14000, Mexico

ARTICLE INFO

Article history:
Received 20 August 2012
Accepted 14 December 2012
Available online 1 March 2013

Keywords:
Massively parallel algorithms
Parallel-computers
Non-overlapping DDM
DDM with constraints
BDDC
FETI-DP

ABSTRACT

Ideally, DDMs seek what we call the DDM-paradigm: "constructing the *global* solution by solving *local* problems, exclusively". To achieve it, it is essential to disconnect the subdomain-problems. In FETI-DP such disconnection is achieved by formulating the method in a product function-space that contains discontinuous functions. However, FETI-DP uses an indirect formulation based on Lagrange-multipliers. BDDC uses instead a more direct formulation, but does not work directly in a space of discontinuous functions, either. Another fact difficult to overcome is: at present competitive algorithms need to incorporate constraints that prevent full disconnection of the subdomains. This paper is devoted to explain a direct (primal) approach to DDMs in which all the numerical work is done in a product-space (the derived-vector space), which supplies a unified setting for non-overlapping DDMs and can be used to formulate and discuss in a general and systematic manner the theory of DDMs for non-symmetric problems. Furthermore, in this realm four general-purposes preconditioned algorithms with constraints applicable to non-symmetric matrices, which achieve the DDM-paradigm, have been obtained. Two of them have been identified as DVS-versions of BDDC and FETI-DP. The uniformity of the matrix-formulas expressing such algorithms should be highlighted.

1. Introduction

Mathematical models of many systems of interest, including very important continuous systems of Engineering and Science, lead to a great variety of partial differential equations whose solution methods are based on the computational processing of large-scale algebraic systems. Furthermore, the incredible expansion experienced by the existing computational hardware and software has made amenable to effective treatment problems of an ever increasing diversity and complexity, posed by engineering and scientific applications.

Parallel computing is outstanding among the new computational tools, especially at present when further increases in hardware speed apparently have reached insurmountable barriers. As it is well known, the main barriers for enhancing the efficiency of parallel computing are the difficulties associated with the coordination of the many processors that carry out the different tasks and also those associated with the information-transmission between them. Ideally, given a task, these difficulties disappear when procedures for carrying out that task are available such that the parallel processors work independently of each other. Thus, we will say that a parallel-processing algorithm fulfills the *'paradigm of parallel computing'* when the tasks assigned to different processors are independent of each other.

The emergence of parallel computing during the last 20 or 30 years, prompted on the part of the computational-modeling community a continued and systematic effort with the purpose of harnessing it for the endeavor of solving boundary-value problems (BVPs) of partial differential equations [1]. Very early after such an effort began, it was recognized that domain decomposition methods (DDM) were the most effective technique for applying parallel computing to the solution of partial differential equations, since such an approach drastically simplifies the coordination of the many processors that carry out the different tasks and also reduces very much the requirements of information-transmission between them. Ideally, DDMs intend producing algorithms such that "the *global* solution is obtained by solving *local* problems defined separately in each subdomain of a *coarse-mesh* – the *domain-decomposition*"; in what follows, such a condition is referred to as the *'DDM-paradigm'*. When the *DDM-paradigm* is satisfied full parallelization can be achieved by assigning each subdomain to a different processor.

When intensive DDM research began much attention was given to *overlapping* DDMs, but soon after attention shifted to substructuring methods in non-overlapping partitions, known as *non-overlapping* DDMs. This evolution seems natural when the *DDM-paradigm* is taken into account: it is easier to uncouple the local problems when the domain decomposition is non-overlapping. At present, however, numerically competitive algorithms need to incorporate *constraints*,

* Corresponding author.
E-mail address: iherrerarevilla@gmail.com (I. Herrera).

0955-7997/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.enganabound.2012.12.003

such as continuity on *primal-nodes* [2–6]. This poses an additional challenge for fulfilling the *DDM-paradigm* [7], which has been difficult to overcome.

Most of the work done in DDM refers to symmetric and positive definite problems. At present it is generally accepted that the most effective non-overlapping DDMs are BDDC and FETI-DP. The finite-element tearing and interconnecting method (FETI), was introduced by Farhat [8,9] and later modified by the introduction of dual-primal constraints [10–12], which originated the FETI-DP method. As for the balancing domain decomposition (BDD), it was originally introduced by Mandel [13,14] and more recently modified by Dohrmann who incorporated constraints in its formulation [3,4]; this resulted in BDDC (BDD with constraints).

To achieve full disconnection of the local problems, FETI is formulated in a product space of functions, W (see p.133 of [2]), which contains discontinuous functions. However, FETI does not work directly in such a space; instead, it avoids working in it by resourcing to an indirect approach based on Lagrange-multipliers. On the other hand, BDDC is more direct and does not resort to Lagrange multipliers, but it does not work in a product-space either in spite of the fact that it is there where full disconnection of the local problems is achieved (in Section 10, this point is further discussed). Thus, until the line of research here reported started to appear a direct treatment in a product-space had not been presented.

In standard approaches to DDMs two different processes can be identified: numerical discretization of partial differential equations and the design of strategies for achieving the DDM-paradigm: "solving the *global* BVP by solving *local* BVPs, exclusively". Standard DDM-frameworks generally do not separate these two processes and the difficulties associated with one and the other are combined. In particular, they work in linear spaces of functions.

However, discretization methods are a particular topic of numerical methods for partial differential equations that has received intensive attention since the development of electronic computers started and, at present, effective discretization methods are available almost for any well-posed BVP. Once a BVP has been discretized, parallelizing the processing of the resulting discrete system is essentially a purely algebraic problem.

Several years ago, Herrera and a research group at the National University of Mexico (UNAM) started a line of research [15–20] whose goal has been to fully develop a direct approach in which the work is done in a product-space and, furthermore, the discretization process and the domain decomposition procedures are thoroughly separated. The present paper is part of that line of research and is devoted to explain briefly (summarize) some of the most important results obtained so far.

A purely algebraic product-space – the *derived-vector space* (*DVS*) – has been constructed, which contains "continuous and discontinuous vectors" (i.e., algebraic images of continuous and discontinuous functions, respectively) and is very suitable for treating the discrete systems obtained by discretization of BVPs of a single equation, or systems of such equations. The DVS constitutes a finite-dimensional Hilbert-space with respect to a suitable inner-product whose definition is independent of the algebraic system of equations to be parallelized. Therefore, this *DVS-framework* is applicable to any symmetric, non-symmetric and non-definite (positive) matrix. Furthermore, this setting is suitable for both direct (*primal*) formulations, without recourse to Lagrange-multipliers, and indirect (*dual*) formulations, with recourse to Lagrange-multipliers. Another finding has been that, in the DVS-framework, four domain-decomposition algorithms preconditioned and subjected to constraints are possible; corresponding to two *primal* and two *dual* formulations. One primal and one dual algorithm of that set can be interpreted as versions of BDDC and of FETI-DP, respectively. However, nothing similar to the other two algorithms has been identified in the literature, albeit in the DVS setting these four formulations are closely related and can be easily derived from each other. The nomenclature adopted for the *DVS-algorithms* is *DVS-BDDC, DVS-FETI-DP, DVS-PRIMAL* and *DVS-DUAL*.

The *DDM-paradigm* is achievable using each one of the four *DVS-algorithms* and the numerical procedures that can be used to achieve it are explained in Section 8 of this paper. Work is underway to produce codes that can be used to operate efficiently some of the massively parallelized computers available today [21]. The numerical efficiency has been tested comparing standard FETI-DP with the DVS version of FETI-DP [16,17] and the results indicate that their performances are not significantly different. Comparisons presented in Section 9 of this paper between different DVS-algorithms also indicate that their numerical performances are similar. The numerical equivalence between FETI-DP and BDDC is well-known by now [22,23].

Although, two DVS-algorithms can be interpreted as DVS versions of BDDC and FETI-DP, at least two very important differences of the new versions should be highlighted: the DVS-versions are applicable to non-symmetric matrices and, also, they achieve the *DDM-paradigm*. As it is well-known, certain number of applications of FETI-DP and BDDC have been made to non-symmetric and indefinite problems (see, for example, [6,24–27]), but all them have been case specific, while the DVS-algorithms are generic algorithms that can be applied independently of the origin of the problem. As a matter of fact, once the discretized problem is known the very same matrix-formulas can be *blindly* applied (in the sense that the formulas indicate with precision the operations that have to be performed) to the discrete systems originated by a single partial differential equation, or a system of such equations, which in turn may be formally symmetric, non-symmetric or indefinite. Furthermore, this framework can be used to formulate and discuss in a systematic manner a general theory of DDMs for non-symmetric and indefinite problems, as it has been started to do in the work here summarized.

About the other difference mentioned above, standard versions of FETI-DP and BDDC do not achieve the *DDM-paradigm* [7] and their applicability to operate efficiently the huge massively parallelized computers that exist today is hindered by this fact. On the other hand, because the *DVS-algorithms do* achieve the *DDM-paradigm* (as it is shown in Section 8) they are very suitable for that purpose.

The organization of the paper follows. In Section 3, the problem to be treated is introduced. Section 2 motivates *derived-nodes*, while Sections 4 and 5 are devoted to study them and *derived-vectors*, respectively. The equivalent problem in the *derived-vector space* is given in Section 6, and Section 7 is devoted to the (preconditioned) DVS-algorithms with constrains derived from it. In Section 8, where the numerical procedures are explained, it is shown that the DVS-algorithms satisfy the DDM-paradigm. The numerical results reported in this paper are given in Section 9, while Section 10 is devoted to compare the DVS-framework with standard DDMs. Finally, the conclusions of the paper are presented in Section 11. To simplify the presentation, many of the definitions required have been collected in an Appendix at the end of the paper.

2. A non-overlapping system of nodes

The *'derived vector space framework (DVS-framework)'* starts with the system of linear equations that is obtained after the partial differential equation or system of such equations has been discretized, independently of the discretization method used. This system of discrete equations is referred to as the *'original-problem'*

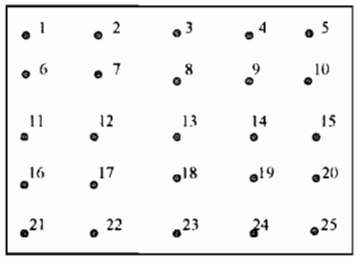

Fig. 1. The '*original nodes*'.

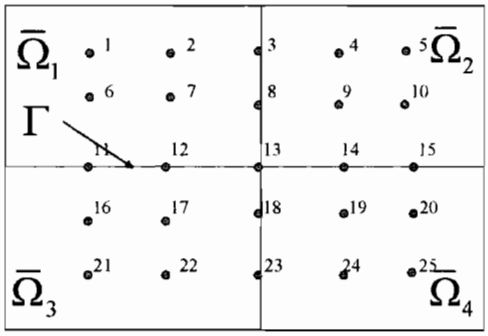

Fig. 2. The *original nodes* in the coarse-mesh.

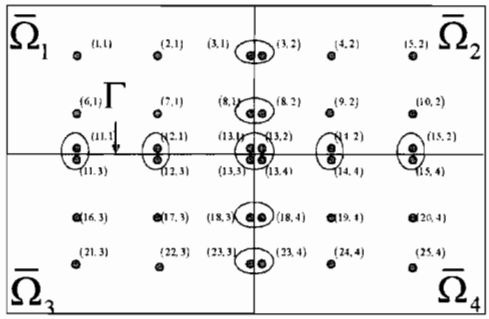

Fig. 3. The *mitosis*.

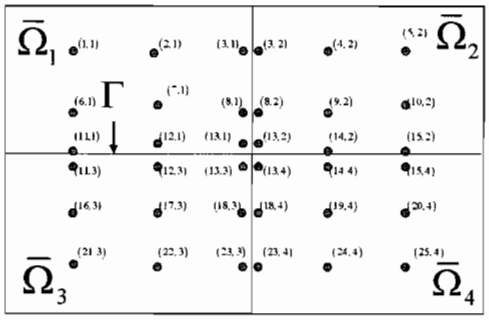

Fig. 4. The *derived-nodes* distributed in the *coarse-mesh*.

and the nodes used in this process are referred to as the *'original-nodes'* (Fig. 1). In domain decomposition methods (DDMs) it is standard to introduce a domain-partition, called *coarse-mesh*. Generally, some of the original-nodes belong to more than one partition-subdomain (Fig. 2). So, although the domain-decomposition is non-overlapping the partition of the nodes is overlapping.

It would be advantageous for achieving the *DDM-paradigm*, if each node would belong to one and only one partition-subdomain, and in the *DVS-framework* a new set of nodes – the *'derived-nodes'* – that enjoy this property is introduced. This new set of nodes is obtained by dividing each original-node (Fig. 3) in as many pieces as required to assign one and only one node-piece to each one of the subdomains (Fig. 4). Then, such node-pieces (called: *'derived-nodes'*) are identified by means of an ordered-pair of numbers: the label of the *original-node*, it comes from, followed by the *partition-subdomain* label, it was assigned to. A *'derived-vector'* is simply defined to be any real-valued function[1] defined in the whole set of *derived-nodes*; the set of all *derived-vectors* constitutes a linear space: the *'derived-vector space (DVS)'*. Then, for each pair of *derived-vectors an* inner-product (the *Euclidean inner-product'*) is defined in the usual manner: as the product of their components summed over all the *derived-nodes*. The DVS constitutes a finite-dimensional Hilbert-space, with respect to such an *inner-product*. We observe that this inner-product definition is independent of the system-matrix, which has not even mentioned thus far.

A new problem, the *DVS-problem*, defined in the *derived-vector space* that is equivalent to the *original problem*, is introduced. Of course, the matrix of this new problem is different to the *original-matrix*, which is only defined in the *original-vector space*, and the theory supplies a formula for deriving it [15]. From there on, in the DVS-framework, all the work is done in the derived-vector space and one never goes back to the original vector-space. In a systematic manner, this framework led to the construction of four preconditioned *DVS-algorithms*: two *primal* formulations (i.e., direct formulations that do not use Lagrange multipliers) and two *dual* formulations (i.e., indirect formulations that do use Lagrange multipliers). Soon after [15], the first *primal* formulation was recognized to be a version of the BDDC (the *balancing domain decomposition with constraints* of Mandel and Dohrmann), while the first *dual* formulation was recognized to be a version of the FETI-DP (the *dual-primal finite element tearing and interconnecting* of Farhat). Therefore, the notation that will be used for them here is:

(a) The *DVS-version of BDDC*;
(b) The *DVS-version of FETI-DP*;
(c) *DVS-primal algorithm*; and
(d) *DVS-dual algorithm*.

All these algorithms are preconditioned and are formulated in vector-spaces subjected to constraints; so, they are preconditioned and constrained algorithms.

3. The original problem

It will be assumed that the system of linear algebraic equations:

$$\widehat{\underline{\underline{A}}}\,\widehat{\underline{u}} = \widehat{\underline{f}} \tag{3.1}$$

is the discretized version of a boundary-value problem (BVP) corresponding to a (linear) partial differential equation, or system of such equations. We observe that the developments that follow are independent of both, the specific BVP considered and the discretization method used; so, both of them will remain unspecified throughout. Instead, we adopt a set of assumptions (axioms) under which our results (the *DVS-framework*) will be applicable. The system of algebraic equations of Eq. (3.1) will be referred to as the *'original problem'* and will be the starting point of our discussions. In this system, the *'original matrix'* $\widehat{\underline{\underline{A}}}$ and the

[1] For the treatment of systems of equations, vector-valued functions are considered, instead.

'*original vectors*' $\underline{\widehat{u}}$ and $\underline{\widehat{f}}$ are:

$$\underline{\underline{\widehat{A}}} \equiv \left(\widehat{A}_{ij}\right),\ \underline{\widehat{u}} \equiv (\widehat{u}_j),\ \underline{\widehat{f}} \equiv \left(\widehat{f}_j\right) \text{ with } i,j = 1,\ldots,N \tag{3.2}$$

here, N is the number of '*original nodes*'.

Let Ω be the domain of definition of the continuous problem, before discretization. Then, it is assumed that a *coarse-mesh* is introduced, which constitutes a non-overlapping domain decomposition of Ω. This is a family $\{\Omega_1,\ldots,\Omega_E\}$ of sub-domains of Ω that fulfills the relations

$$\Omega_\alpha \cap \Omega_\beta = \emptyset \text{ and } \Omega \subset \bigcup_{\alpha=1}^{E} \overline{\Omega}_\alpha \tag{3.3}$$

here, $\overline{\Omega}_\alpha$ stands for the closure of Ω_α. Furthermore, it is assumed that the sets of numbers:

$$\hat{N} \equiv \{1,\ldots,N\} \text{ and } \hat{E} \equiv \{1,\ldots,E\} \tag{3.4}$$

label the *original nodes* and the partition-subdomains, respectively. The real-valued functions defined in $\hat{N} = \{1,\ldots,N\}$ constitute a vector-space that will be denoted by $\widehat{W}$ and referred to as the '*original vector-space*'. The notation $\hat{N}^\alpha$, $\alpha = 1,\ldots,E$, will be used for the subset of *original-nodes* that pertain to $\overline{\Omega}_\alpha$. *Original-nodes* will be classified into '*internal*' and '*interface-nodes*': a node is *internal* if it belongs to only one partition-subdomain closure and it is an *interface-node*, when it belongs to more than one. Furthermore, for the development of domain decomposition methods with *constraints* it will be necessary to choose a subset of *interface-nodes*: the set of *primal nodes*. Interface-nodes that are not primal are said to be *dual-nodes*.

Using the notations thus far introduced, the following assumption is adopted:

Axiom 1. "When the indices $p \in \hat{N}^\alpha$ and $q \in \hat{N}^\beta$ are *internal*, then p and q are *unconnected*". We recall that *unconnected* means:

$$\widehat{A}_{pq} = \widehat{A}_{qp} = 0, \text{ whenever } \alpha \neq \beta \tag{3.5}$$

4. Derived-nodes

As said before, when a *coarse-mesh* is introduced some of the nodes used in the original discretization belong to more than one partition-subdomain and to overcome this inconvenience, in the DVS-framework, another set of nodes is introduced: the '*derived-nodes*'. The general developments are better understood, through a simple example that we explain first.

Consider the set of 25 nodes used in the discretización of a boundary-value problem shown in Fig. 1. We recall that the developments that follow are independent of both, the BVP considered and the discretization method used. After such a discretization is carried out a *coarse-mesh* that consists of four subdomains, as shown in Fig. 2, is introduced. Thus, we have a set of nodes and a family of subdomains, which are numbered using of the index-sets: $\hat{N} \equiv \{1,\ldots,25\}$ and $\hat{E} \equiv \{1,2,3,4\}$, respectively. Then, the sets of nodes corresponding to such a *non-overlapping* domain decomposition is actually *overlapping*, since the four subsets

$$\begin{aligned} &\hat{N}^1 \equiv \{1,2,3,6,7,8,11,12,13\}, && \hat{N}^2 \equiv \{3,4,5,8,9,10,13,14,15\} \\ &\hat{N}^3 \equiv \{11,12,13,16,17,18,21,22,23\}, && \hat{N}^4 \equiv \{13,14,15,18,19,20,23,24,25\} \end{aligned} \tag{4.1}$$

are not disjoint (see, Fig. 2). Indeed, for example:

$$\hat{N}^1 \cap \hat{N}^2 = \{3,8,13\} \tag{4.2}$$

In order to obtain a "truly" non-overlapping decomposition, we replace the set of '*original nodes*' by another set: the set of '*derived-nodes*'; a '*derived node*' is defined to be a pair of numbers: (p,α), where p corresponds a node that belongs to $\overline{\Omega}_\alpha$. In symbols: a '*derived node*' is a pair of numbers (p,α) such that $p \in \hat{N}_\alpha$. We denote by X the set of all *derived nodes*

$$X \equiv \left\{(p,\alpha) | \alpha \in \hat{E} \text{ and } p \in \hat{N}^\alpha\right\} \tag{4.3}$$

We observe that the total number of *derived-nodes* is 36 (Fig. 3) while that of *original-nodes* is 25. In order to minimize repetitions in the developments that follow, where we deal extensively with *derived-nodes*, the notation (p,α) is reserved for pairs such that $(p,\alpha) \in X$; Then, we assign to each subdomain Ω_α a unique '*local set of derived-nodes*':

$$X^\alpha \equiv \{(p,\alpha)\} \tag{4.4}$$

Taking α successively as 1, 2, 3 and 4, we obtain a family of four subsets, $\{X^1, X^2, X^3, X^4\}$, Fig. 4, which is a *truly disjoint* decomposition of the set of *derived-nodes* X, in the sense that each one of the derived-nodes belongs to one and only one *coarse-mesh* subdomains; i.e.,

$$X = \bigcup_{\alpha=1}^{E} X^\alpha \text{ and } X^\alpha \cap X^\beta = \varnothing, \text{ when } \alpha \neq \beta \tag{4.5}$$

Of course, the cardinality (i.e., the number of members) of each one of these subsets is 36/4 equal to 9. Given any $p = 1,\ldots,25$, the set of derived-nodes that derive from p is given by (see, Fig. 3)

$$Z(p) \equiv \left\{(p,\alpha) | p \in \overline{\Omega}_\alpha\right\} \tag{4.6}$$

The '*multiplicity*' $m(p)$ of any *original-node* $p \in \hat{N}$ is defined to be the *cardinality* of the set $Z(p)$. A *derived-node*, (p,α), is classified as *internal*, *interface*, *primal* and *dual*, depending on whether p is *internal*, *interface*, *primal* and *dual*, respectively. The notations: $I \subset X$, $\Gamma \subset X$, $\pi \subset X$ and $\Delta \subset X$ are adopted for the corresponding sets of derived-nodes, respectively. Furthermore, $\Pi \equiv I \cup \pi$.

The above discussion had the sole purpose of illustrating the role played by *derived nodes*, as well as some notation to be used. The formal developments were introduced and discussed in detail in previous papers [15–20] (we draw mainly from [15]).

5. The "derived vector-space (DVS)"

By a '*derived-vector*' we mean a function defined in the set of *derived-nodes*, X. In general the values of such functions, at each derived-node, may be chosen to be vectors of $\mathbb{R}^n$; the choice $n = 1$ (real-valued functions) permits treating single-equation problems and $n > 1$ systems of partial differential equations.

The set of *derived-vectors* constitute a linear space W: the '*derived-vector space*'. Corresponding to each *local subset of derived-nodes* X^α, there is a '*local subspace of derived-vectors*', $W^\alpha \subset W$, which is defined by the condition that vectors of W^α vanish at every *derived-node* that does not belong to X^α. An important property of the subspaces W^α should be observed:

$$W = W^1 \oplus \ldots \oplus W^E \tag{5.1}$$

In words: the space W is the direct sum of the family of subspaces $\{W^1,\ldots,W^E\}$.

For every pair of vectors, $\underline{u} \in W$ and $\underline{w} \in W$, the '*Euclidean inner product*' is defined to be

$$\underline{u} \bullet \underline{w} = \sum_{(p,\alpha) \in X} \underline{u}(p,\alpha) \odot \underline{w}(p,\alpha) \tag{5.2}$$

here, the symbol $\odot$ stands for the standard inner-product of $\mathbb{R}^n$-vectors; i.e.,

$$\underline{u}(p,\alpha) \odot \underline{w}(p,\alpha) \equiv \sum_{i=1}^{n} \underline{u}(p,\alpha,i) \odot \underline{w}(p,\alpha,i) \tag{5.3}$$

It should be observed that this definition of the *Euclidean inner product* is independent of the *original-matrix* $\widehat{\underline{\underline{A}}}$, which can be non-symmetric or indefinite. When $n=1$, members of the derived-vector space are real-valued function. Then, Eq. (5.2) reduces to

$$\underline{u}\bullet\underline{w}=\sum_{(p,\alpha)\in X}\underline{u}(p,\alpha)\,\underline{w}(p,\alpha) \tag{5.4}$$

Another significant property of the *derived-vector space* W is that it constitutes a finite dimensional *Hilbert-space* with respect to the Euclidean inner product whose definition is independent of the *original-matrix* $\widehat{\underline{\underline{A}}}$, which can be non-symmetric or indefinite. Due to this property the algorithms derived in the *DVS-framework* are also applicable when the *original-matrix* $\widehat{\underline{\underline{A}}}$ is non-symmetric or indefinite.

A *derived-vector*, $\underline{u}$, is said to be *'continuous'* when $\underline{u}(p,\alpha)$ is independent of α, for every $(p,\alpha)\in X$. The subset of $W_{12}\subset W$, of *continuous* vectors, constitutes a linear subspace of W. The *natural injection*, $R:\widehat{W}\to W$, of $\widehat{W}$ into W, is defined by the condition that, for every $\widehat{\underline{u}}\in\widehat{W}$, one has

$$(R\widehat{\underline{u}})(p,\alpha)=\widehat{\underline{u}}(p),\ \forall(p,\alpha)\in X \tag{5.5}$$

Clearly, $R\widehat{\underline{u}}\in W$ so defined is continuous for every $\widehat{\underline{u}}\in\widehat{W}$. Furthermore, it can be seen that $R:\widehat{W}\to W$ defines a *bijection* of $\widehat{W}$ in W_{12}; thus we write $R^{-1}:W_{12}\to\widehat{W}$ for the inverse of R, when restricted to W_{12}.

The subspace $W_{11}\subset W$ is defined to be the orthogonal complement of W_{12}, with respect to the Euclidean inner-product. In this manner the space W is decomposed into two orthogonal complementary subspaces: W_{11} and W_{12}, which fulfill

$$W=W_{11}\oplus W_{12} \tag{5.6}$$

Two matrices $\underline{\underline{a}}:W\to W$ and $\underline{\underline{j}}:W\to W$ are now introduced; they are the orthogonal-projection operators, with respect to the *Euclidean inner-product*, on W_{12} and W_{11}, respectively. The first one will be referred to as the *'average operator'* and the second one will be the *'jump operator'*. If $\underline{u}\in W_{11}$, then $\underline{\underline{a}}\,\underline{u}=0$; i.e., vectors of $W_{11}\subset W$ are *'zero-average vectors'*. We observe that in view of Eq. (5.6), every *derived-vector*, $\underline{u}\in W$, can be written in a unique manner as the sum of a *zero-average vector* plus a *continuous vector*; indeed:

$$\underline{u}=\underline{u}_{11}+\underline{u}_{12}\ \text{ with }\begin{cases}\underline{u}_{11}\equiv\underline{\underline{j}}\,\underline{u}\in W_{11}\\ \underline{u}_{12}\equiv\underline{\underline{a}}\,\underline{u}\in W_{12}\end{cases} \tag{5.7}$$

Next, we define several subspaces of W that will be used in the sequel. They are W_I, W_Γ, W_π, W_Δ and W_Π; vectors that belong to each one of these subspaces, vanish at every derived-node does not belong to I,Γ,π,Δ and Π, respectively. Furthermore, to specify the restrictions used in the algorithms here discussed, a subspace $W_r\subset W$ is introduced, which is assumed to have the property:

$$W_r\equiv W_I\oplus\underline{\underline{a}}^{r}W_\pi\oplus W_\Delta \tag{5.8}$$

here, $\underline{\underline{a}}^{r}$ stands for the projection operator on W_r. The linear subspaces introduced above satisfy:

$$W=W_I\oplus W_\Gamma=W_I\oplus W_\pi\oplus W_\Delta\ \text{ and }\ W_r=W_\Pi\oplus W_\Delta \tag{5.9}$$

Thus far, only dual-primal restrictions have been implemented. In that case, W_r is defined by $W_r\equiv W_I+\underline{\underline{a}}W_\pi+W_\Delta$.

6. The DVS-problem with constraints

A proof, of the following result as well a definition of the matrix $\underline{\underline{A}}:W_r\to W_r$ here used, is given in Appendix "**A**" of [15].

"A vector $\widehat{\underline{u}}\in\widehat{W}$ is solution of the *original problem*, if and only if, $\underline{u}'=R\widehat{\underline{u}}\in W$ fulfills the equalities:

$$\underline{\underline{a}}\,\underline{\underline{A}}\,\underline{u}'=\underline{f}\ \text{ and }\ \underline{\underline{j}}\,\underline{u}'=0 \tag{6.1}$$

The vector $\underline{f}\equiv\left(R\hat{f}\right)\in W_{12}\subset W_r$, will be written as $\underline{f}\equiv\underline{f}_\Pi+\underline{f}_\Delta$, with $\underline{f}_\Pi\in W_\Pi$ and $\underline{f}_\Delta\in W_\Delta$. We recall the definition of the *natural injection* of Eq. (5.5).

We remark that this problem is formulated in W: the *derived-vector space*. In what follows, the matrix $\underline{\underline{A}}$, when restricted to W_r (i.e., $\underline{\underline{A}}:W_r\to W_r$), is assumed to be invertible. In many cases this can be granted when a sufficiently large number of *primal-nodes*, adequately located, are taken. Let $\underline{u}'\in W_r$ be solution of it, then $\underline{u}'\in W_{12}\subset W$ necessarily, since $\underline{\underline{j}}\,\underline{u}'=0$, and one can apply the inverse of the *natural injection* to obtain

$$\hat{\underline{u}}=R^{-1}\underline{u}' \tag{6.2}$$

Thereby, we observe that the mapping R^{-1} is only defined when $\underline{u}'$ is a *continuous vector* (i.e., $\underline{u}'\in W_{12}$). The condition $\underline{u}'\in W_{12}$ can only be granted when no rounding errors are present. Due to this fact, in numerical applications Eq. (6.2) should be replaced by:

$$\widehat{\underline{u}}=R^{-1}\underline{\underline{a}}\,\underline{u}' \tag{6.3}$$

Indeed, when $\underline{u}'$ is not continuous, $\underline{\underline{a}}\,\underline{u}'$ is the *continuous vector* closest to $\underline{u}'$ (here, 'closest' is with respect to the *Euclidean distance*).

Before finishing this Section we observe that Eq. (6.1) is a key element of the *DVS-framework*, because it supplies a formulation of the *original-problem* in the *derived-vector space* in which the nodes have been partitioned into disjoint packages of nodes, which in turn permit decomposing the matrix into fully independent sub-matrices that can be processed in different processors with negligible coordination and communication between them.

7. The DVS-algorithms with constraints

The matrix $\underline{\underline{A}}$ of Eq. (6.1) can be written as (see [8,9]):

$$\underline{\underline{A}}=\begin{pmatrix}\underline{\underline{A}}_{\Pi\Pi} & \underline{\underline{A}}_{\Pi\Delta}\\ \underline{\underline{A}}_{\Delta\Pi} & \underline{\underline{A}}_{\Delta\Delta}\end{pmatrix} \tag{7.1}$$

Using this notation, we define the 'Schur-complement matrix with constraints', by

$$\underline{\underline{S}}\equiv\underline{\underline{A}}_{\Delta\Delta}-\underline{\underline{A}}_{\Delta\Pi}\underline{\underline{A}}^{-1}_{\Pi\Pi}\underline{\underline{A}}_{\Pi\Delta} \tag{7.2}$$

Furthermore, in what follows:

$$\underline{f}_\Delta\equiv\left(\widehat{R\underline{f}}\right)_\Delta-\underline{\underline{A}}_{\Delta\Pi}\underline{\underline{A}}^{-1}_{\Pi\Pi}\left(\widehat{R\underline{f}}\right)_\Pi \tag{7.3}$$

Then, in a simple and direct manner Eq. (6.1) is transformed into

$$\underline{\underline{a}}\,\underline{\underline{S}}\,\underline{u}_\Delta=\underline{f}_\Delta\ \text{ and }\ \underline{\underline{j}}\,\underline{u}_\Delta=0 \tag{7.4}$$

together with

$$\widehat{\underline{u}}=\tilde{R}^{-1}\underline{\underline{a}}\left\{\left(\underline{\underline{I}}-\underline{\underline{A}}^{-1}_{\Pi\Pi}\underline{\underline{A}}_{\Pi\Delta}\right)\underline{u}_\Delta+\underline{\underline{A}}^{-1}_{\Pi\Pi}\left(\widehat{R\underline{f}}\right)_\Pi\right\} \tag{7.5}$$

Generally, DDM-algorithms are classified into *primal* and *dual* algorithms. The first of these classes includes those that are direct, without recourse to Lagrange-multipliers, and the second one those that use such multipliers. However, the *DVS-framework* supplies a very general *primal* setting that permits including both classes in it, but to do that the guidelines for their formulations

are changed. Actually, each one of the algorithms is completely determined by the *sought-information*; i.e., the information that the algorithms seek for in an immediate manner. The *sought-information* may be chosen in several alternative manners; all what is required is that the solution of the *DVS-problem* can be derived from it, in an *inexpensive* manner (computationally wise).

7.1. The DVS version of the BDDC algorithm

The *sought-information* is defined to be $\underline{u}_{\Delta} \in W_{\Delta}$. With this choice a version of the *DVS-BDDC algorithm* is obtained [8]: Find $\underline{u}_{\Delta} \in W_{\Delta}$ such that

$$\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\underline{S}}\underline{u}_{\Delta} = \underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{f}_{\Delta} \text{ and } \underline{\underline{j}}\underline{u}_{\Delta} = 0 \tag{7.6}$$

Once $\underline{u}_{\Delta} \in W_{\Delta}$ has been obtained, $\widehat{\underline{u}} \in \widehat{W}$ solution of Eq. (3.1) is given by

$$\widehat{\underline{u}} = R^{\sim 1}\underline{\underline{a}}\left\{\left(\underline{\underline{I}} - \underline{\underline{A}}_{\Pi\Pi}^{-1}\underline{\underline{A}}_{\Pi\Delta}\right)\underline{u}_{\Delta} + \underline{\underline{A}}_{\Pi\Pi}^{\sim 1}\left(\widehat{R\underline{f}}\right)_{\Pi}\right\} \tag{7.7}$$

7.2. The DVS version of FETI-DP algorithm

The *sought-information* is chosen to be $\underline{\lambda} \equiv -\underline{\underline{j}}\underline{\underline{S}}\underline{u}$. With this choice a version of the *DVS-FETI-DP algorithm* is obtained [15]: Given $\underline{f}_{\Delta} \in \underline{\underline{a}}W_{\Delta}$, find $\underline{\lambda} \in W_{\Delta}$ such that

$$\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\lambda} = \underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_{\Delta} \text{ and } \underline{\underline{a}}\underline{\lambda} = 0 \tag{7.8}$$

Once $\underline{\lambda} \in \underline{\underline{j}}W_{\Delta}$ has been obtained, $\underline{u}_{\Delta} \in \underline{\underline{a}}W_{\Delta}$ is given by:

$$\underline{u}_{\Delta} = \underline{\underline{a}}\underline{\underline{S}}^{-1}\left(\underline{f}_{\Delta} - \underline{\lambda}\right) \tag{7.9}$$

Then, Eq. (7.7) can be applied to obtain $\widehat{\underline{u}} \in \widehat{W}$ solution of Eq. (3.1).

7.3. The DVS primal-algorithm

The *sought-information* is chosen to be $\underline{v}_{\Delta} \equiv \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{u}$. This algorithm consists in searching for a derived-vector $\underline{v}_{\Delta} \in W_{\Delta}$, which fulfills

$$\underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{v}_{\Delta} = -\underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_{\Delta} \text{ and } \underline{\underline{a}}\underline{\underline{S}}\underline{v}_{\Delta} = 0 \tag{7.10}$$

Once $\underline{v}_{\Delta} \in \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}W_{\Delta}$ has been obtained, then

$$\underline{u}_{\Delta} = \underline{\underline{a}}\left(\underline{\underline{S}}^{-1}\underline{f}_{\Delta} + \underline{v}_{\Delta}\right) \tag{7.11}$$

and Eq. (7.7) can be applied to obtain $\widehat{\underline{u}} \in \widehat{W}$ solution of Eq. (3.1).

7.4. The DVS dual-algorithm

The *sought-information* is chosen to be $\underline{\mu} \equiv \underline{\underline{S}}\underline{u}$. This algorithm consists in searching for a function $\underline{\mu} \in W_{\Delta}$, which fulfills

$$\underline{\underline{S}}\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\mu} = \underline{\underline{S}}\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{f}_{\Delta} \text{ and } \underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\mu} = 0 \tag{7.12}$$

Once $\underline{\mu} \in \underline{\underline{a}}\underline{\underline{S}}^{-1}W_{\Delta}$ has been obtained, $\underline{u}_{\Delta} \in \underline{\underline{a}}W_{\Delta}$ is given by:

$$\underline{u}_{\Delta} = \underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\mu}_{\Delta} \tag{7.13}$$

and Eq. (7.7) can be applied to obtain $\widehat{\underline{u}} \in \widehat{W}$ solution of Eq. (3.1).

8. Numerical procedures

The numerical experiments were carried out with each one of the four preconditioned *DVS-algorithms* with constraints enumerated in Section 7; they are:

$$\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\underline{S}}\underline{u}_{\Delta} = \underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{f}_{\Delta} \text{ and } \underline{\underline{j}}\underline{u}_{\Delta} = 0;\ \text{DVS-BDDC} \tag{8.1}$$

$$\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\lambda} = \underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_{\Delta} \text{ and } \underline{\underline{a}}\underline{\lambda} = 0;\ \text{DVS-FETI-DP} \tag{8.2}$$

$$\underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{v}_{\Delta} = \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_{\Delta} \text{ and } \underline{\underline{a}}\underline{\underline{S}}\underline{v}_{\Delta} = 0;\ \text{DVS-PRIMAL-2} \tag{8.3}$$

and

$$\underline{\underline{S}}\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\mu} = \underline{\underline{S}}\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_{\Delta} \text{ and } \underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\mu} = 0;\ \text{DVS-DUAL-2} \tag{8.4}$$

8.1. Comments on the DVS numerical procedures

The outstanding uniformity of the formulas given in Eqs. (8.1)–(8.4) yields clear advantages for code development, especially when such codes are built using object-oriented programming techniques. Such advantages include:

I. The construction of very robust codes. This is an advantage of the *DVS-algorithms*, which stems from the fact the definitions of such algorithms exclusively depend on the discretized system of equations (which will be referred to as the *original problem*) that is obtained by discretization of the partial differential equations considered, but that is otherwise independent of the problem that motivated it. In this manner, for example, essentially the same code was applied to treat 2-D and 3-D problems; indeed, only the part defining the geometry had to be changed, and that was a very small part of it;
II. The codes may use different local solvers, which can be direct or iterative solvers;
III. Minimal modifications are required for transforming sequential codes into parallel ones; and
IV. Such formulas also permit to develop codes in which "the *global*-problem-solution is obtained by exclusively solving *local problems*".

This last property must be highlighted, because it makes the DVS-algorithms very suitable as a tool to be used in the construction of massively parallelized software, which is needed for efficiently programming the most powerful parallel computers available at present. Thus, procedures for constructing codes possessing Property IV are outlined and analyzed next.

All the DVS-algorithms of Eqs. (8.1) to (8.4) are iterative and can be implemented with recourse to Conjugate Gradient Method (CGM), when the matrix is definite and symmetric, or some other iterative procedure such as GMRES, when that is not the case. At each iteration step, one has to compute the action on a *derived-vector* of one of the following matrices: $\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\underline{S}}$, $\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}$, $\underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}$ or $\underline{\underline{S}}\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{a}}$, depending on the *DVS-algorithm* that is applied. Such matrices in turn are different permutations of the matrices $\underline{\underline{S}}$, $\underline{\underline{S}}^{-1}$, $\underline{\underline{a}}$ and $\underline{\underline{j}}$. Thus, to implement any of the preconditioned *DVS-algorithms*, one only needs to separately develop codes capable of computing the action of one of the matrices $\underline{\underline{S}}$, $\underline{\underline{S}}^{-1}$, $\underline{\underline{a}}$ or $\underline{\underline{j}}$ on an arbitrary vector of W, the *derived-vector-space*. Therefore, next we separately explain how to compute the application of each one of the matrices $\underline{\underline{S}}$ and $\underline{\underline{S}}^{-1}$. As for $\underline{\underline{a}}$ and $\underline{\underline{j}}$, as it will be seen, their application requires exchange of information between derived-nodes that are *descendants* of the same *original-node* and that is a very simple operation for which such exchange of information is minimal.

8.2. Application of $\underline{\underline{S}}$

From Eq. (7.2), we recall the definition of the matrix $\underline{\underline{S}}$:

$$\underline{\underline{S}} \equiv \underline{\underline{A}}_{\Delta\Delta} - \underline{\underline{A}}_{\Delta\Pi}\underline{\underline{A}}_{\Pi\Pi}^{-1}\underline{\underline{A}}_{\Pi\Delta} \tag{8.5}$$

In order to evaluate the action of $\underline{\underline{S}}$ on any *derived-vector*, we need to successively evaluate the action of the following matrices $\underline{\underline{A}}_{\Pi\Delta}$, $\underline{\underline{A}}^{\sim 1}_{\Pi\Pi}$, $\underline{\underline{A}}_{\Delta\Pi}$ and $\underline{\underline{A}}_{\Delta\Delta}$. Nothing special is required except for $\underline{\underline{A}}^{\sim 1}_{\Pi\Pi}$, which is explained next.

We have

$$\underline{\underline{A}}_{\Pi\Pi} \equiv \begin{pmatrix} \underline{\underline{A}}_{\mathrm{II}} & \underline{\underline{A}}_{\mathrm{I}\pi} \\ \underline{\underline{A}}_{\pi\mathrm{I}} & \underline{\underline{A}}_{\pi\pi} \end{pmatrix} = \begin{pmatrix} \underline{\underline{A}}^{t}_{\mathrm{II}} & \underline{\underline{A}}^{t}_{\mathrm{I}\pi}\underline{\underline{a}}^{r} \\ \underline{\underline{a}}^{r}\underline{\underline{A}}^{t}_{\pi\mathrm{I}} & \underline{\underline{a}}^{r}\underline{\underline{A}}^{t}_{\pi\pi}\underline{\underline{a}}^{r} \end{pmatrix} \tag{8.6}$$

Let $\underline{w} \in W$, be an arbitrary *derived-vector*, and write

$$\underline{v} \equiv \underline{\underline{A}}^{-1}_{\Pi\Pi}\underline{w} \tag{8.7}$$

Then, $\underline{v}_{\pi} \in W(\pi)$ is characterized by

$$\sigma_{\pi\pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)\underline{v}_{\pi} = \underline{w}_{\pi} - \underline{\underline{A}}_{\pi\mathrm{I}}\underline{\underline{A}}^{\sim 1}_{\mathrm{II}}\underline{w}_{\mathrm{I}}, \text{ subjected to } \underline{\underline{j}}^{\pi}\underline{v}_{\pi} = 0 \tag{8.8}$$

and can obtained iteratively. Here,

$$\sigma_{\pi\pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right) \equiv \left\{\underline{\underline{A}}_{\pi\pi} - \underline{\underline{A}}_{\pi\mathrm{I}}\underline{\underline{A}}^{\sim 1}_{\mathrm{II}}\underline{\underline{A}}_{\mathrm{I}\pi}\right\} \tag{8.9}$$

and, with $\underline{\underline{a}}^{\pi}$ as the projection-matrix into $W_r(\pi)$,

$$\underline{\underline{j}}^{\pi} \equiv \underline{\underline{\mathrm{I}}} - \underline{\underline{a}}^{\pi} \tag{8.10}$$

We observe that the computation in parallel of the action of $\underline{\underline{A}}^{-1}_{\mathrm{II}}$ is straightforward because

$$\underline{\underline{A}}^{-1}_{\mathrm{II}} = \sum_{\alpha=1}^{E}\left(\underline{\underline{A}}^{\alpha}_{\mathrm{II}}\right)^{-1} \tag{8.11}$$

Once $\underline{v}_{\pi} \in W_r(\pi)$ has been obtained, to derive $\underline{v}_{\mathrm{I}}$ one can apply:

$$\underline{v}_{\mathrm{I}} = \underline{\underline{A}}^{-1}_{\mathrm{II}}\left(\underline{w}_{\mathrm{I}} - \underline{\underline{A}}_{\mathrm{I}\pi}\underline{v}_{\pi}\right) \tag{8.12}$$

This completes the evaluation of $\underline{\underline{S}}$.

8.3. Application of $\underline{\underline{S}}^{-1}$

We define

$$\Sigma \equiv \mathrm{I} \cup \Delta \tag{8.13}$$

A property that is relevant for the following discussion is:

$$W_r(\Sigma) = W(\Sigma) \tag{8.14}$$

Therefore, the matrix $\underline{\underline{A}}^{-1}$ can be written as:

$$\underline{\underline{A}}^{-1} = \begin{pmatrix} \left(\underline{\underline{A}}^{-1}\right)_{\Pi\Pi}\left(\underline{\underline{A}}^{-1}\right)_{\Pi\Delta} \\ \left(\underline{\underline{A}}^{-1}\right)_{\Delta\Pi}\left(\underline{\underline{A}}^{-1}\right)_{\Delta\Delta} \end{pmatrix} = \begin{pmatrix} \left(\underline{\underline{A}}^{-1}\right)_{\Sigma\Sigma}\left(\underline{\underline{A}}^{-1}\right)_{\Sigma\pi} \\ \left(\underline{\underline{A}}^{-1}\right)_{\pi\Sigma}\left(\underline{\underline{A}}^{-1}\right)_{\pi\pi} \end{pmatrix} \tag{8.15}$$

Then, $\underline{\underline{S}} : W_{\Delta} \to W_{\Delta}$ fulfills

$$\underline{\underline{S}}^{-1} = \left(\underline{\underline{A}}^{-1}\right)_{\Delta\Delta} \tag{8.16}$$

For any $\underline{w} \in W$, let us write

$$\underline{v} \equiv \underline{\underline{A}}^{-1}\underline{w} \tag{8.17}$$

Then, $\underline{v}_{\pi}$ fulfills

$$\sigma_{\pi\pi}\left(\underline{\underline{A}}\right)\underline{v}_{\pi} = \underline{w}_{\pi} - \underline{\underline{A}}_{\pi\Sigma}\left(\underline{\underline{A}}^{t}_{\Sigma\Sigma}\right)^{-1}\underline{w}_{\Sigma}, \text{ subjected to } \underline{\underline{j}}^{r}\underline{v}_{\pi} = 0 \tag{8.18}$$

here, $\underline{\underline{j}}^{r} \equiv \underline{\underline{\mathrm{I}}} - \underline{\underline{a}}^{r}$, where the matrix $\underline{\underline{a}}^{r}$ is the projection operator on W_r, while

$$\sigma_{\pi\pi}\left(\underline{\underline{A}}\right) \equiv \underline{\underline{A}}_{\pi\pi} - \underline{\underline{A}}_{\pi\Sigma}\left(\underline{\underline{A}}^{t}_{\Sigma\Sigma}\right)^{-1}\underline{\underline{A}}_{\Sigma\pi} \tag{8.19}$$

Furthermore, we observe that

$$\left(\underline{\underline{A}}^{t}_{\Sigma\Sigma}\right)^{-1} = \sum_{\alpha=1}^{E}\left(\underline{\underline{A}}^{\alpha}_{\Sigma\Sigma}\right)^{-1} \tag{8.20}$$

Eq. (8.18) is solved iteratively. Once $\underline{v}_{\pi}$ has been obtained, we apply:

$$\underline{v}_{\Sigma} = \left(\underline{\underline{A}}^{t}_{\Sigma\Sigma}\right)^{-1}\left(\underline{w}_{\Sigma} - \underline{\underline{A}}_{\Sigma\pi}\underline{v}_{\pi}\right) \tag{8.21}$$

This procedure permits obtaining $\underline{\underline{A}}^{-1}\underline{w}$ in general; however, what we need only is $\left(\underline{\underline{A}}^{-1}\right)_{\Delta\Delta}\underline{w}$. We observe that

$$\left(\underline{\underline{A}}^{-1}\right)_{\Delta\Delta}\underline{w} = \left(\underline{\underline{A}}^{-1}\underline{w}_{\Delta}\right)_{\Delta} \tag{8.22}$$

The vector $\underline{\underline{A}}^{-1}\underline{w}_{\Delta}$ can be obtained by the general procedure presented above. Thus, take $\underline{w} \equiv \underline{w}_{\Delta} \in W_{\Delta} \subset W$ and

$$\underline{v} \equiv \underline{\underline{A}}^{-1}\underline{w}_{\Delta} \tag{8.23}$$

Therefore,

$$\underline{v}_{\mathrm{I}} + \underline{v}_{\Delta} = \underline{v}_{\Sigma} = -\left(\underline{\underline{A}}^{t}_{\Sigma\Sigma}\right)^{-1}\underline{\underline{A}}_{\Sigma\pi}\underline{v}_{\pi} = -\left(\underline{\underline{A}}^{t}_{\Sigma\Sigma}\right)^{-1}\underline{\underline{A}}^{t}_{\Sigma\pi}\underline{\underline{a}}^{r}\underline{v}_{\pi} \tag{8.24}$$

Using Eq. (8.20), these operations can be fully parallelized. However, the detailed discussion of such procedures will be presented separately [21].

8.4. Application of $\underline{\underline{a}}$ and $\underline{\underline{j}}$.

We use the notation

$$\underline{\underline{a}} = \left(a_{(i,\alpha)(j,\beta)}\right) \tag{8.25}$$

Then [16]:

$$a_{(i,\alpha)(j,\beta)} = \frac{1}{m(i)}\delta_{ij}, \ \forall\alpha \in \mathrm{Z}(i) \text{ and } \forall\beta \in \mathrm{Z}(j) \tag{8.26}$$

While

$$\underline{\underline{j}} = \underline{\underline{\mathrm{I}}} - \underline{\underline{a}} \tag{8.27}$$

Therefore,

$$\underline{\underline{j}}\underline{w} = \underline{w} - \underline{\underline{a}}\underline{w}, \text{ for every } \underline{w} \in W \tag{8.28}$$

As for the right hand-sides of Eqs. (7.6), (7.8), (7.10) and (7.12), all they can be obtained by successively applying to $\underline{f}_{\Delta}$ some of the operators that have already been discussed. Recalling Eq. (7.3), we have

$$\underline{f}_{\Delta} \equiv \left(\widehat{R\underline{f}}\right)_{\Delta} - \underline{\underline{A}}_{\Delta\Pi}\underline{\underline{A}}^{-1}_{\Pi\Pi}\left(\widehat{R\underline{f}}\right)_{\Pi} \tag{8.29}$$

The computation of $\widehat{R\underline{f}}$ does present any difficulty and the evaluation of the actions of $\underline{\underline{A}}^{-1}_{\Pi\Pi}$ and $\underline{\underline{A}}_{\Delta\Pi}$ were already analyzed.

9. Numerical results

All the partial differential equations treated had the form:

$$\begin{aligned} &-a\nabla^2 u + \underline{b}\cdot\nabla u + cu = f(x), \ \ x \in \Omega \\ &u = g(x), \ \ x \in \partial\Omega \\ &\Omega = \prod_{i=1}^{d}(\alpha_i, \beta_i) \end{aligned} \tag{9.1}$$

where a,c are constants, while $b = (b_1,\ldots,b_{\dim})$ is a constant vector and $\dim = 2,3$. In the applications we present, n is equal to the number of degrees of freedom (*dof*) of the *original problem*; we use linear functions and only one of them is associated with each *original* node. As for the *original problems* treated, they have the standard form of Eq. (3.1):

$$\widehat{A}\cdot\widehat{u} = \widehat{f} \tag{9.2}$$

They were obtained by discretization of three different differential equations, in two and three dimensions, of the above boundary value problem with $a=1$. The selection of the *primal*

Table 1

Symmetric .2-D	Eps=1 e−6						
	Subdomains	Dof	Primal	DVS-BDDC	DVS-primal	DVS-FETI-DP	DVS-dual
(2×2) × (2×2)	4	9	1	2	1	2	1
(4×4) × (4×4)	16	225	9	7	7	6	5
(6×6) × (6×6)	36	1225	25	9	9	7	6
(8×8) × (8×8)	64	3969	49	10	10	9	7
(10×10) × (10×10)	100	9801	81	11	11	10	8
(12×12) × (12×12)	124	20,449	121	12	11	13	9
(14×14) × (14×14)	196	38,025	169	12	12	13	12
(16×16) × (16×16)	256	65,025	225	13	12	14	12
(18×18) × (18×18)	324	104,329	289	13	13	15	13
(20×20) × (20×20)	400	159,201	361	13	13	15	14
(22×22) × (22×22)	484	233,289	441	13	14	15	16
(24×24) × (24×24)	576	330,625	529	14	14	15	15
(26×26) × (26×26)	676	455,625	625	14	14	15	15
(28×28) × (30×30)	784	613,089	729	14	14	15	15
(30×30) × (30×30)	900	808,201	841	15	14	15	15

Table 2

Non-symmetric 2-D	Eps=1 e−6						
	Subdomains	Dof	Primal	DVS-BDDC	DVS-primal	DVS-FETI-DP	DVS-dual
(2×2) × (2×2)	4	9	1	2	1	2	1
(4×4) × (4×4)	16	225	9	8	6	6	6
(6×6) × (6×6)	36	1225	25	10	8	8	7
(8×8) × (8×8)	64	3969	49	12	10	9	9
(10×10) × (10×10)	100	9801	81	13	12	9	10
(12×12) × (12×12)	124	20,449	121	14	12	10	11
(14×14) × (14×14)	196	38,025	169	15	13	11	11
(16×16) × (16×16)	256	65,025	225	15	14	11	12
(18×18) × (18×18)	324	104,329	289	16	14	11	12
(20×20) × (20×20)	400	159,201	361	16	15	12	12
(22×22) × (22×22)	484	233,289	441	17	16	12	12
(24×24) × (24×24)	576	330,625	529	17	16	12	13
(26×26) × (26×26)	676	455,625	625	17	16	13	13
(28×28) × (30×30)	784	613,089	729	18	17	13	13
(30×30) × (30×30)	900	808,201	841	18	17	13	13

constraints is crucial for the performance of the algorithms. In the numerical examples they were chosen according to Algorithm "*D*" of Toselli and Widlund (p. 173 of [2]). Choosing $\underline{b}=0$ symmetric matrices were obtained, which corresponds to the choices $c=1$ and to $c=0$, respectively; in this latter case, the differential operator treated is the Laplacian. The choices $\underline{b}=(1,1)$ and $\underline{b}=(1,1,1)$, with $c=0$, yield non-symmetric matrices. As for the right-hand side, the following choices were made:

- For the Laplacian operator

$$\begin{cases} f \equiv \sin(n\pi x)\sin(n\pi y), \text{ in 2D} \\ f \equiv \sin(n\pi x)\sin(n\pi y)\sin(n\pi z), \text{ in 3D} \end{cases} \tag{9.3}$$

- Other differential operators

$$\begin{cases} f \equiv \exp(xy)\{1-x^2-y^2\}, \text{in 2D} \\ f \equiv \exp(xyz)\{yz(2+xyz)+x^3(y^2+z^2)\}, \text{ in 3D} \end{cases} \tag{9.4}$$

Discretization was accomplished using central finite differences, but streamline artificial viscosity was incorporated in the treatment when $\underline{b}\neq 0$. The DQGMRES algorithm was implemented for the iterative solution of the non-symmetric problems. The *restrictions* used were continuity on the *primal* nodes. The numerical results that were obtained are reported in six tables that follow. Tables 1–3 refer to 2D problems while Tables 4–6 to 3D problems.

10. Comparisons with standard BDDC and FETI-DP

The DVS-framework has been developed with the sole intention of contributing to further the effective application of parallel hardware to the solution of partial differential equations, especially elliptic equations but order to evaluate the possible merits of the formulations discussed in this paper, it is mandatory to compare them with well-established algorithms such as BDDC and FETI-DP, as we do in this Section. As seen in previous pages, the DVS-framework yields, in addition to a non-standard setting for DDMs non-standard algorithms as well. Therefore, these two aspects will be included in the comparisons that follow.

We think the relation between the DVS-framework and the FETI-DP setting is easy to understand, because it can be summarized as follows:

"FETI is formulated in a product space of functions, W (see p.133 of [2]), which contains discontinuous functions. However, FETI does not work directly in such a space; instead, it avoids working in it by resourcing to Lagrange-multipliers. On the contrary, the DVS-framework precisely consists in constructing a product space – called *derived-vector space* and denoted by W – in a discrete (finite-dimensional) setting, which contains

Table 3

Laplacian 2-D	Eps=1 e−6						
	Subdomains	Dof	Primal	DVS-BDDC	DVS-primal	DVS-FETI-DP	DVS-dual
((2×2) × (2×2)	4	9	1	1	1	1	1
(4×4) × (4×4)	16	225	9	4	3	5	5
(6×6) × (6×6)	36	1225	25	7	6	6	8
(8×8) × (8×8)	64	3969	49	7	7	7	7
(10×10) × (10×10)	100	9801	81	8	8	9	8
(12×12) × (12×12)	124	20,449	121	8	9	10	9
(14×14) × (14×14)	196	38,025	169	9	9	10	9
(16×16) × (16×16)	256	65,025	225	9	9	10	9
(18×18) × (18×18)	324	104,329	289	9	9	10	9
(20×20) × (20×20)	400	159,201	361	9	9	10	10
(22×22) × (22×22)	484	233,289	441	9	9	10	10
(24×24) × (24×24)	576	330,625	529	10	9	11	10
(26×26) × (26×26)	676	455,625	625	10	9	11	11
(28×28) × (30×30)	784	613,089	729	10	10	11	11
(30×30) × (30×30)	900	808,201	841	10	10	11	11

Table 4

Symmetric 3-D			Eps 1 e−6		Number of iterations		
Partition	Subdomains	Dof	Primals	DVS-BDDC	DVS-Primal	DVS-FETI-DP	DVS-dual
(2×2×2)×(2×2×2)	8	27	7	2	2	2	2
(3×3×3)×(3×3×3)	27	512	80	4	4	3	3
(4×4×4)×(4×4×4)	64	3375	351	5	5	4	3
(5×5×5)×(5×5×5)	125	13,824	1024	6	5	4	3
(6×6×6)×(6×6×6)	216	42,875	2375	6	6	4	4
(7×7×7)×(7×7×7)	343	110,592	4752	7	6	4	4
(8×8×8)×(8×8×8)	512	250,047	8575	8	7	5	6
(9×9×9)×(9×9×9)	8019	512,000	14,336	8	8	7	7
(10×10×10)×(10×10×10)	10,000	970,299	22,599	8	8	8	8

"continuous and discontinuous vectors" (i.e., algebraic images of continuous and discontinuous functions, respectively) and exploring the consequences of working directly in it.

It is relatively straightforward to see that the conceptual framework is greatly simplified in this latter approach. In this respect it is important to distinguish, in the procedures for solving partial differential equations using highly parallelized hardware, two different aspects: discretization of partial differential equations and designing strategies for achieving the DDM-paradigm: "solving the *global* BVP by solving *local* BVPs, exclusively". Standard frameworks generally do not separate these two processes and the difficulties associated with one and the other are combined. In the DVS-framework, on the contrary, these two processes are clearly separated and one works directly with the finite system of discrete equations that has been obtained after discretization.

To illustrate in a more specific and direct manner the advantages that the DVS-framework yields, we compare next the standard version of BDDC with its DVS-version:

10.1. The Standard Bddc

The notation of the standard BDDC [3–6,10] will be used here:

$$M^{-1}Su = M^{-1}f \tag{10.1}$$

where S and the preconditioner M^{-1} are

$$S = \sum_{i=1}^{N} \overline{R}_i{}^T S_i \overline{R}_i \ \text{ and } \ M^{-1} = \sum_{i=1}^{N} R^T{}_i S_i{}^{-1} R_i \tag{10.2}$$

respectively. Furthermore, N is the number of subdomains and for each $i=1,\ldots,N$

$$S_i = A^i{}_{\Gamma\Gamma} - A^i{}_{\Gamma I}\left(A^i{}_{II}\right)^{-1} A^i{}_{I\Gamma}, \ \text{for each } i=1,\ldots,N \tag{10.3}$$

$R_i{:}\Gamma \to \Gamma_i$ is the restriction operator from Γ into Γ_i; when applied to a function defined in Γ, it yields its restriction to Γ_i. As for $\overline{R}_i$, $\overline{R}_i : \Gamma \to \Gamma_i$ is given by $\overline{R}_i \equiv D_i R_i$. Here, $D_i = \mathrm{diag}\{\delta_i\}$ is a diagonal matrix defining a partition of unity. Substituting S and M^{-1} in Eq. (10.1), we obtain

$$\left(\sum_{i=1}^{N}\left(D_i{}^{-1}\overline{R}_i\right)^T S_i{}^{-1} D_i{}^{-1}\overline{R}_i\right)\left(\sum_{i=1}^{N}\overline{R}_i{}^T S_i \overline{R}_i\right)u = \left(\sum_{i=1}^{N}\left(D_i{}^{-1}\overline{R}_i\right)^T S_i{}^{-1} D_i{}^{-1}\overline{R}_i\right)f \tag{10.4}$$

This equation is to be compared with our Eq. (9.1). For the purpose of comparison, the vectors u and f of Eq. (10.4) can be identified with vectors $\underline{u}$ and $\underline{f}$ of our *original space*, W. Furthermore, we apply our *natural injection*, $R : W \to W$, defined by Eq. (5.5), to Eq. (9.1) and premultiply the resulting equation also by the *natural injection*, with $\underline{u}$ and $\underline{f}_{\Delta}$ replaced by $R\underline{u}$ and $R\underline{f}$, respectively. In this manner we obtain

$$R\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\underline{S}}R\underline{u} = \underline{\underline{a}}\underline{\underline{S}}^{-1}R\underline{f} \tag{10.5}$$

We have verified that indeed Eqs. (10.4) and (10.5) are equivalent.

Table 5

Non-symmetric 3-D			Eps 1 e−6		Number of iterations		
Partition	Subdomains	Dof	Primals	DVS-BDDC	DVS-PRIMAL	DVS-FETI-DP	DVS-dual
$(2\times2\times2)\times(2\times2\times2)$	8	27	7	3	2	2	2
$(3\times3\times3)\times(3\times3\times3)$	27	512	80	6	4	4	4
$(4\times4\times4)\times(4\times4\times4)$	64	3375	351	7	6	5	5
$(5\times5\times5)\times(5\times5\times5)$	125	13,824	1024	8	7	5	5
$(6\times6\times6)\times(6\times6\times6)$	216	42,875	2375	10	7	6	6
$(7\times7\times7)\times(7\times7\times7)$	343	110,592	4752	11	8	6	6
$(8\times8\times8)\times(8\times8\times8)$	512	250,047	8575	11	9	7	7
$(9\times9\times9)\times(9\times9\times9)$	8019	512,000	14,336	12	10	8	8
$(10\times10\times10)\times(10\times10\times10)$	10,000	970,299	22,599	13	11	9	9

Table 6

Laplacian 3-D			Eps 1 e−6		Number of iterations		
Partition	Subdomains	Dof	Primals	DVS-BDDC	DVS-primal	DVS-FETI-DP	DVS-dual
$(2\times2\times2)\times(2\times2\times2)$	8	27	7	1	1	1	1
$(3\times3\times3)\times(3\times3\times3)$	27	512	80	3	2	2	2
$(4\times4\times4)\times(4\times4\times4)$	64	3375	351	1	1	2	2
$(5\times5\times5)\times(5\times5\times5)$	125	13,824	1024	5	4	5	4
$(6\times6\times6)\times(6\times6\times6)$	216	42,875	2375	6	6	5	6
$(7\times7\times7)\times(7\times7\times7)$	343	110,592	4752	6	6	6	6
$(8\times8\times8)\times(8\times8\times8)$	512	250,047	8575	7	7	8	8
$(9\times9\times9)\times(9\times9\times9)$	8019	512,000	14,336	8	9	9	9
$(10\times10\times10)\times(10\times10\times10)$	10,000	970,299	22,599	10	10	10	10

10.2. Comparison of the Standard and DVS Versions of Bddc

The DVS-framework, and therefore also the DVS-version of BDDC, starts with the matrix that is obtained after the problem has been discretized and for its application does not require any information about the system of partial differential equations from which it originated. Throughout all the developments it is assumed that the *dual-primal Schur-complement matrix* $\underline{\underline{S}}$, defined in Section 7, Eq. (7.2), is non-singular.

When comparing the DVS-BDDC with the standard BDDC we encountered some substantial differences. For example, when the inverses of the local Schur-complements exist, which is granted by choosing the *primal-nodes* adequately, in the *DVS framework* the inverse of $\underline{\underline{S}}^t$ is given by (see [5,6]):

$$\left(\underline{\underline{S}}^t\right)^{-1}=\sum_{\alpha=1}^{N}\left(\underline{\underline{S}}^{\alpha}\right)^{-1} \tag{10.6}$$

A similar relation does not hold for the BDDC algorithm. Indeed, in this latter approach, we have instead:

$$S=\sum_{\alpha=1}^{N}\overline{R}_{\alpha}^{T}S_{\alpha}\overline{R}_{\alpha} \tag{10.7}$$

and

$$(S)^{-1}\neq\sum_{\alpha=1}^{N}\left(\overline{R}_{\alpha}^{T}S_{\alpha}\overline{R}_{\alpha}\right)^{-1} \tag{10.8}$$

even when the inverses of the local Schur-complements exist and no restrictions are used.

The origin of this problem, encountered in the BDDC formulation, may be traced back to the fact that the BDDC approach does not work directly in the product space. Indeed, one frequently goes back to degrees of freedom associated with the *original nodes*. This is done by means of the restriction operators $R_i:\Gamma\rightarrow\Gamma_i$ which can be interpreted as transformations of the *original vector-space* into the *product vector-space* (or, *derived vector-space*). If the algorithm of Eq. (10.4) is analyzed from this point of view, it is seen that it repeatedly goes from the *original vector-space* to the product-space (or *derived vector-space*) and back. For example, consider the expression:

$$\left(\sum_{i=1}^{N}\left(D_i^{-1}\overline{R}_i\right)^{T}S_i^{-1}D_i^{-1}\overline{R}_i\right)\left(\sum_{i=1}^{N}\overline{R}_i^{T}S_i\overline{R}_i\right)u$$
$$=\left(\sum_{i=1}^{N}\left(D_i^{-1}\overline{R}_i\right)^{T}S_i^{-1}D_i^{-1}\overline{R}_i\right)\left(\sum_{i=1}^{N}\overline{R}_i^{T}S_i\overline{R}_iu\right) \tag{10.9}$$

occurring in Eq. (10.4). After starting with the vector u in the *original vector-space*, we go to the *derived-vector space* with $\overline{R}_iu$ and remain there when we apply S_i. However, we go back to the *original vector-space* when $\overline{R}_i^T$ is applied. A similar analysis can be made of the term

$$\left(D_i^{-1}\overline{R}_i\right)^{T}S_i^{-1}D_i^{-1}\overline{R}_i. \tag{10.10}$$

Summarizing, in the operations indicated in Eq. (10.9) four trips between the *original vector space* and the *derived-vector space* were made, two one way and the other two in the way back. In the *DVS-framework*, on the other hand, from the start the *original problem* is transformed into one defined in *derived-vector space*, where all the work is done afterwards, and then such trips become unnecessary. Thereby, the matrix formulas are simplified and so is code development. The unification and simplification achieved in this manner, permits producing more effective and robust software. This explains in part, why the DVS-BDDC algorithm achieves the DDM-paradigm in spite of the fact that standard versions of it do not.

In summary, some of the advantages of the DVS-framework as a setting for domain decomposition methods are:

1. It constitutes a unique setting for noh-overlapping domain decomposition methods that is applicable to any well-posed boundary-value problem of an elliptic partial differential equation, or system of such equations. All what is required is that, after discretization, the system matrix satisfy Axiom 1, Eq. (3.5). Hence, the conceptual unification achieved is very significant;

2. Such a setting has permitted formulating a family of four algorithms of general applicability: the *DVS-algorithms*. For member of that family possesses the following properties: the very same algorithm is applicable to a single-equation and to a system of equations. When applied to a matrix, the condition of being symmetric positive-definite is not required;
3. The DVS-framework can be applied independently of the methods and function-spaces used in the discretization. Also, the DDM-formulations may be *primal-formulations* (without Lagrange multipliers) and *dual-formulations* (with Lagrange multipliers). In particular, both FETI-DP and BDDC are formulated in the same setting;
4. Furthermore, the four DVS-algorithms can be implemented with codes that achieve the DDM-paradigm. To our knowledge this is the first time that the DVS-paradigm has been achieved [7].

11. Conclusions

In this paper, the main purpose of domain decomposition methods has been summarized by means of the *DDM-paradigm*: "to solve the *global* BVP by solving *local* boundary-value problems (BVPs), exclusively". Also, arguments have been presented sustaining the view that in order to achieve it, it is necessary to completely disconnect the local problems from each other. This requires formulating the problems in a product space that contains discontinuous functions.

At present, of the two most effective DDMs, FETI-DP method is indeed formulated in a product function space. However, FETI does not work directly in such a space of discontinuous functions, because it avoids doing so by recourse to Lagrange-multipliers. As for BDDC, although it is a more direct formulation its setting is not a space of discontinuous functions, as it was explained with some detail in Section 10. On the other hand, standard formulations of DDMs address simultaneously the problems of numerical discretization and parallel processing, in spite of the fact that effective discretization methods are available for almost any BVP.

In view of the above, a line of research has been carried out to explore an approach in which non-overlapping DDMs are formulated in a purely algebraic space that contains both "continuous and discontinuous vectors" (i.e., algebraic images of continuous and discontinuous functions) and the most important results so far obtained in it are summarized in the present paper. They are:

- A purely algebraic setting – the *derived-vector space* – has been constructed, which contains "continuous and discontinuous vectors" (i.e., algebraic images of continuous and discontinuous functions, respectively);
- The *derived-vector space* is a finite dimensional Hilbert space with respect to an inner product whose definition is independent of the system-matrix considered. In particular, the (non-singular) system-matrix may be any; i.e., symmetric, non-symmetric and indefinite (non-positive-definite). Therefore, this finite dimensional Hilbert space supplies a unified setting for non-overlapping DDMs that simplifies much not only their formulation but the methods themselves. Furthermore, this framework can be used to formulate and discuss in a general and systematic manner the theory of DDMs for non-symmetric and indefinite problems, as it has been started to do in the work here summarized;
- The processes of numerical discretization and parallel algorithms construction, which in standard settings are mixed, have been thoroughly separated;
- A unified theory of non-overlapping DDMs has been obtained, which permits treating single-partial-differential-equations and systems of such equations that may be non-symmetric or indefinite (non-positive-definite);
- In the setting of the derived-vector space four algorithms (the DVS-algorithms) of general applicability have been developed. For each one of them, the following is true: the very same algorithm is applicable a single-differential equation and to a system of differential equations independently of whether it is symmetric, non-symmetric or indefinite; furthermore, independently of the method used in the discretization of the BVP;
- Each one of the *DVS-algorithms* achieves the *DDM-paradigm*, in the sense that, for its implementation, codes can be developed that solve the global BVP by solving local BVPs, exclusively (this is shown in Section 8). In particular, standard versions of FETI-DP and BDDC do not achieve the *DDM-paradigm* [7] and their applicability to operate efficiently the huge massively parallelized computers that exist today is hindered by this fact. On the other hand, because the *DVS-algorithms do* achieve the *DDM-paradigm* (as it is shown in Section 8) they are very suitable for that purpose.

A further remark is timely; at present effective discretization methods are available for almost any well-posed BVP, while domain decomposition methods are a more specialized topic whose study has not been as extensive. Therefore, the availability of DDMs that can be applied to the discrete system independently of the discretization method used is especially valuable.

The answers to the questions opened by the line of research here presented are, however, far from exhausted. Much more numerical experiments and analysis are being carried out to answer many of the questions still opened.

Algorithm nomenclature

- Primal #1 → DVS-BDDC.
- Primal #2 → DVS-Primal.
- Dual #1 → DVS-FETI-DP.
- Dual #2 → DVS-Dual.

Appendix. On Notations

General

1. *Original-problem*, the original discretized version of the problem;
2. *Original-node* is any node that was used to obtain the original discretized version of the boundary-value problem;
3. *Original-vector* is any function defined in the whole set of original-nodes;
4. $\hat{N} \equiv \{1,...,N\}$ and $\hat{E} \equiv \{1,...,E\}$ sets of natural numbers used to label the original-nodes and the subdomains of the *coarse-mesh*, respectively;
5. $\hat{N}^{\alpha} \subset \hat{N}$, $\alpha = 1,...,E$, the subset of $\hat{N}$ corresponding to nodes of $\overline{\Omega}_{\alpha}$;
6. *Derive-node* is an ordered pair of numbers such that the first one labels an original-node and the second one labels any of the subdomains of the *coarse-mesh* to which the original-node pertains;
7. X is the whole set of derived-nodes;
8. $X^{\alpha} \subset X$, $\alpha = 1,\ldots,E$, the set of derived-nodes associated with Ω_{α};
9. For each original node p, $Z(p) \subset X$, is the set of derived-nodes that derived from p;
10. $I \subset X$, $\Gamma \subset X$, $\pi \subset X$ and $\Delta \subset X$, represent sets of derived-nodes; namely, they are *internal*, *interface*, *primal* and *dual* derived-nodes, respectively;

11. $\Pi \equiv I \cup \pi$;
12. A *derived-vector* is any function (scalar or vector-valued) defined in X;
13. The *derived-vector space*, W, is the whole set of derived-vectors;
14. For every pair of vectors, $\underline{u} \in W$ and $\underline{w} \in W$, the '*Euclidean inner product*' is defined to be

$$\underline{u} \bullet \underline{w} = \sum_{(p,\alpha) \in X} \underline{u}(p,\alpha)\underline{w}(p,\alpha)$$

In applications to systems of equations, $\underline{u}(p,\alpha)$ itself is a vector and this equation is replaced by

$$\underline{u} \bullet \underline{w} = \sum_{(p,\alpha) \in X} \underline{u}(p,\alpha) \odot \underline{w}(p,\alpha)$$

here, $\underline{u} \odot \underline{w}$ means the inner product of the vectors involved;
15. The *DVS-problem*, is a problem formulated in the *derived-vector space*, which is equivalent to the original-problem;
16. $W^{\alpha} \subset W$, $\alpha = 1,...,E$, the set of derived-vectors, which vanish at each derived-node that is not associated with Ω_{α};
17. A derived-vector $\underline{u}$, is *continuous* when its value $\underline{u}(p,\alpha)$ is independent of α, at every derived-node (p,α);
18. $W_{12} \subset W$ is the linear subspace constituted by continuous derived-vectors;
19. $\underline{\underline{a}}$ is the orthogonal-projection matrix on the subspace of continuous vectors;
20. A derived-vector $\underline{u}$, has *zero-average* when $\underline{\underline{a}}\,\underline{u} = 0$ and the linear subspace $W_{11} \subset W$ is constituted by all zero-average derived-vectors;
21. $\underline{\underline{j}}$ is the orthogonal-projection matrix on the subspace of zero-average derived-vectors.

References

[1] DDM Organization, Proceedings of twenty one international conferences on domain decomposition methods. ⟨www.ddm.org⟩, 1988-2012.
[2] Toselli A, Widlund O. Domain decomposition methods, Algorithms and theory. Springer series in computational mathematics. Berlin: Springer-Verlag; 2005 p. 450.
[3] Dohrmann CR. A preconditioner for substructuring based on constrained energy minimization. SIAM J Sci Comput 2003;25(1):246–58.
[4] Mandel J, Dohrmann CR. Convergence of a balancing domain decomposition by constraints and energy minimization. Numer Linear Algebra Appl 2003;10(7):639–59.
[5] Mandel J, Dohrmann CR, Tezaur R. An algebraic theory for primal and dual substructuring methods by constraints. Appl Numer Math 2005;54:167–93.
[6] Da Conceição, D. T. Jr., Balancing domain decomposition preconditioners for non-symmetric problems, Instituto Nacional de Matemática Pura e Aplicada, Agencia Nacional do Petróleo PRH-32, Rio de Janeiro, May 9, 2006.
[7] Widlund O. Personal communication, 2010.
[8] Ch. Farhat, Roux F. A method of finite element tearing and interconnecting and its parallel solution algorithm. Int J Numer Methods Eng 1991;32: 1205–27.
[9] Mandel J, Tezaur R. Convergence of a substructuring method with Lagrange multipliers. Numer Math 1996;73(4):473–87.
[10] Farhat C, Lessoinne M, LeTallec P, Pierson K, Rixen D. FETI-DP a dual-primal unified FETI method, Part I: a faster alternative to the two-level FETI method. Int J Numer Methods Eng 2001;50:1523–44.
[11] Farhat C, Lessoinne M, Pierson K. A scalable dual-primal domain decomposition method. Numer Linear Algebra Appl 2000;7:687–714.
[12] Mandel J, Tezaur R. On the convergence of a dual-primal substructuring method. SIAM J Sci Comput 2001;25:246–58.
[13] Mandel J. Balancing domain decomposition. Commun Numer Methods Eng 1993;1:233–41.
[14] Mandel J, Brezina M. Balancing domain decomposition for problems with large jumps in coefficients. Math Comput 1996;65:1387–401.
[15] Herrera I, Carrillo-Ledesma A, Rosas-Medina Alberto. A brief overview of non-overlapping domain decomposition methods. Geofis Int 2011;50(4):445–63.
[16] Herrera, I. & Yates R. A. The multipliers-free dual primal domain decomposition methods for nonsymmetric matrices and their numerical testing. Numer Methods Part D E. 2011;27:1262–1289, http://dx.doi.org/10.1002/Num, (Published online April 28, 2010).
[17] Herrera, I. & Yates R. A. The multipliers-free domain decomposition methods. Numer Methods Part D. E. 26:874–905 July 2010, http://dx.doi.org/10.1002/num. (Published online Jan 28, 2009).
[18] Herrera I, Yates R. Unified multipliers-free theory of dual-primal domain decomposition methods. Numer Methods Part D E 2009;25:552–81, http://dx.doi.org/10.1002/num.20359 (Published online May 13, 08).
[19] Herrera I. New formulation of iterative substructuring methods without Lagrange multipliers: Neumann-Neumann and FETI. Numer Methods Part D E 2008;24(3):845–78 (Published online Sept 17, 2007) doi:10.1002 NO. 20293.
[20] Herrera I. Theory of differential equations in discontinuous piecewise-defined-functions. Numer Methods Part D E 2007;23(3):597–639 doi:10. 1002.NO.20182.
[21] De la Cruz, L.M, Herrera I. To be published.
[22] Brenner S, Sung L. BDDC and FETI-DP without matrices or vectors. Comput Methods Appl Mech Eng 2007;196(8):1429–35.
[23] Li J, Widlund O. FETI-DP, BDDC and block Cholesky methods. Int J Numer Methods Eng 2005;66:250–71.
[24] Cai X-C, Widlund OB. Domain decomposition algorithms for indefinite elliptic problems. SIAM J Sci Stat Comput 1992;13:243–58.
[25] Farhat C, Li J. An iterative domain decomposition method for the solution of a class of indefinite problems in computational structural dynamics. Elsevier Sci Direct Appl Numer Math 2005;54:150–66.
[26] Li J, Tu X. Convergence analysis of a balancing domain decomposition method for solving a class of indefinite linear systems. Numer Linear Algebra Appl 2009;16:745–73.
[27] Tu X, Li J. A balancing domain decomposition method by constraints for advection-diffusion problems. ⟨www.ddm.org/DD18/⟩.