Evidences that Software Based on Non-overlapping Discretization Is Most Efficient for Applying Highly Parallelized Supercomputers to Solving Partial Differential Equations

Ismael Herrera-Revilla(✉) and Iván Contreras

Instituto de Geofísica Universidad Nacional Autónoma de México, UNAM,
Ciudad de México, Mexico
iherrerarevilla@gmail.com

Abstract. One of the main problems for applying the highly parallelized supercomputers available today to computational physico-mathematical modeling of science and engineering is to develop software capable of effectively solving in parallel partial differential equations or systems of such equations. For this purpose much work on domain decomposition methods has been done. Recently, I. Herrera introduced a new 'non-overlapping discretization method' that for the application of domain decomposition methods has many advantages over standard methods of discretization. Based on theoretical grounds, some of these advantages have been indicated in previous publications. This paper, however, is devoted to present numerical evidences of such advantages and some of the outstanding parallelization-efficiencies that are feasible when domain decomposition methods are applied to the discrete system derived using non-overlapping discretization methods.

Keywords: Discretization-methods, Highly-parallelized super-computers · 100 %-parallel algorithms · Parallel-solution of pdes · Parallel software for elasticity · High performance computing · DDM

1 Introduction

Partial differential equations (PDEs) and systems of such equations are very important in science and engineering since their basic models are constituted by such equations [1]. Due to this fact progress in many fields of engineering and science heavily depends on the effective application of advanced computational tools to the solution of PDEs [2].

For this purpose much work on *domain decomposition methods* has been done [3–10]. In general, after a PDE has been discretized the effective application of parallel computing reduces to efficiently treating the matrix-equation that the discretization method yields. Recently, I. Herrera [11–17] introduced a new *'non-overlapping discretization method'* that for the application of *domain decomposition methods* has many advantages over standard methods of discretization. Methods of this class have the *conspicuous* feature that each node of the *fine-mesh* belongs to one and only one subdomain of the domain decomposition. Based on theoretical grounds, some of these advantages have been indicated in previous publications. This paper, however, is devoted to present numerical evidences of such advantages and some of the outstanding parallelization-efficiencies that are feasible when domain decomposition methods are applied to the discrete system derived using *non-overlapping discretization methods*.

Non-overlapping discretization methods have a wide range of applicability; in part, this stems from their axiomatic formulation. They are applicable to symmetric matrices -independently of whether or not they are (positive) definite- as well as to non-symmetric matrices. A general procedure developed in this line of research [11] permits transforming standard discretizations -defined on overlapping systems of nodes-, independently of the problems that originated them, into *non-overlapping discretizations*. Such a procedure is applicable to symmetric and non-symmetric matrices, as well as positive-definite or non-positive-definite ones. Because of all this, *non-overlapping discretizations* -and the concomitant *DVS-algorithms*- have arisen high expectations as a means for harnessing highly parallelized supercomputers to the task of solving the partial differential equations of science and engineering, but up to recently no software based on *non-overlapping discretizations* was available on which such expectations could be tested.

Recently, software based on *non-overlapping discretizations* has been carefully coded and applied using highly parallelized supercomputers [13] and tests for the DVS version of BDDC was carried out yielding parallelization-efficiencies close to 100 %. Such results constitute important evidences, which confirm that software based on *non-overlapping discretizations* is most efficient for applying highly parallelized supercomputers to resolve boundary-value problems of partial differential equations. Up to now only the case of symmetric and positive-definite matrices has been fully developed; however, the *non-overlapping discretization method* is also applicable to symmetric and non-symmetric matrices and the results so far obtained indicate that its development is a very worthwhile endeavor ahead.

This paper is devoted to present a summary of recent numerical experiments that verify the high parallelization efficiencies theoretically predicted. In particular, to avoid repetitions here the method is not explained in detail, but extensive background material on the *DVS* methodology is given in the references [11–17].

2 Non-overlapping Discretizations and DVS-Algorithms

In summary, this methodology consists of

I. A general method for transforming standard *overlapping discretizations* into *non-overlapping* ones;
II. Procedures to accelerate convergence:
    a. Restrictions and
    b. Preconditioning;
III. Definition of four DVS-algorithms; and
IV. Developing critical routes to construct highly efficient parallelization codes.

As said before, for the application of *DDMs* the *non-overlapping discretization methods* have many advantages over standard methods of discretization, because the matrix-equations they yield are better suited to be treated in parallel. The concomitant algorithms so obtained are known as *DVS-algorithms*. They are four; the construction of two of them is similar to *BDDC* and *FETI-DP*, albeit an essential difference is that the starting matrix-equation is obtained by application of a *non-overlapping discretization*. As for the other two, to our knowledge they are fully independent of previous developments reported in the literature. The DVS-algorithms also benefit of many algebraic properties that have been systematically established in previous work of this line of research [11–17].

3 Transforming a Discretization into a Non-overlapping One

In this Section, the general procedure for transforming a standard discretization method into a *non-overlapping* one is outlined. Consider the partial differential equation, or system of such equations:

$$\mathcal{L}u = f_\Omega \tag{1.1}$$

subjected to certain boundary conditions. Then, a mesh (*'the fine-mesh'*) is introduced and a standard (overlapping) method is applied. In this manner *'the original discretization'* is obtained:

$$\underline{\underline{M}}\,\underline{U} = \underline{F} \tag{1.2}$$

Thereafter, a domain-decomposition (or, *coarse-mesh*) is introduced. Generally, when this is done some of the nodes of the *fine-mesh* are shared by more than one subdomain. The procedure for transforming such an *overlapping* system of nodes into a *non-overlapping* one is summarized in Figs. 1a–d to 4. The *non-overlapping* nodes of Fig. 1d is the system that is used thereafter in the sequel and are referred to as *derived-nodes*. Furthermore, the symbol X is used for the total set of *derived-nodes*, and the functions defined on X are the *derived-vectors*. The general procedure for transforming the original overlapping-discretization into the *non-overlapping* one, consists essentially in defining an equivalent problem in the *space of derived-vectors (DVS)*. The reader is referred to [11–13] for details.

The non-overlapping decomposition of X is given by the family of subsets $X^\alpha \equiv \{X^\alpha | \alpha = 1, \ldots, E\}$, which for each $\alpha = 1, \ldots, E$, is constituted by the *derived-nodes* that belong to $\bar{\Omega}_\alpha$ and are defined by

$$X^\alpha \equiv \left\{(p, \alpha) | p \in \bar{\Omega}_\alpha\right\} \tag{1.3}$$

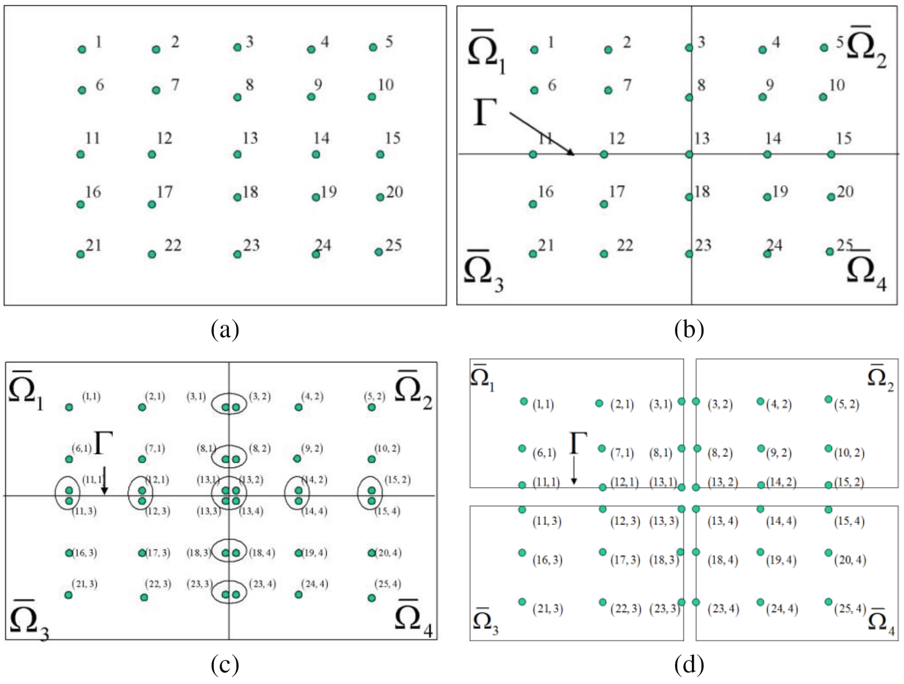

Fig. 1. Transformation of the nodes into a non-overlapping system

Clearly,

$$X = \bigcup_{\alpha=1}^{E} X^{\alpha} \, and \, X^{\alpha} \cap X^{\beta} = \emptyset, when \, \alpha \neq \beta \tag{1.4}$$

Given an *original-node* $p \in \overset{\frown}{X}$, the set of *'heirs of p'* is defined to be:

$$Z(p) \equiv \{(p, \alpha) | (p, \alpha) \in X\} \tag{1.5}$$

The multiplicity $m(p)$ of $p \in \overset{\frown}{X}$, is the cardinality of $Z(p)$; i.e., the number of heirs that p has. Additional notation and definitions are given in the Appendix, which may be complemented by those of the list of References.

4 The "Derived Vector-Space (DVS)"

Any function, real-valued or vector-valued, defined on the whole set $\overset{\frown}{X}$ of *original-nodes* is an *original-vector* and $\overset{\frown}{W}$ stands for the vector-space constituted by such vectors. Similarly, any function, real-valued or vector-valued, defined on the whole set

X of *derived-nodes* is a *derived-vector* and W stands for the vector-space constituted by such vectors. The value of a *derived-vector* at a node (p,α) is written as $\underline{u}(p,\alpha)$. When the value itself is a vector, $\underline{u}(p,\alpha,i)$ denotes the $i-th$ component of such a vector, which is a real-number.

Assume $\Lambda \subset \mathrm{X}$ is any subset of *derived-nodes*. Then, the notation $W(\Lambda)$ is used to represent the vector subspace of W, whose elements vanish at every *derived-node* that does not belong to Λ. Corresponding to each *local subset of derived-nodes* X^{α}, $\alpha = 1,\ldots,$ or E, there is a *'local subspace of derived-vectors'*, defined by $W^{\alpha} \equiv W(\mathrm{X}^{\alpha}) \subset W$. The space W is the direct sum of the family of subspaces $\{W^1,\ldots,W^E\}$; i.e.,

$$W = W^1 \oplus \ldots \oplus W^E \tag{1.6}$$

This is an important property, because it implies that every $\underline{u} \in W$ can be written uniquely as

$$\underline{u} = \underline{u}^1 + \ldots + \underline{u}^E, with\, \underline{u}^{\alpha} \in W^{\alpha}, \alpha = 1,\ldots,E \tag{1.7}$$

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

$$\underline{u} \bullet \underline{w} \equiv \sum_{(p,\alpha)\in \mathrm{X}} \underline{u}(p,\alpha) \odot \underline{w}(p,\alpha) = \sum_{\alpha=1}^{E} \sum_{p\in\bar{\Omega}_{\alpha}} \underline{u}(p,\alpha) \odot \underline{w}(p,\alpha) \tag{1.8}$$

Here, the symbol $\odot$ stands for the standard inner-product of $\mathbb{R}^n$ - vectors. When $n = 1$, Eq. (1.8) reduces to

$$\underline{u} \bullet \underline{w} = \sum_{(p,\alpha)\in \mathrm{X}} u(p,\alpha)w(p,\alpha) = \sum_{\alpha=1}^{E} \sum_{p\in\bar{\Omega}_{\alpha}} u(p,\alpha)w(p,\alpha) \tag{1.9}$$

We observe that the definition of *Euclidean inner product* is independent of the BVP considered, although it depends on the meshes that are introduced; both, the *fine* and *coarse meshes*.

5 Matrix Notations

Linear transformations of the space W into itself, and also of $\overset{\frown}{W}$ into itself, are considered in the *non-overlapping discretizations theory*. There is a one-to-one correspondence between such linear transformations and matrices. For matrices such as $\underline{\underline{M}}$, occurring in Eq. (1.2), which transform $\overset{\frown}{W}$ into itself, we use the notation:

$$\underline{\underline{M}} \equiv \left(M_{pq}\right), where\, p,q \in \overset{\frown}{\mathrm{X}}. \tag{1.10}$$

Matrices such as a matrix $\underline{\underline{A}}$ that will be introduced later, which transforms W into itself, we use the notation:

$$\underline{\underline{A}} \equiv \left(A_{(p,\alpha)(q,\beta)}\right), where (p, \alpha), (q, \beta) \in \mathrm{X} \tag{1.11}$$

6 Immersion of the Original-Vector Space

Some *derived-vectors* will be said to be *continuous*. A *derived-vector* $\underline{u}$ is *continuous* when for every p, $\underline{u}(p, \alpha)$ is independent of α. The subset of *continuous* vectors, $W_{12} \subset W$, constitutes a linear subspace of W.

Furthermore, there is a one-to-one mapping (see [11]) called the *natural immersion*, $R : \widehat{W} \to W$, of $\widehat{W}$ into W, and defined by the condition that, for every $\underline{u} \in \widehat{W}$, one has

$$\left(R\widehat{\underline{u}}\right)(p, \alpha) = \widehat{\underline{u}}(p), \forall (p, \alpha) \in \mathrm{X} \tag{1.12}$$

In addition, $R\widehat{W} = W_{12}$. Therefore, $R : \widehat{W} \to W$, when restricted to W_{12} has the an inverse $R^{-1} : W_{12} \to \widehat{W}$. Essentially what is done in the *DVS-method* is to formulate an equivalent problem in W_{12} and then apply $R^{-1} : W_{12} \to \widehat{W}$ to obtain the solution in the *original-space*.

To complete the scheme the orthogonal complement of W_{12} is introduced, so that the relation

$$W = W_{11} \oplus W_{12} \tag{1.13}$$

is fulfilled. Here, W_{11} is the above mentioned orthogonal-complement subspace.

The projections $\underline{\underline{a}}$ and $\underline{\underline{j}}$, on W_{12} and W_{11} respectively, are introduced. Clearly,

$$\underline{\underline{I}} = \underline{\underline{j}} + \underline{\underline{a}} \tag{1.14}$$

since $\underline{\underline{a}}$ and $\underline{\underline{j}}$ are complementary projections.

Using these results it can be seen that every *derived-vector*, $\underline{u} \in W$, can be written in a unique manner as:

$$\underline{u} = \underline{u}_{11} + \underline{u}_{12} with \begin{cases} \underline{u}_{11} \equiv \underline{\underline{j}}\underline{u} \in W_{11} \\ \underline{u}_{12} \equiv \underline{\underline{a}}\underline{u} \in W_{12} \end{cases} \tag{1.15}$$

Furthermore,

$$\underline{\underline{j}} = \underline{\underline{I}} - \underline{\underline{a}} \tag{1.16}$$

An explicit expression for $\underline{\underline{a}}$ is:

$$\underline{\underline{a}} = \left(a_{(p,\alpha)(q,\beta)}\right) with\, a_{(p,\alpha)(q,\beta)} = \frac{\delta_{pq}\delta_{\alpha\beta}}{m(q)} \tag{1.17}$$

7 The Non-overlapping Discretization with Constrains

In the development of the *non-overlapping discretization* methodology and the *DVS-algorithms* a general procedure for transforming any standard (overlapping) discretization into a non-overlapping one was introduced, which is briefly explained in this Section, for further details see [11–13], where additional references are given.

Although such a procedure has a wide range of applicability, including symmetric and non-symmetric matrices, there is an assumption that the *original-matrix* $\underline{\underline{M}}$ of Eq. (1.2) must fulfill and is here stated. To this end, we define

$$\delta_{pq}^{\alpha} \equiv \begin{cases} 1, \; if\, p,q \in \bar{\Omega}_{\alpha} \\ 0, otherwise \end{cases}, \alpha = 1,\ldots,E;\; and \tag{1.18}$$

together with

$$m(p,q) \equiv \sum_{\alpha=1}^{E} \delta_{pq}^{\alpha} \tag{1.19}$$

The function $m(p,q)$ is the *'multiplicity of the pair* (p,q)*'*, which can be zero, when the pair p and q do not occur simultaneously in any subdomain-closure. The general procedure for transforming a standard (overlapping) discretization into a non-overlapping one can be applied whenever the *original-matrix* $\underline{\underline{M}} \equiv \left(M_{pq}\right)$ fulfills the following condition:

$$m(p,q) = 0 \Rightarrow M_{pq} = 0 \tag{1.20}$$

The *non-overlapping discretization with constraints* has the form

$$\underline{\underline{a}}\underline{\underline{A}}\underline{u} = \underline{f} \; and \; \underline{\underline{j}}\underline{u} = 0 \tag{1.21}$$

where $\underline{f}$ and $\underline{u}$ are *derived-vectors*. As for $\underline{\underline{a}}$ and $\underline{\underline{j}}$ they are the *projection-matrices* on the subspaces of *continuous-vectors* and *zero-average-vectors*, respectively. The matrix $\underline{\underline{A}}$ is defined by:

$$\underline{\underline{A}} \equiv \underline{\underline{a}}'\underline{\underline{A}}^{t} \tag{1.22}$$

Above, $\underline{\underline{a}}'$ is the projection on the subspace $W' \subset W$, which the DVS-method introduces to accelerate convergence and can be defined by:

$$W' \equiv W(\mathrm{I}) \oplus W(\Delta) \oplus \underline{\underline{a}}W(\pi) \tag{1.23}$$

The matrix $\underline{\underline{A}}^t$, the *matrix A total* is a block-diagonal matrix that is derived from the original matrix $\underline{\underline{M}}$; each one of its blocks $\underline{\underline{A}}^\alpha$, $\alpha = 1, \ldots, E$, $\underline{\underline{A}}^\alpha : W^\alpha \to W^\alpha$, transforms W^α into itself. Such block-matrices $\underline{\underline{A}}^\alpha$ are similar to restrictions of the original matrix $\underline{\underline{M}}$, but they are already defined in the subspaces W^α of the *derived-vector space*. The full expression of $\underline{\underline{A}}^t$ is:

$$\underline{\underline{A}}^t \equiv \sum_{\alpha=1}^{E} \underline{\underline{A}}^\alpha \tag{1.24}$$

Its detailed expression, in terms of the original matrix $\underline{\underline{\mathrm{M}}}$ of Eq. (1.2), is given in [11].

8 The Preconditioned DVS-Algorithms with Constrains

Direct application of Eq. (1.21) is not sufficiently efficient, in spite that a constrained-space formulation has already been incorporated in it. Thus, to enhance parallelization efficiency, in this Section we review the incorporation of the Schur-complement, as well as preconditioning. Finally, the four *DVS-algorithms* obtained in this manner will be listed and briefly explained.

In general, the right-hand member of Eq. (1.21) is written as: $\underline{f} = \underline{f}_\Delta + \underline{f}_\Pi$. When $\underline{f}_\Pi = 0$, the *DVS-algorithms* are easier to write and, furthermore, the transformation of cases when $\underline{f}_\Pi \neq 0$ into others in which $\underline{f}_\Pi = 0$ is straightforward. Thus, in what follows the formulas will be written under the assumption that $\underline{f}_\Pi = 0$. Then, the basic *Schur-complement formulation* is:

$$\underline{\underline{aS}}\underline{u}_\Delta = \underline{f}_\Delta \textit{ and } \underline{\underline{j}}\underline{u}_\Delta = 0 \tag{1.25}$$

complemented by

$$\underline{u}_\Pi = -\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{\sim 1} \underline{\underline{A}}_{\Pi\Delta}\underline{u}_\Delta \tag{1.26}$$

Here, in general, in the DVS approach, the *Schur-complement* is defined by:

$$\underline{\underline{S}} \equiv \underline{\underline{A}}_{\Delta\Delta} - \underline{\underline{A}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{\sim 1} \underline{\underline{A}}_{\Pi\Delta} \tag{1.27}$$

We recall that this *Schur-complement* definition already incorporates constraints.

Four *preconditioned DVS-algorithms with constraints* have been obtained [11–13]; two of them can be derived by applying BDDC and FETI-DP using for that purpose a *non-overlapping discretization*, while other two are derived following a fully independent path [11]. One feature that characterizes each one of these algorithms is the *sought-information*; in this context, the *sought-information* is a piece of information - the most computationally-expensive to obtain- such that once it has been obtained the *complementary information* (i.e., remaining information required to get the full solution of the problem)- is easy and non-computationally-costly to gather (see [11] for additional details).

8.1 The DVS-BDDC Algorithm

This algorithm follows by application of $\underline{\underline{aS}}^{-1}$ as a *preconditioner* to the *Schur-complement formulation* of Eq. (1.25). In this algorithm the *sought information* is $\underline{u}_\Delta$, and the algorithm that we get for it is:

$$\underline{\underline{aS}}^{-1}\underline{\underline{aS}}\underline{u}_\Delta = \underline{\underline{aS}}^{-1}\underline{f}_\Delta \textit{ and } \underline{\underline{j}}\underline{u}_\Delta = 0 \tag{1.28}$$

while the *complementary information* fulfills Eq. (1.26), which for completeness here we repeat:

$$\underline{u}_\Pi = -\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{\sim 1}\underline{\underline{A}}_{\Pi\Delta}\underline{u}_\Delta \tag{1.29}$$

8.2 The DVS-PRIMAL Algorithm

We set $\underline{v}_\Delta \equiv -\underline{\underline{S}}^{\sim 1}\underline{\underline{j}}\underline{\underline{S}}\underline{u}_\Delta$ and the algorithm consists in searching for a function $\underline{v}_\Delta \in W_\Delta$, which fulfills

$$\underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{v}_\Delta = \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \textit{ and } \underline{\underline{aS}}\underline{v}_\Delta = 0 \tag{1.30}$$

Once $\underline{v}_\Delta \in W(\Delta)$ has been obtained, then

$$\underline{u}_\Delta = \underline{\underline{a}}\left(\underline{\underline{S}}^{-1}\underline{f}_\Delta - \underline{v}_\Delta\right) \tag{1.31}$$

8.3 The DVS-FETI-DP Algorithm

In this case the *sought information* is $\underline{\lambda} \equiv -\underline{\underline{j}}\underline{\underline{S}}\underline{u}_\Delta$, which is denoted by $\underline{\lambda}$. Thus, the algorithm is: “Given $\underline{f}_\Delta \in \underline{\underline{a}}W_\Delta$, find $\underline{\lambda} \in W_\Delta$ such that

$$\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{\sim 1}\underline{\lambda} = \underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{\sim 1}\underline{f}_\Delta \textit{ and } \underline{\underline{a}}\underline{\lambda} = 0 \tag{1.32”}$$

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

$$\underline{u}_\Delta = \underline{\underline{aS}}^{\sim 1}\left(\underline{f}_\Delta + \underline{\lambda}\right) \tag{1.33}$$

8.4 The DVS-Dual Algorithm

The *sought information* is $\underline{\underline{S}}\underline{u}_\Delta$, which is denoted by $\underline{\mu}$. Then, we seek for $\underline{\mu} \in W(\Delta)$ such that

$$\underline{\underline{SaS}}^{-1}\underline{\underline{a}}\underline{\mu} = \underline{\underline{SaS}}^{-1}\underline{f}_\Delta \text{ and } \underline{\underline{jS}}^{-1}\underline{\mu} = 0 \tag{1.34}$$

Once $\underline{\mu} \in W(\Delta)$ has been obtained, $\underline{u}_\Delta \in W(\Delta)$ is given by:

$$\underline{u}_\Delta = \underline{\underline{aS}}^{-1}\underline{\mu} \tag{1.35}$$

Remark 1. In these algorithms the matrices to be iterated are:$\underline{\underline{aS}}^{-1}\underline{\underline{aS}}$, $\underline{\underline{S}}^{-1}\underline{\underline{jSj}}$, $\underline{\underline{jSjS}}^{-1}$ and $\underline{\underline{SaS}}^{-1}\underline{\underline{a}}$, respectively, and it should be observed that the action of each one of them on any *derived-vector* yields a vector that fulfills the restriction corresponding to each one of the Eqs. (1.28), (1.30), (1.32)and (1.34), respectively. This property is necessary for any iterative algorithm.

Remark 2. The application of the projection-operator $\underline{\underline{a}}$ at the end, in Eqs. (1.31), (1.33) and (1.35), would be unnecessary if the algorithms of Eqs. (1.30), (1.32) and (1.34), for the the vectors $\underline{v}_\Delta$, $\underline{\lambda}$ and $\underline{\mu}$ would yield exact results; however, their results are only approximate and therefore the application of $\underline{\underline{a}}$, which furthermore is very cheap, significantly improves the precision.

9 Elementary Pieces of DVS-Software and Critical Coding Routes

The DVS-algorithms are domain-decomposition algorithms. As most of this kind they are iterative algorithms 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, depending on the *DVS-algorithm* that is applied, one has to compute the action on a *derived-vector* of one of the following matrices: $\underline{\underline{aS}}^{\sim 1}\underline{\underline{aS}}$, $\underline{\underline{jSjS}}^{\sim 1}$, $\underline{\underline{S}}^{\sim 1}\underline{\underline{jSj}}$ or $\underline{\underline{SaS}}^{\sim 1}\underline{\underline{a}}$. In turn, such matrices are different permutations of $\underline{\underline{S}}$, $\underline{\underline{S}}^{\sim 1}$, $\underline{\underline{a}}$ and $\underline{\underline{j}}$. Thus, a code for implementing any of the DVS-algorithms can be easily developed when codes for carrying out the action of each one of such matrices are available.

In the absence of constraints the matrices $\underline{\underline{S}}$ and $\underline{\underline{S}}^{\sim 1}$ are block-diagonal and, for its implantation in parallel, each block can be allocated to a different processor. However, when constraints are introduced they are weakly coupled. Such a coupling occurs only when the matrix $\underline{\underline{a}}$ is applied; the application $\underline{\underline{j}}$ is equivalent to an application of $\underline{\underline{a}}$, since $\underline{\underline{j}}\underline{u} \equiv \underline{u} - \underline{\underline{a}}\underline{u}$. Because of these facts it has been possible to develop optimized routes of general applicability for its parallel implementation that are coded independently in each one of the processors of the parallel hardware. These properties have permitted us to obtain the outstanding parallelization-efficiencies that are reported in the Section on numerical results (see Table 1).

10 Application to Elasticity Problems

As an illustration of the methods described in previous Sections, in this one the *non-overlapping discretization method* will be applied to a system of linear partial differential equations; namely, the system that governs the equilibrium of isotropic elastic solids, whose detailed treatment was explained in the Ph.D. thesis of Iván Contreras [17].

In particular, the following boundary-value problem will be treated:

$$(\lambda+\mu)\nabla\nabla \bullet \underline{u} + \mu\Delta\underline{u} = \underline{f}_{\Omega} \tag{1.36}$$

Subjected to the *Dirichlet* boundary conditions:

$$\underline{u} = 0, \text{ on } \partial\Omega \tag{1.37}$$

Other boundary conditions can also be accommodated.

The software that we have developed treats in parallel the discrete system of linear equations that is obtained when the *standard discretization method* used to obtain the *original discretization* of the Dirichlet BVP defined by Eqs. (1.36) and (1.37) is the finite element method (FEM). In particular, it was obtained applying the well-known variational principle:

$$\int_{\Omega}\{(\lambda+\mu)(\nabla\bullet\underline{u})(\nabla\bullet\underline{w}) + \mu\nabla\underline{u} : \nabla\underline{w}\}dx = \int_{\Omega}\underline{f}_{\Omega}\bullet\underline{w}dx \tag{1.38}$$

In particular, linear functions were used.

Such system of equations can be written as

$$\underline{\underline{M}}\underline{U} = \underline{F} \tag{1.39}$$

Here, it is understood that the vectors $\underline{U}$ and $\underline{F}$, are functions defined on the whole set of *original-nodes* of the mesh used in the FEM discretization, whose values at each node are $3-D$ vectors. They can be written as $\underline{U} \equiv \left(\underline{U}_p\right) \equiv \left(U_{pi}\right)$ and $\underline{F} \equiv \left(\underline{F}_p\right) \equiv \left(F_{pi}\right)$. As for the matrix $\underline{\underline{M}}$, the notation

$$\underline{\underline{M}} \equiv \left(\underline{\underline{M}}_{pq} \right) \equiv \left(M_{piqj} \right) \tag{1.40}$$

is adopted. Above, the range of *p and q* is the whole set of *original-nodes*, while i and j may take any of the values $1, 2, 3$.

11 Numerical Results

In the numerical experiments that were carried out to test the *DVS-software*, the boundary-value problem for static elasticity introduced in Sect. 10 was treated. Only the DVS-BDDC algorithm was tested. The elastic material was assumed to be homogeneous; so, the Lamé parameters were

Table 1. Numerical Results

Number of Subdomains. = Number of processors	DoF.	Nodes by Subdomain	Primal Nodes	Processing Time in seconds	Parallel efficiency $\left(\frac{p_{\min}}{p_{\max}} \bullet s\right) \times 100$	Speed up $s = \frac{T(p_{\min})}{T(p_{\max})}$	Norm of error $\Vert \underline{e} \Vert_\infty$
8	22,244,625	941,192	583	14,959	1	1	0.0263
27	21,904,152	274,625	2,312	5,882	75 %	2.543	0.018
64	22,244,625	117,649	5,211	2,676	70 %	5.59	0.029
125	21,904,152	59,319	9,184	1,212	79 %	12.342	0.011
216	22,936,119	35,937	14,525	703	79 %	21.280	0.010
343	22,244,625	21,952	20,628	406	86 %	36.845	0.010
512	23,641,797	13,824	27,391	242	97 %	61.814	0.011
729	23,287,176	10,648	36,800	183	90 %	81.74	0.010
1000	23,641,797	8,000	46,899	136	88 %	109.992	0.009
1331	22,936,119	5,832	57,100	96	94 %	155.823	0.010
1728	20,903,613	4,096	66,671	89	78 %	168.078	0.009
2197	21,904,152	3,375	80,352	64	85 %	233.734	0.008
2744	22,244,625	2,744	94,471	51	86 %	293.313	0.009

assumed to be constant and their values were taken to be

$$\begin{aligned} \lambda &= \frac{Ev}{(1+v)(1-2v)} = 29.6412 \times 10^9 \frac{\mathrm{N}}{m^2} \\ \mu &= \frac{E}{2(1+v)} = 27.3611 \times 10^9 \frac{\mathrm{N}}{m^2} \end{aligned} \tag{1.41}$$

These values correspond to a class of cast iron whose *Young modulus*, E, and *Poison ratio*, v, are (for further details about such a material see, http://en.wikipedia.org/wiki/Poisson's_ratio):

$$E = 68.95 \times 10^9 \frac{N}{m^2} \; and \; \nu = 0.26 \tag{1.42}$$

The domain $\Omega \subset R^3$ that the homogeneous-isotropic linearly-elastic solid considered occupies is a unitary cube. The boundary-value problem considered is a *Dirichlet problem*, with homogeneous boundary conditions, whose exact solution is:

$$\underline{u} = (\sin\, \pi x \sin \pi y \sin \pi z,\; \sin\, \pi x \sin \pi\, y\, \sin \pi z,\; \sin\, \pi x \sin \pi y \sin \pi z) \tag{1.43}$$

The *fine-mesh* that was introduced consisted of $(193)^3 = 7,189,057$ cubes, which yielded $(194)^3 = 7,301,384$ *original-nodes*.

The *coarse-mesh* consisted of a family of subdomains $\{\Omega_1, \ldots, \Omega_E\}$, whose interfaces constitute the *internal-boundary* Γ. The number E of subdomains was varied taking successively the values 8, 27, 64, 125, 216, 343 and 512 and so on up to 2, 744 . The total number of *derived-nodes* and corresponding number of *degrees-of-freedom* are around 7.5×10^6 and 22.5×10^6, respectively. The constraints that were imposed consisted of continuity at *primal-nodes*; in every one of the numerical experiments all the nodes located at edges and vertices of the *coarse mesh* were taken as *primal-nodes*. In this manner, the total number of *primal-nodes* varied from a minimum of 583 to a maximum of 94, 471. Thereby, it should be mentioned that these conditions granted that at each one of the numerical experiments the matrix $\underline{\underline{A}}$ was positive definite and possessed a well-defined inverse.

All the codes were developed in C ++ and MPI was used. The computations were performed at the Mitzli Supercomputer of the National Autonomous University of Mexico (UNAM), operated by the DGTIC. All calculations were carried out in a 314-node cluster with 8 processors per node. The cluster consists 2.6 GHz Intel Xeon Sandy Bridge E5-2670 processors with 48 GB of RAM.

As it was exhibited in the analysis of the operations, the transmission of information between different processors exclusively occurs when the *average-operators* $\underline{\underline{a}}$ and $\underline{\underline{a}}'$ are applied. In a first version of the software reported in the present paper such exchange of information was carried out through a *master-processor*, which is time expensive. However, the efficiency of the software (as a parallelization tool) improved very much when the participation of the *master-processor* in the communication and exchange of information process was avoided. In its new version, the *master-processor* was eliminated altogether. A Table 1, above, summarizes the numerical results.

It should be noticed that the *computational efficiency* is very high, reaching a maximum value of 96.6 %. Furthermore, the efficiency increases as the number of processors increases, a commendable feature for software that intends to be top as a tool for programming the largest supercomputers available at present.

12 Conclusions

The *non-overlapping discretization method* originally introduced by I. Herrera is new procedure for discretizing partial differential equations, or systems of such equations. For the application of highly parallelized hardware to the resulting discrete-problem the *non-overlapping discretization method* has many advantages over standard methods of discretization.

Based on theoretical grounds, in previous publications some of these advantages had been indicated. However, for the first time experimental evidences of such outstanding parallelization-efficiencies are exhibited in this paper. There are *DVS-algorithms* concomitant to the *non-overlapping discretization method*. Here, one of them – the *DVS-BDDC* algorithm- was used to carry out such numerical experiments.

These results constitute a confirmation that the *non-overlapping discretization method* and its concomitant *DVS-algorithms* are very effective tools for harnessing parallelized supercomputers to the task of solving the partial differential equations of science and engineering:

1. Using them very high parallelization effectiveness, close to 100 %, is feasible;
2. The range of its applicability is very wide since there is a general procedure for transforming *standard discretizations* into *non-overlapping discretizations* independently of the problems that originated them;
3. Up to now only the case of symmetric and positive-definite matrices has been fully developed;
4. However, the *non-overlapping discretization method* is also applicable to symmetric and non-symmetric matrices and the results so far obtained indicate that its development is a worthwhile endeavor ahead.

This paper has been mainly devoted to present experimental evidences of the effectiveness of the methodology, but extensive references to the background material have been included.

Acknowlegement. We thank DGTIC-UNAM for the significant support we received to perform the computational experiments presented in Table 1.

Appendix

Here some of the notation used is recalled. The non-overlapping domain-decomposition is $\{\Omega_1, \ldots, \Omega_E\}$. The symbol X is used for the total set of *derived-nodes*. The labels p, q, etc. are reserved for denoting the *original-nodes* whose range is the set $\{1, \ldots, N\}$ of natural-numbers, while the labels α, β, etc. are reserved for the subdomains of the *coarse-mesh*, whose range is the set $\{1, \ldots, E\}$. *Derived-nodes* are identified by pairs: (p, α), p being the *original-node* it derives from and α the subdomain it belongs to. The non-overlapping decomposition of the total set of *derived-nodes* is given by the family of subsets $\mathrm{X}^\alpha \equiv \{\mathrm{X}^\alpha | \alpha = 1, \ldots, E\}$, which for each $\alpha = 1, \ldots, E$, is constituted by the *derived-nodes* that belong to $\bar{\Omega}_\alpha$; i.e., they are defined by

$$X^{\alpha} \equiv \left\{ (p, \alpha) | p \in \bar{\Omega}_{\alpha} \right\} \tag{1.44}$$

and satisfy:

$$X = \bigcup_{\alpha=1}^{E} X^{\alpha} \; and \; X^{\alpha} \cap X^{\beta} = \emptyset, when \, \alpha \neq \beta \tag{1.45}$$

Given any *original-node* $p \in \overset{\frown}{X}$, the set of *heirs of* p is defined to be

$$Z(p) \equiv \{(p, \alpha) | (p, \alpha) \in X\} \tag{1.46}$$

The multiplicity $m(p)$ of $p \in \overset{\frown}{X}$, is the number of *heirs of* p.

In DVS developments *derived-nodes* are classified into *interior*, *interphase*, *primal* and *dual nodes*. A *derived-node* is: *interior*, when its multiplicity is one and it is *interphase*, otherwise. In the *DVS-methodology* some *interphase-nodes* are chosen to be *primal* and, when they are not *primal*, they are said to be *dual-nodes*. The symbols used are:

(i) I is the set of *interior-nodes*;
(ii) Γ is the set of *interphase-nodes*;
(iii) π is the set of *primal-nodes*; and
(iv) Δ is the set of *dual-nodes*.

These subsets of *derived-nodes* fulfill the following identities:

$$X = I \cup \Gamma = I \cup \pi \cup \Delta = \Pi \cup \Delta = \Sigma \cup \pi \tag{1.47}$$

and

$$\emptyset = I \cap \Gamma = I \cap \pi = \pi \cap \Delta = \Pi \cap \Delta = \Sigma \cap \pi \tag{1.48}$$

Two more subsets of *derived-nodes* are significant in *non-overlapping discretization methods*: the classes of *extended-primal* and *extended-dual derived-nodes* that are denoted by Π and Σ respectively, and are defined by

$$\Pi \equiv I \cup \pi \, and \, \Sigma \equiv I \cup \Delta \tag{1.49}$$

References

1. Herrera, I., Pinder, G.F.: Mathematical Modelling in Science and Engineering: An axiomatic approach. Wiley, Hoboken (2012)
2. President's Information Technology Advisoty Committee: Pitac. Computational Science: Ensuring America´s Competitiveness. Report to the President June 2005. 104 p. www.nitrd.gow/pitac

3. DDM Organization, Proceedings of 23 International Conferences on Domain Decomposition Methods 1988–2015. www.ddm.org
4. Dohrmann, C.R.: A preconditioner for substructuring based on constrained energy minimization. SIAM J. Sci. Comput. **25**(1), 246–258 (2003)
5. Mandel, J., Dohrmann, C.R.: Convergence of a balancing domain decomposition by constraints and energy minimization. Numer. Linear Algebra Appl. **10**(7), 639–659 (2003)
6. Mandel, J., Dohrmann, C.R., Tezaur, R.: An algebraic theory for primal and dual substructuring methods by constraints. Appl. Numer. Math. **54**, 167–193 (2005)
7. Farhat, Ch., Roux, F.: A method of finite element tearing and interconnecting and its parallel solution algorithm. Internat. J. Numer. Methods Engrg. **32**, 1205–1227 (1991)
8. Mandel, J., Tezaur, R.: Convergence of a substructuring method with Lagrange multipliers. Numer. Math. **73**(4), 473–487 (1996)
9. 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 Engrg. **50**, 1523–1544 (2001)
10. Farhat, C., Lessoinne, M., Pierson, K.: A scalable dual-primal domain decomposition method. Numer. Linear Algebra Appl. **7**, 687–714 (2000)
11. Herrera, I., de la Cruz, L.M., Rosas-Medina, A.: Non-Overlapping Discretization Methods for Partial, Differential Equations. Numer. Meth. Part D E **30**, 1427–1454 (2014). doi:10.1002/num21852. (Open source)
12. Herrera, I., Rosas-Medina, A.: The derived-vector space framework and four general purposes massively parallel DDM algorithms. EABE (Engineering Analysis with Boundary Elements) **37**, 646–657 (2013)
13. Herrera, I., Contreras, I.: An innovative tool for effectively applying highly parallelized hardware to problems of elasticity. Geofísica Int. **55**(1), 363–386 (2016)
14. Herrera, I.: Theory of differential equations in discontinuous piecewise-defined-functions. Numer. Meth. Part D E **23**(3), 597–639 (2007). DOI10.1002NO.20182
15. Herrera, I., Yates, R.A.: The multipliers-free domain decomposition methods. Numer. Meth. Part D. E. **26**, 874–905 (2010). doi:10.1002/num.20462
16. Herrera, I., Yates, R.A.: The multipliers-free dual primal domain decomposition methods for nonsymmetrical matrices numer. Meth. Part D. E. **27**(5), 1262–1289 (2011). doi:10.1002/Num.20581
17. Contreras, I.: Parallel Processing of PDEs. Ph.D. thesis, Advisor Herrera I. UNAM (2016)