Blucher Mechanical Engineering Proceedings
May 2014, vol. 1 , num. 1
www.proceedings.blucher.com.br/evento/10wccm

10th World Congress on Computational Mechanics
8-13 July 2012 • São Paulo • Brazil

# THE DERIVED-VECTOR-SPACE: A UNIFIED FRAMEWORK FOR NON-OVERLAPPING DDM

I. Herrera[1]

[1]Instituto de Geofisica, National University of Mexico, UNAM, (iherrerarevilla@gmailcom)

**Abstract.** *We present and discuss a framework called the 'derived-vector-space (DVS) framework'. In part, the relevance of this framework is because in its realm four preconditioned massively parallelizable DVS-algorithms with constraints of general applicability (by this we mean: applicable to symmetric-definite, non-symmetric and indefinite matrices) have been developed. Such algorithms yield codes that are almost 100% in parallel; more precisely, they achieve what we call the 'leitmotif of DDM research': to obtain algorithms that yield the global solution by solving local problems exclusively. This has been possible because in the DVS-approach the original PDE, or system of such equations, is transformed into a problem formulated in the derived-vector-space, which is a Hilbert-space that is well-defined by itself independently of the particular PDE considered, and afterwards all the numerical work is done in that space. Thus, the applicability of the algorithms developed in this framework is essentially independent of the specific problem considered, and furthermore this allows the development of codes, which to a large extent are of universal applicability. Thus, the DVS-algorithms are very suitable for programming in an efficient manner the most powerful parallel computers available at present.*

**Keywords:** *Massively-parallel-algorithms; parallel-computers; non-overlapping-DDM; BDDC; FETI-DP.*

## 1. INTRODUCTION

In this paper, we present and discuss a framework recently introduced called the 'derived-vector-space (DVS) framework' [1-7] (refer to [1-4] for the most recent developments and to [5-7] for background material). The main conspicuous features of the *DVS-framework* are:

1. It is an axiomatic approach;

2. Its starting point is the system of equations that is obtained after the partial differential (PDE), or system of such equations, has been discretized;

3. The *derived-vector-space (DVS)* has a Hilbert-space structure with respect to an inner-product that is defined independently of the problem considered. This permits to treat in the very same setting symmetric-definite, non-symmetric or indefinite matrices. Furthermore, the space of continuous vectors is a subspace of the DVS;

4. The problem is transformed into a problem defined in the *DVS* and, once this has been done, all the numerical work is carried out in the *DVS*.

The *DVS-framework* is innovative in many respects and has yielded a good number of significant new results. One especially important is a set four preconditioned massively parallelizable DVS-algorithms with constraints of general applicability (by this we mean: applicable to symmetric-definite, non-symmetric and indefinite matrices). Such algorithms produce codes that are almost 100% in parallel; more precisely, they achieve what we call in what follows the *'leitmotif of DDM research'*: to obtain algorithms that yield the <u>global</u> solution by solving <u>local</u> problems exclusively. Furthermore, when the processing of different subdomains is sent to different processors the communication required among them is insignificant [1]. The *DVS-algorithms*, in spite their generality, are independent of the problem considered. This fact, together with the outstanding uniformity of the algorithmic formulas that are obtained, yields clear advantages for code development, especially when such codes are built using object-oriented programming techniques. This allows the development of codes that to a large extent are of universal applicability.

The *DVS-algorithms* here presented are based on the extensive work on DDM done during the last twenty years or so by many authors (see, for example [8,9]), albeit what is special of the *DVS-approach* is the use of the *DVS-framework* as the setting for its developments. Of the set of four preconditioned *DVS-algorithms* with constraints mentioned above apparently two of them are fully novel, since to our knowledge nothing similar has been reported previously in the literature [1-2]. As for the other two (to be referred as DVS-BDDC and DVS-FETI-DP), they are the *DVS-versions* of two well-known, very efficient non-overlapping DDM approaches: the balancing domain decomposition with constraints (BDDC) and the dual-primal finite-element tearing and interconnecting (FETI-DP) methods, respectively. However, it should be noticed that the BDDC and FETI-DP methods were originally developed for definite-symmetric problems, while the DVS-BDDC and DVS-FETI-DP algorithms are applicable to very general non-symmetric and indefinite matrices, independently of the specific PDE that originated them. It should also be noticed that standard versions of BDDC and FETI-DP do not fully achieve the *leitmotif of DDM*; i.e., they do not obtain the <u>global</u> solution by solving <u>local</u> problems exclusively.

We would like to recall that, as it is well-known, the balancing domain decomposition method was originally introduced by Mandel [10,11] and more recently modified by Dohrmann who introduced constraints in its formulation [12-14], while the original finite-element tearing and interconnecting method was introduced by Farhat [15,16] and later modified by the introduction of a dual-primal approach [17-19]. To be fair, we also would like to notice that certain number of applications of these methods have been made to non-symmetric and indefinite matrices (see, for example, [20-23]).

## 2. OVERVIEW OF THE DVS-FRAMEWORK

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; this system of discrete equations is referred to as the *'original problem'*. Independently of the discretization method used, it is assumed that a set of nodes and a domain-partition have been introduced and that both the nodes and the partition-subdomains have been numbered. Generally, some of these *original-nodes* belong to more than one partition-subdomain (Fig.1). For the formulation of non-overlapping domain decomposition methods it would be better if each node would belong to one and only one partition-subdomain, and 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* in as many pieces as required to obtain a set with the desired property (Fig.2). Then, each *derived-node* is 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 belongs to. A *'derived-vector'* is simply defined to be any real-valued function[*] defined in the whole set of *derived-nodes*; the set of all *derived-vectors* constitutes a linear space: the *'derived-vector space (DVS)'*. The *'Euclidean inner-product'*, for each pair of *derived-vectors*, is defined to be the product of their components summed over all the *derived-nodes*. The DVS constitutes a finite-dimensional Hilbert-space, with respect to the *Euclidean inner-product*. A new problem, which is equivalent to the *original problem*, is defined in the *derived-vector space*. 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 [2]. 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 the following preconditioned *DVS-algorithms*: *DVS-primal* formulation #1, *DVS-dual* formulation #1, *DVS-primal* formulation #2 and *DVS-dual* formulation #2. However, the *DVS-primal* formulation #1 and *DVS-dual* formulation #1 were later identified as *DVS-versions* of *BDDC* and *FETI-DP*, respectively. With this adjustment in nomenclature, in [1,2] they were summarized as it is indicated next. Thereby, it should be mentioned that standard versions of *BDDC* and *FETI-DP* (see references mentioned above) were originally formulated for definite-symmetric matrices (positive-definite or negative-definite); thus, their extension to non-symmetric and indefinite matrices is among the achievements of the *DVS-framework.*

**The Four General Preconditioned Algorithms with Constraints** that will be discussed in this paper are:

a).- The *DVS-version of BDDC*;

b).-The *DVS-version of FETI-DP*;

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

c).- *DVS-primal* formulation #2;

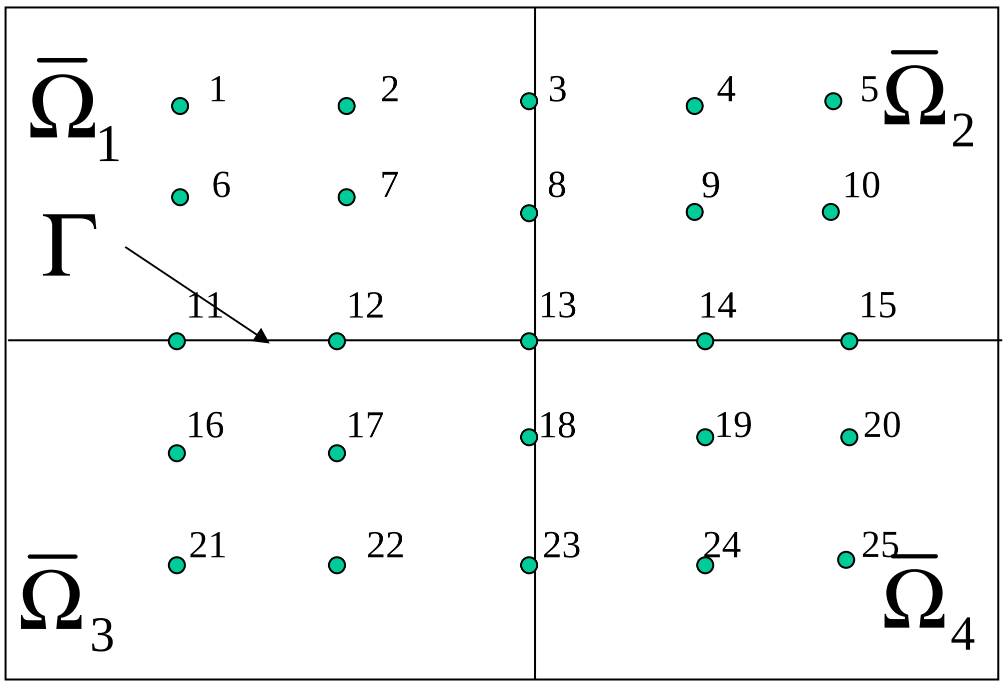

Figure 1. The *'original nodes'*

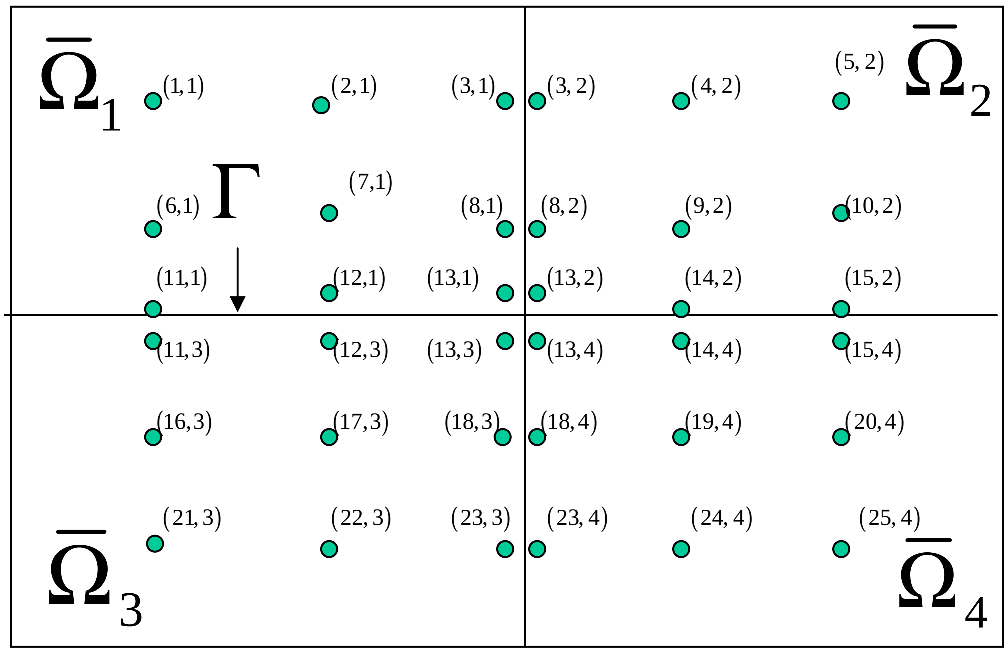

Figure 2. The *'derived nodes'*

d).- *DVS-dual* formulation #2.

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

## 3. THE DVS-ALGORITHMS WITH CONSTRAINTS

In this Section we summarize some results previously obtained. The interested reader is referred, in particular, to [1-4] for details. In the *DVS-framework*, the *original problem* is transformed into one defined in the *derived-vector space with restrictions*, $W_r$. Thus, we look for a vector $\underline{u} \in W_r$, such that

$$\underline{\underline{a}}\,\underline{\underline{A}}\,\underline{u} = \underline{f} \textit{ and } \underline{\underline{j}}\,\underline{u} = 0 \tag{3.1}$$

Where $\underline{u} \equiv \underline{u}_\Pi + \underline{u}_\Delta$ and $\underline{f} \equiv \underline{f}_\Pi + \underline{f}_\Delta$, while $\underline{u}_\Delta, \underline{f}_\Delta \in W_\Delta$ are the *dual* parts of the respective vectors. For simplicity of exposition, here we assume that $\underline{f}$ takes only *dual-values*; $\underline{f} = \underline{f}_\Delta$ (i.e., $\underline{f}_\Pi = 0$). Then, Eq.(3.1) is equivalent to

$$\underline{\underline{a}}\,\underline{\underline{S}}\,\underline{u}_\Delta = \underline{f}_\Delta \textit{ and } \underline{\underline{j}}\,\underline{u}_\Delta = 0 \tag{3.2}$$

Together with

$$\underline{u}_\Pi = -\underline{\underline{A}}_{\Pi\Pi}^{-1}\underline{\underline{A}}_{\Pi\Delta}\underline{u}_\Delta \tag{3.3}$$

Above, the matrix $\underline{\underline{a}}$ is the projection on the subspace of *continuous derived-vectors*, while $\underline{\underline{j}}$ is the projection on the subspace of *zero-average derived-vectors*, which is the orthogonal complement of the subspace of *continuous* vectors. Furthermore,

$$\underline{\underline{A}} = \begin{pmatrix} \underline{\underline{A}}_{\Pi\Pi} & \underline{\underline{A}}_{\Pi\Delta} \\ \underline{\underline{A}}_{\Delta\Pi} & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix} \textit{ and } \underline{\underline{S}} \equiv \underline{\underline{A}}_{\Delta\Delta} - \underline{\underline{A}}_{\Delta\Pi}\underline{\underline{A}}_{\Pi\Pi}^{-1}\underline{\underline{A}}_{\Pi\Delta} \tag{4}$$

### 3.1- THE DVS VERSION OF THE BDDC ALGORITHM

The *DVS-BDDC algorithm* is [2]: "Find $\underline{u}_\Delta \in W_\Delta$ such that

$$\underline{\underline{a}}\,\underline{\underline{S}}^{-1}\underline{\underline{a}}\,\underline{\underline{S}}\,\underline{u}_\Delta = \underline{\underline{a}}\,\underline{\underline{S}}^{-1}\underline{f}_\Delta \textit{ and } \underline{\underline{j}}\,\underline{u}_\Delta = 0 \quad (3.5)\text{"}$$

### 3.2- THE DVS VERSION OF FETI-DP ALGORITHM

The *DVS-FETI-DP algorithm* is [2]: "Given $\underline{f}_\Delta \in \underline{\underline{a}}W_\Delta$, find $\underline{\lambda} \in W_\Delta$ such that

$$\underline{\underline{j}}\,\underline{\underline{S}}\,\underline{\underline{j}}\,\underline{\underline{S}}^{-1}\underline{\lambda} = \underline{\underline{j}}\,\underline{\underline{S}}\,\underline{\underline{j}}\,\underline{\underline{S}}^{-1}\underline{f}_\Delta \textit{ and } \underline{\underline{a}}\,\underline{\lambda} = 0 \quad (3.6)\text{"}$$

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

$$\underline{u}_\Delta = \underline{\underline{a}}\,\underline{\underline{S}}^{-1}\left(\underline{f}_\Delta - \underline{\lambda}\right) \tag{3.7}$$

### 3.3- THE DVS PRIMAL-ALGORITHM #2

This algorithm consists in searching for a function $\underline{v}_\Delta \in W_\Delta$, which fulfills [2]

$$\underline{\underline{S}}^{-1}\underline{\underline{j}}\,\underline{\underline{S}}\,\underline{\underline{j}}\,\underline{v}_\Delta = \underline{\underline{S}}^{-1}\underline{\underline{j}}\,\underline{\underline{S}}\,\underline{\underline{j}}\,\underline{\underline{S}}^{-1}\underline{f}_\Delta \textit{ and } \underline{\underline{a}}\,\underline{\underline{S}}\,\underline{v}_\Delta = 0 \quad (3.8)\text{"}$$

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

$$\underline{u}_\Delta = \underline{\underline{a}}\left(\underline{\underline{S}}^{-1}\underline{f}_\Delta - \underline{v}_\Delta\right) \tag{3.9}$$

### 3.4- THE DVS DUAL-ALGORITHM #2

This algorithm consists in searching for a function $\underline{\mu} \in W_\Delta$, which fulfills [2]

$$\underline{\underline{S}}\,\underline{\underline{a}}\,\underline{\underline{S}}^{-1}\underline{\underline{a}}\,\underline{\mu} = \underline{\underline{S}}\,\underline{\underline{a}}\,\underline{\underline{S}}^{-1}\underline{\underline{a}}\,\underline{\underline{S}}\,\underline{\underline{j}}\,\underline{\underline{S}}^{-1}\underline{f}_\Delta \textit{ and } \underline{\underline{j}}\,\underline{\underline{S}}^{-1}\underline{\mu} = 0 \tag{3.10}$$

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

$$\underline{u}_\Delta = \underline{\underline{a}}\,\underline{\underline{S}}^{-1}\left(\underline{f}_\Delta + \underline{\mu}\right) \tag{3.11}$$

## 4. NUMERICAL PROCEDURES

The four preconditioned *DVS-algorithms* with constraints enumerated in Section 3, are

$$\underline{\underline{a}}\,\underline{\underline{S}}^{-1}\underline{\underline{a}}\,\underline{\underline{S}}\,\underline{u}_\Delta = \underline{\underline{a}}\,\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \mathit{and} \quad \underline{\underline{j}}\,\underline{u}_\Delta = 0;\ \mathit{DVS\text{ - }BDDC} \qquad (4.1)$$

$$\underline{\underline{j}}\,\underline{\underline{S}}\,\underline{\underline{j}}\,\underline{\underline{S}}^{-1}\underline{\lambda} = \underline{\underline{j}}\,\underline{\underline{S}}\,\underline{\underline{j}}\,\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \mathit{and} \quad \underline{\underline{a}}\,\underline{\lambda} = 0;\ \mathit{DVS\text{ - }FETI\text{ - }DP} \qquad (4.2)$$

$$\underline{\underline{S}}^{-1}\underline{\underline{j}}\,\underline{\underline{S}}\,\underline{\underline{j}}\,\underline{v}_\Delta = \underline{\underline{S}}^{-1}\underline{\underline{j}}\,\underline{\underline{S}}\,\underline{\underline{j}}\,\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \mathit{and} \quad \underline{\underline{a}}\,\underline{\underline{S}}\,\underline{v}_\Delta = 0;\ \mathit{DVS\text{ - }PRIMAL\text{ - }2} \qquad (4.3)$$

and

$$\underline{\underline{S}}\,\underline{\underline{a}}\,\underline{\underline{S}}^{-1}\underline{\underline{a}}\,\underline{\mu} = \underline{\underline{S}}\,\underline{\underline{a}}\,\underline{\underline{S}}^{-1}\underline{\underline{a}}\,\underline{\underline{S}}\,\underline{\underline{j}}\,\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \mathit{and} \quad \underline{\underline{j}}\,\underline{\underline{S}}^{-1}\underline{\mu} = 0;\ \mathit{DVS\text{ - }DUAL\text{ - }2} \qquad (4.4)$$

In numerical experiments that have been carried out thus far, with symmetric and non-symmetric problems[1,2,4], they have exhibited update numerical efficiency. More precisely, for symmetric problems their efficiency has been slightly better that FETI-DP and BDDC; while for non-symmetric problems it is similar. Of course, in numerical implementations CGM has been used for symmetric-definite matrices and other algorithms such as GM-RES have been used in for non-symmetric ones.

### 4.1- COMMENTS ON THE DVS NUMERICAL PROCEDURES

The outstanding uniformity of the formulas given in Eqs.(4.1) to (4.4) 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.(4.1) to (4.4) 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, in [1] it has been explained how to compute the action of each one of the matrices $\underline{\underline{S}}$ and $\underline{\underline{S}}^{-1}$ by means of computations that are carried out separately in each one of domain decomposition subdomains, exclusively. As for $\underline{\underline{a}}$ and $\underline{\underline{j}}$, their applications require exchange of information between derived-nodes that are *descendants* of the same *original-node*, and that is a very simple operation for which the exchange of information between different subdomains is minimal [1-4].

## 4. CONCLUSIONS

The *derived-vector-space (DVS) framework* unifies domain decomposition methods and has permitted to develop four preconditioned algorithms with constraints that are almost 100% in parallel, in the sense that they achieve the *leitmotif of DDM research*, and which at the same time are to a large extent of universal applicability, in the sense that each one of them is applicable to symmetric-definite, indefinite and non-symmetric matrices, independently of the PDE (or system of such equations) that originated them. Furthermore, each one of them has also update numerical efficiency. Therefore, *DVS-algorithms* are very suitable for programming in an efficient manner the most powerful parallel computers available at present.

## 5. REFERENCES

[1] Herrera, I. & Rosas-Medina A. "Four General Purpose Massively Parallel DDM Algorithms", Available as Memoria #7, GMMC, Instituto de Geofísica, UNAM, 2012.

[2] Herrera, I., Carrillo-Ledesma A. & Rosas-Medina Alberto "A Brief Overview of Non-overlapping Domain Decomposition Methods", Geofisica Internacional, Vol. 50(4), pp 445-463, 2011.

[3] Herrera, I. & Yates R. A. The Multipliers-Free Dual Primal Domain Decomposition Methods for Nonsymmetric Matrices NUMER. METH. PART D. E. 27(5) pp. 1262-1289, 2011. DOI 10.1002/Num. 20581. (Published on line April 28, 2010).

[4] Herrera, I. & Yates R. A. The Multipliers-free Domain Decomposition Methods NUMER. METH. PART D. E. 26: 874-905 July 2010, DOI 10.1002/num. 20462. (Published on line Jan 28, 2009)

[5] Herrera I. and R. Yates "Unified Multipliers-Free Theory of Dual-Primal Domain Decomposition Methods. NUMER. METH. PART D. E. Eq. 25:552-581, May 2009, (Published on line May 13, 08) DOI 10.1002/num. 20359.

[6] Herrera, I. "New Formulation of Iterative Substructuring Methods without Lagrange Multipliers: Neumann-Neumann and FETI", NUMER METH PART D E 24(3) pp 845-878, 2008 (Published on line Sep 17, 2007) DOI 10.1002 NO. 20293.

[7] Herrera, I. "Theory of Differential Equations in Discontinuous Piecewise-Defined-Functions", NUMER METH PART D E, **23**(3), pp 597-639, 2007 DOI 10.1002 NO. 20182.

[8] DDM Organization, Proceedings of 20 International Conferences on Domain Decomposition Methods. www.ddm.org, 1988-2009.

[9] Toselli A. and O. Widlund, Domain decomposition methods- *Algorithms and Theory*, Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2005, 450p.

[10] Mandel J. Balancing domain decomposition, Commun. Numer. Methods Engrg. 1(1993) 233-241.

[11] Mandel J. and Brezina M. Balancing domain decomposition for problems with large jumps in coefficients. Math. Comput. 65, pp 1387-1401, 1996.

[12] Dohrmann C.R., A preconditioner for substructuring based on constrained energy minimization. SIAM J. Sci. Comput. 25(1):246-258, 2003.

[13] Mandel J. and C. R. Dohrmann, Convergence of a balancing domain decomposition by constraints and energy minimization, Numer. Linear Algebra Appl., 10(7):639-659, 2003.

[14] Brenner S. and Sung L. BDDC and FETI-DP without matrices or vectors Comput. Methods Appl. Mech. Engrg. 196(8): 1429-1435. 2007.

[15] Farhat Ch., and Roux F. A method of finite element tearing and interconnecting and its parallel solution algorithm. Internat. J. Numer. Methods Engrg. 32:1205-1227, 1991.

[16] Mandel J. and Tezaur R. Convergence of a substructuring method with Lagrange multipliers. Numer. Math 73(4): 473-487, 1996.

[17] Farhat C., Lessoinne M. and Pierson K. A scalable dual-primal domain decomposition method, Numer. Linear Algebra Appl. 7, pp 687-714, 2000.

[18] Mandel J. and Tezaur R., On the convergence of a dual-primal substructuring method, SIAM J. Sci. Comput., 25, pp 246-258, 2001.

[19] Farhat C., Lessoinne M. LeTallec P., Pierson K. and 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, pp 1523-1544, 2001.

[20] Cai, X-C. & Widlund, O.B., Domain Decomposition Algorithms for Indefinite Elliptic Problems, SIAM J. Sci. Stat. Comput. 1992, Vol. 13 pp. 243-258

[21] Farhat C., and Li J. An iterative domain decomposition method for the solution of a class of indefinite problems in computational structural dynamics. ELSEVIER Science Direct Applied Numerical Math. 54 pp 150-166. 2005.

[22] Li J. and Tu X. Convergence analysis of a Balancing Domain Decomposition method for solving a class of indefinite linear systems. Numer. Linear Algebra Appl. 2009; **16**:745–773

[23] Tu X. and Li J., A Balancing Domain Decomposition method by constraints for advection-diffusion problems. www.ddm.org/DD18/