Special Section
Mathematical and Computational Geophysics

Geofísica Internacional (2026) **65-1**: 2037- 2058

Review of the Domain Decomposition Method in the *Derived-Vector* Space DDM-DVS and its Applicability to Nonsymmetric Matrices

Alicia De la Mora Cebada[1]*, Ernesto Rubio Acosta[2] and Ismael Herrera Revilla[3]

Abstract

This paper examines the Domain Decomposition Method in the Derived Vector Space (DDM-DVS), introduced by Herrera I. and further developed in Herrera *et al.*, (2014), extending its applicability to nonsymmetric matrices. The DDM-DVS or DVS framework, is a domain decomposition method with non-overlapping subdomains and non-overlapping subdomains nodes, differing from standard Iterative Substructuring Methods, in which boundary nodes between subdomains are overlapping. The discretized problem is approached in an extended vector space known as the *Derived Vector Space* (DVS), where the system matrix is structured as a block diagonal matrix, enabling efficient high-performance parallel computing. This structure allows each processor to solve a single subdomain independently. This work contributes to the DVS framework introducing the DVS-Schur and DVS-BDDC methods, employing BiC-Gstab as the global solver. These methods operated in three levels: globally solving for dual nodes, globally solving for primal nodes, and locally solving for nodes inside (internal) the subdomains. It is demonstrated that the Schur complement of the system matrix is also block diagonal. The paper establishes the equivalence of the discretized problem through standard methods and within the DVS framework. The superlinear efficiency of the proposed algorithms is confirmed, with an analysis of their algorithmic complexity in relation to the ratio of dual to interior nodes. Additionally, the paper provides examples based on the general second-order differential operator, which is relevant in Earth Sciences for modeling the steady state of the transport equation. Systems with up to 100 million degrees of freedom were successfully solved using as many as 441 processors.

Key words: DVS-framework, Iterative Substructuring Methods, General Second-Order Differential Equation, DVS-BDDC, BiCGstab, High Performance Parallel Computing.

Resumen

En este trabajo se hace una revisión del Método de Descomposición de Dominio en el Espacio de Vectores Derivados DDM-DVS, propuesto por Herrera I. y desarrollado en (Herrera *et al.*, 2014), extendiendo su aplicabilidad a matrices no simétricas. DDM-DVS o DVS framework es un DDM con subdominios sin traslape y sin nodos traslapados, diferenciándose de los Métodos Iterativos de Subestructuración estándares cuyos nodos de las fronteras entre subdominios están traslapados. Entonces el problema discretizado se resuelve en un espacio vectorial ampliado denominado Espacio Vectorial de Vectores Derivados (DVS). En este espacio, la matriz del sistema de ecuaciones es una matriz diagonal por bloques, lo cual da ventajas significativas para la implementación de algoritmos utilizando cómputo en paralelo de alto desempeño. Lo anterior permite que cada procesador resuelva independientemente un único subdominio.
Las aportaciones de este trabajo al DVS-framework son el desarrollo de los métodos DVS-Schur y DVS-BDDC utilizando como resolvedor global BiCGstab. Estos métodos se estructuran en tres niveles: el primero para resolver globalmente nodos duales, el segundo para resolver globalmente nodos primales y el tercero para resolver localmente nodos del interior de subdominios. Se muestra que el complemento de Schur de la matriz del sistema de ecuaciones también es una matriz diagonal por bloques. Se demuestra la equivalencia del problema discretizado de forma estándar y utilizando DVS-framework. Se corrobora la superlinealidad de estos algoritmos y se explica mediante su complejidad algorítmica, estableciendo regiones de operación según la razón entre el número de nodos duales y de nodos de interior. Se muestran ejemplos con el operador diferencial general de segundo orden. La ecuación diferencial relacionada con este operador es relevante en Ciencias de la Tierra ya que representa el estado estacionario de la ecuación de transporte. Se logró resolver sistemas con 100,000,000 de grados de libertad, utilizando hasta 441 procesadores disponibles.

Palabras clave: DVS-framework, Métodos Iterativos de Subestructuración, Ecuación Diferencial General de Segundo Orden, DVS-BDDC, BiCGstab, Cómputo en paralelo de alto desempeño.

Received: July 20, 2024; Accepted: August 13, 2025; Published on-line: January 1, 2026.

Editorial responsibility: Dra. Graciela Herrera-Zamarrón

* Corresponding author: Alicia De la Mora Cebada, alidelmc@ciencias.unam.mx
[1] Universidad Nacional Autónoma de México, Posgrado en Ciencias de la Tierra, Mexico City, Mexico.
[2] Universidad Nacional Autónoma de México, Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Mexico City, Mexico.
[3] Universidad Nacional Autónoma de México, Instituto de Geofísica, Mexico City, Mexico.

Alicia De la Mora Cebada; Ernesto Rubio Acosta; Ismael Herrera Revilla
https://doi.org/10.22201/igeof.2954436xe.2026.65.1.1813

1. Introduction

Domain Decomposition Methods (DDM) trace back to the 19th century, well before the emergence of the DVS framework. Their origin lies in the work of H. A. Schwarz, who introduced an iterative technique to construct harmonic functions over composite domains. This procedure, now known as the alternating Schwarz method (Quarteroni and Valli, 1999) laid the foundation for future developments in domain decomposition. The core idea of splitting a complex domain into simpler subregions, an early example of the divide and conquer paradigm, gained renewed importance with the advent of parallel computing architectures (Smith *et al.*, 2004 and Toselli and Widlund ,2005).

Over time, a rich variety of DDM strategies has emerged, including overlapping and non-overlapping approaches tailored to different classes of Partial Differential Equations (PDEs). These methods are often formulated through hybrid strategies such as the Schwarz method, Steklov-Poincaré (Schur complement) formulations, Lagrange multiplier methods, and least-squares control formulations (Quarteroni and Valli 1999; Dolean *et al.*, 2015). Important advances in non-overlapping domain decomposition include the development of the Balancing Domain Decomposition method (BDD) (Mandel, 1993) and the Balancing Domain Decomposition by Constraints method (BDDC) (Dohrmann, 2003), which provide highly effective preconditioners for large-scale parallel finite element simulations.

Together these developments created the broader theoretical and computational context in which the DVS framework was later introduced as novel paradigm for substructuring and iterative solution methodology.

A comprehensive treatment of general DDM techniques can be found in foundational references such as Proceedings of Domain Decomposition Meetings. (n.d.), (Herrera *et al.*, 2003; Smith *et al.*, 2004; Toselli and Widlund, 2005; Mathew, 2008; Dolean *et al.*, 2015) can be approached in two ways. The first approach to DDM focuses on the development of new discretization methods (Herrera, 1985; Herrera, 2000; Herrera *et al.*, 2002; Herrera *et al.*, 2007; Rubio, 2008; De la Mora, 2021), while the second aims to develop new methods for computational parallelization. The DVS methodology (DVS framework), initially proposed by Herrera I., belongs to the latter approach.

The background of the DVS framework is described as follows. Herrera (2007) asserts that a general theory of the Finite Element Method (FEM) should be developed based on a space of piecewise functions that are discontinuous across element boundaries. This idea is adapted to Domain Decomposition Methods (DDM) by considering subdomains instead of elements. In Herrera (2008), a new method for iterative substructuring is introduced that does not require Lagrange multipliers, making it suitable for the previously mentioned function space. A unified approach to formulate Dual-Primal DDM without Lagrange multipliers is presented in Herrera and Yates (2009); Herrera and Yates (2010); Herrera and Yates (2011), allowing direct application to matrices regardless of the differential equations they originate from. This framework allows the derivation of iterative substructuring methods like BDDC (Mandel and Dohrmann, 2003; Dohrmann 2003; Mandel *et al.*, 2005) and FETI-DP (Farhat *et al.*, 2000; Farhat *et al.* 2001) through a so called "round-trip" approach, which consists of transitioning from a global formulation to local subdomain problems and then back to a global solution. In Herrera *et al.* (2014), the DVS framework is formally established, defined by DDM with non-overlapping subdomains and non-overlapping nodes. The discretized problem is adressed within an expanded vector space known as the *Derived Vector Space* (DVS), which employs a strictly block-diagonal matrix. This setup improves the parallelization of numerical algorithms, as each block (submatrix) represents a separate subdomain processed independently by a single processor.

Four preconditioned algorithms, DVS-BDDC, DVS-FETI-DP, DVS-Primal, and DVS-Dual are introduced, along with their non-preconditioned versions. The use of symmetric matrices is investigated in Herrera *et al.* (2014), using a symmetric second-order operator as a case study, while earlier contributions include Herrera and Rosas (2013), Carrillo-Ledesma *et al.* (2013) and Carrillo (2013). Herrera and Contreras (2016) and Contreras (2016) focus on the isotropic linear elasticity operator. Additionally, Herrera-Revilla *et al.* (2020) highlights the superlinear performance of algorithms within the DVS framework for problems with one million degrees of freedom. Recently, an improved version called Enhanced DVS has been proposed in Herrera (2021) and is still in development.

A challenge of the DVS framework outlined in Herrera *et al.* (2014) is that it cannot be used for problems involving non-symmetric matrices. Even when the original problem is altered by multiplying by the transposed matrix, resulting in a symmetric matrix, it does not satisfy condition (4.3) of Herrera *et al.* (2014). Therefore, it is necessary to revise the DVS framework to extend its applicability to non-symmetric matrices.

It is worth noting that the DVS framework was specifically developed to numerically solve partial differential equations (PDEs) in the field of Computational Geosciences. This work is part of a broader methodology for physical-mathematical, numerical, and computational modeling with an axiomatic approach proposed in Herrera and Pinder (2012).

The paper is organized as follows: First, it outlines the standard DDM framework. Second, it reviews the DVS framework and the related problem posed in DVS space, proposing an equivalence theorem between this and the conventional problem. The article introduces the DVS-Schur and DVS-BDDC algorithms for non-symmetric matrices, employing BiCGstab as the

global solver on dual nodes. Third, it analyzes the algorithmic complexity of these methods to explain their superlinear speed-up, accompanied by numerical examples related to the general second-order differential operator. Finally, the paper concludes with concluding remarks.

1.1 Notation

To maintain clarity and consistency throughout this manuscript, we adopt the following notational conventions to distinguish between quantities defined on the original mesh, those in the derived vector space (DVS), and their restrictions.

Spaces and objects:

- Quantities associated with the *original node set* (i.e., the fine mesh) are denote with an overline. For example:

$$\bar{u}, \quad \bar{f}, \quad \bar{M}, \quad \bar{X}, \quad \bar{W} = \bar{W}(\bar{X}).$$

- Quantities in the *derived vector space (DVS) W = W(X)*, constructed via node replication from domain decomposition, are written without decoration:

$$u, \ \ f, \ \ A.$$

- A tilde denotes *restrictions* of vectors or matrices to subsets of nodes:

$$\widetilde{A}_{MN}, \quad \widetilde{u}_M, \quad \widetilde{f}_N.$$

Operators and matrices:

- $R : \bar{W} \to W_{j0}$ is the injection (lifting) operator from the original space to the space of continuous derived vectors.
- $a : W \to W_{j_0}$ is the symmetric averaging operator.
- $j : W \to W_{a_0}$ is the jump operator. These satisfy the identity $a + j = I$.
- The matrix $A^D = \sum_\gamma A^\gamma$ represents the global block-diagonal system matrix.

Inner products and restricted forms:

- The standard Euclidean inner product is denoted by $u \cdot f$.
- Notations such as $W(I)$, $W(\pi)$, and $W(\Delta)$ denote derived vectors restricted to interior, primal, and dual nodes, respectively.
- Matrices and vectors may appear in block-restricted form. For instance, $\widetilde{A}_{MN}$ is the restriction of A to node pairs $M \times N$.

Schur complement:

Given a partition $X = M \cup N$, the generalized Schur complement of A with respect to N is denoted by:

$$\underline{\underline{\sigma}}^M_{NN}\{A\} = \widetilde{A}_{NN} - \widetilde{A}_{NM}\left(\widetilde{A}_{MM}\right)^{-1}\widetilde{A}_{MN},$$

and the corresponding right-hand side as:

$$\widehat{f}^M_N = \widetilde{f}_N - \widetilde{A}_{NM}\left(\widetilde{A}_{MM}\right)^{-1}\widetilde{f}_M.$$

Double underlining highlights that $\underline{\underline{\sigma}}^M_{NN}$ is an operator, and the use of curly braces $\{\cdot\}$ indicates that it acts functionally on the matrix A.

Remark on the overline symbol: The overline is used in two contexts. When applied to spatial domains such as $\bar{\Omega}_\alpha$, it denotes the topological closure. When used with vectors, matrices, or node sets (e.g., $\bar{f}, \bar{u}, \bar{X}$), it indicates that the object belongs to the original, high-resolution space.

The notation is applied consistently to make derivations more transparent and to clearly separate original, derived, and restricted components.

2. Standard Framework

2.1 Original Problem

Let a well-defined boundary value problem that involves either a linear equation or a system of linear equations:

$$\begin{aligned} &\mathcal{L}u = f_\Omega \quad \text{in } \Omega \\ &\text{with b.c.} \quad u(\partial\Omega) = 0, \end{aligned} \tag{1}$$

which is defined within an open domain Ω, with an exterior boundary $\partial\Omega$, and subject to boundary conditions ensuring that the problem is well-defined. The linear differential operator is $\mathcal{L}$ specified as follows. For simplicity, homogeneous Dirichlet boundary conditions $u(\partial\Omega) = 0$ are assumed. The formulation (1) is referred to as the *original problem*.

2.2 Standard Discretization (Fine Mesh)

To apply a standard-nodal discretization method such as FEM to equation (1), a fine mesh is introduced in Ω. This leads to the standard discretization of the original problem, resulting in a linear algebraic system of equations:

$$\bar{M}\bar{u} = \bar{f}. \tag{2}$$

The set of nodes introduced by the high quality mesh is referred the *original node set* $\overline{X}$. The vectors $\overline{u}$ and $\overline{f}$ are functions

defined on $\overline{X}$, hence they are called *original vectors*. The matrix $\overline{M}$ is identified as the *matrix of the original system*, while $\overline{f}$ is designated as the original vector of *independent terms*.

Since each original node $p \in \overline{X}$ can be assigned multiple degrees of freedom, an original vector is a function $\bar{u}(p) : \bar{X} \to \mathbb{R}^d$, where d is the number of degrees of freedom per original node. The i-th component of is denoted as $\bar{u}\,(p)$. If d=1, the index i is omitted.

The set of all original vectors, together with the conventional operations $\{+,\bullet\}$, forms the vector space of original vectors $\overline{W}=\overline{W}\,(\overline{X})$ defined on $\overline{X}$.

2.3 Standard Domain Decomposition (Coarse Mesh)

To utilize standard Domain Decomposition Methods (DDM), a coarse mesh is introduced in Ω. This coarse mesh induces a partition of Ω into E open subdomains $\{\Omega_\alpha | \alpha$=1,...,$E\}$. A partition of Ω satisfies the following properties: $\bar{\Omega} = \bigcup_{\forall\alpha} \bar{\Omega}_\alpha$ and $\Omega_i \cap \Omega_j = \varnothing$ if $i \neq j$. The interface between subdomains, also known as the interior *boundary*, is defined as $B = \bigcup_{\forall(i,j)|i\neq j} \bar{\Omega}_i \cap \bar{\Omega}_j$. The coarse mesh also induces a partition of $\overline{X}$ into two subsets: *the subset of original nodes on the interior boundary* $\overline{\Gamma}=\{p\in\overline{X} \mid p\in B\}$ and the *subset of original interior nodes* $\overline{I}=\{p\in\overline{X} \mid p\notin B\}$ ($\overline{I}=\overline{X}-\overline{\Gamma}$). Furthermore, the coarse mesh induces a partition of $\overline{I}$ into E subsets of original interior nodes per subdomain $\bar{I}_\alpha = \left\{p \in \bar{I} \mid p \in \bar{\Omega}_\alpha\right\}$ for $\alpha = 1,...,E$. Finally, the subsets of original nodes are defined individually for each subdomain $\bar{X}^\gamma = \left\{p \in \bar{X} \mid p \in \bar{\Omega}_\gamma\right\}$ with $\gamma = 1,\ldots E$. All these $\bar{X}^\gamma$ form a complete cover of $\overline{X}$. Figure 1 illustrates an example of standard discretization and standard DDM.

Despite having a partition of Ω with non-overlapping subdomains, the original nodes $p\in\overline{\Gamma}$ may belong to multiple subdomains. This characteristic defines the Standard Framework as a DDM with non-overlapping subdomains but overlapping nodes. In such cases, the structure of the original system matrix can be arranged as follows:

$$\bar{M} = \begin{bmatrix} \bar{M}_{\bar{I}_1\bar{I}_1} & \cdots & 0 & \bar{M}_{\bar{I}_1\Gamma} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & \bar{M}_{\bar{I}_E\bar{I}_E} & \bar{M}_{\bar{I}_E\Gamma} \\ \bar{M}_{\Gamma\bar{I}_1} & \cdots & \bar{M}_{\Gamma\bar{I}_E} & \bar{M}_{\Gamma\Gamma} \end{bmatrix}. \tag{3}$$

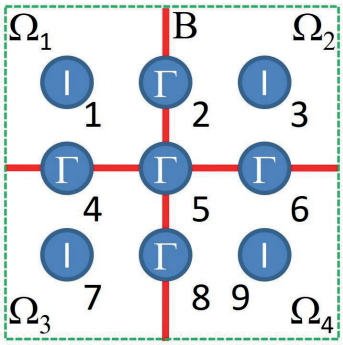
Notice that overlapping interior boundary nodes present challenges when applying computational parallelization techniques, primarily because the original system matrix does not form a block-diagonal structure.

In contrast, the DVS framework is a Domain Decomposition Method (DDM) characterized by non-overlapping subdomains and eliminates overlapping nodes. This framework strictly produces a block-diagonal matrix, where each block represents a submatrix linked to a single subdomain and is processed by a single processor. This setup greatly simplifies the implementation of high-performance computational parallelization techniques.

3. DVS framework

3.1 Derived Nodes

Each original node $p \in \overline{X}$ belongs to the closure of one or several subdomains $\overline{\Omega}_\alpha$. If $p \in \bar{I}$, it is associated with a single subdomain, but if $p \in \overline{\Gamma}$, it is with more than one. Then a derived node is defined as the pair (p,α) for each subdomain where $p \in \overline{\Omega}_\alpha$. This process consists of a replication (or "derivation") of original



$$\overline{X} = \{1,2,3,4,5,6,7,8,9\}$$

$$\overline{\Gamma} = \{2,4,5,6,8\}$$

$$\overline{I} = \{1,3,7,9\}$$

$$\bar{I}_1 = \{1\}, \bar{I}_2 = \{3\}, \bar{I}_3 = \{7\}, \bar{I}_4 = \{9\}$$

Figure 1. Example of standard discretization and standard DDM. Consider a fine mesh generating a set of original nodes $\overline{X}$ with cardinality 9, and a coarse mesh dividing Ω into 4 subdomains Ω_α. B represents the interface between subdomains or the interior boundary. The sets are shown: $\overline{\Gamma}$ (original nodes on the interior boundary), $\bar{I}$ (original interior nodes), $\bar{I}_\alpha$ (original interior nodes per subdomain $\bar{X}^\alpha$), and $\bar{X}^\alpha$ (original nodes per subdomain).

nodes into a certain number of associated derived nodes; that is, each original node is replicated $p \in \overline{X}$ into a set of derived nodes associated with the original node p, $Z(p)=\{(p,\alpha) \mid p \in \overline{\Omega}_\alpha\}$. The cardinality of $Z(p)$ is called the node multiplicity $m(p)=|Z(p)|$. The set of derived nodes is defined as $X = \bigcup_{p\in\bar{X}} Z(p)$.

Indirectly, the coarse mesh induces a partition of X into E subsets $\{X^\gamma \mid \gamma = 1,\ldots,E\}$ called subsets of derived nodes per subdomain, such that $X^\gamma = \{(p,\alpha) \in X \mid \alpha = \gamma\}$. Thus, each derived node $(p,\alpha) \in X$ belongs to a single subset X^γ and consequently, we have a DDM with non-overlapping subdomains and non-overlapping nodes.

3.1.1 Derived Nodes Classification

The following subsets of derived nodes are defined. First let the partition of X into the subset of interior boundary derived nodes $\Gamma = \bigcup_{p\in\bar{\Gamma}} Z(p)$ and the subset of interior derived nodes $I = \bigcup_{p\in\bar{I}} Z(p)$, such that $X=I \cup \Gamma$ and $I \cap \Gamma = \varnothing$. Note that if $p \in \Gamma$ then $m(p) > 1$ and if $p \in I$, $m(p) = 1$.

Second, partition the set of interior boundary derived nodes Γ into the subset of dual nodes Δ and the subset of primal nodes π, where $\Gamma = \Delta \cup \pi$ and $\Delta \cap \pi = \varnothing$. The purpose of primal nodes π is to enforce continuity constraints across the interior boundary B. In practice, the selection of primal nodes depends on the geometry and dimension of the problem. For instance, in structured 2D domains, primal nodes π are commonly chosen at the corners of the subdomains (i.e., vertices), or alternatively at both the vertices and the midpoints of the edges. The remaining interface nodes form the dual set Δ. In 3D domains, the primal nodes correspond to degrees of freedom associated with subdomain vertices and averages over edges, while dual nodes are typically associated with constraints over shared faces. This classification is consistent with the construction of corner and edge constraints presented in Dohrmann (2003).

Finally, other useful subsets of derived nodes include $\Pi = I \cup \pi$ in the context of the operator S, (which will be defined later) and $\Sigma = I \cup \Delta$ in the context of the operator S^{-1}, (which will also be defined later).

Figure 2 and 3 illustrate the structure of the derived nodes and their classification into primal and dual sets.

3.2 Derived Vectors

The vector u is a function defined on the set of derived nodes X, hence it is called a derived vector. Therefore, a derived vector is a function $u(p,\alpha) : X \to \mathbb{R}^d$, where d is the number of degrees of freedom (DOF) per derived node. The i-th component of $u(p, \alpha)$ is denoted as $u(p,\alpha,i)$.

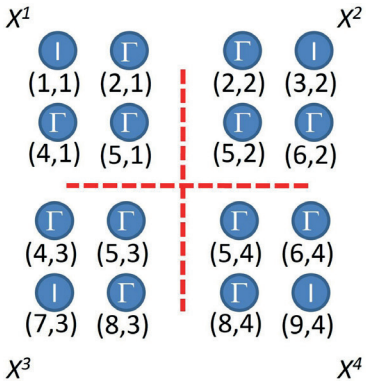
3.3 Vector Spaces of Derived Vectors

The set of all derived vectors, combined with standard operations $\{+,\bullet\}$, constitutes the vector space of derived vectors $W=W(X)$ defined on the set of derived nodes X.

Due to the partition of X into $\{X^1,X^2,\ldots,X^E\}$, the vector space of derived vectors W is segmented into E subspaces $W^\Gamma \subset W$ such that $W^\gamma = W(X^\gamma)$ defined within X^γ, referred to as the vector spaces of derived vectors by subdomain. Consequently, the space of derived vectors $u = u^1 + u^2 + \ldots + u^E$ such that $u^\gamma \in W(X^\gamma)$ is the direct sum of all these subspaces W.

The above results in each vector $u \in W$ have a unique decomposition $u = u^1 + u^2 + \ldots + u^E$ such that $u^\gamma \in W(X^\gamma)$.

Another partition it allows X is I,π,Δ. Then, let $W(I)$ the vector space of vectors derived from interior nodes; $W(\pi)$ the vector space of vectors derived from primal nodes; and $W(\Delta)$ the vector space of vectors derived from dual nodes. Correspondingly, each



$$Z(1)=\{(1,1)\}, m(1)=1$$
$$Z(2)=\{(2,1),(2,2)\}, m(2)=2$$
$$Z(4)=\{(4,1),(4,3)\}, m(4)=2$$
$$Z(5)=\{(5,1),(5,2),(5,3),(5,4)\},$$
$$m(5)=4$$
$$X^1=\{(1,1),(2,1),(4,1),(5,1)\}$$
$$X^2=\{(2,2),(3,2),(5,2),(6,2)\}$$
$$X^3=\{(4,3),(5,3),(7,3),(8,3)\}$$
$$X^4=\{(5,4),(6,4),(8,4),(9,4)\}$$

Figure 2. Example of derived nodes and their sets. Continuing the example presented in Figure 1, the following sets are shown: Γ (interior boundary derived nodes), I (interior derived nodes), X^γ (derived nodes per subdomain), $Z(p)$ (derived nodes associated with an original node) and $m(p)$ (node multiplicity).

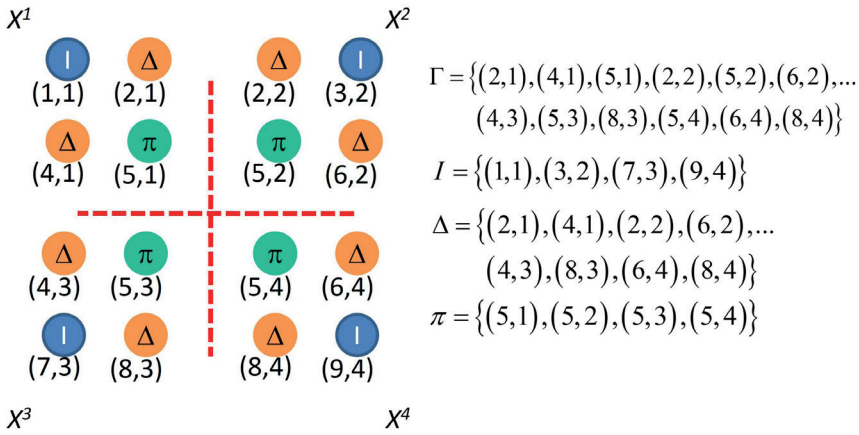


Figure 3. Classification of interior boundary nodes into primal (π) and dual (Δ) sets. This partition plays a central role in primal-dual domain decomposition methods.

vector $u \in W(X)$ has a unique decomposition $u = u_I + u_\pi + u_\Delta$ such that, $u_I \in W(\mathrm{I})$, $u_\pi \in W(\pi)$ and $u_\Delta \in W(\Delta)$.

3.3.1 Vector Space of Derived "Continuous" or "Zero Jump" Vectors

Let the natural injection of the vector space of original vectors $\overline{W}$ into the derived vector space W be defined for a vector $\overline{u} \in \overline{W}$ and given a vector $u \in W$ as:

$$u(p,\alpha) = \overline{u}(p). \tag{4}$$

The image set of the natural injection of $\overline{W}$ into W is called the vector space of "continuous" or "zero-jump" derived vectors $W_{j0} \subset W$.

The term "continuous vector" is used analogously to describe a function that maintains continuity across a boundary. Thus, a vector is said to be "continuous" if all values in each set $Z(p)$, considered individually, are equal. Ensuring this condition holds is particularly pertinent for interior boundary derived nodes $p \in \Gamma$, as $m(p) > 1$ (interior derived nodes $p \in I$ inherently satisfy this criterion).

The projection operator $R: \overline{W} \to W_{j0}$ is then defined as the natural injection of $\overline{u} \in \overline{X}$ into X

$$u = R\,\overline{u}. \tag{5}$$

Note that the operator $R: \overline{W} \to W_{j0}$ is bijective, which guarantees the existence of its inverse $R^{-1}: W_{j0} \to \overline{W}$. It is also important to note that if one considered $R : \overline{W} \to W$ instead, the operator would not be invertible, as it would merely be injective (one-to-one), but not surjective.

The assumption that the projection operator $R : \overline{W} \to W_{j0}$ is invertible is essential for establishing the equivalence between the standard discretization of the original problem (2) and its associated formulation in the derived vector space. Figure 4 illustrates the natural injection R.

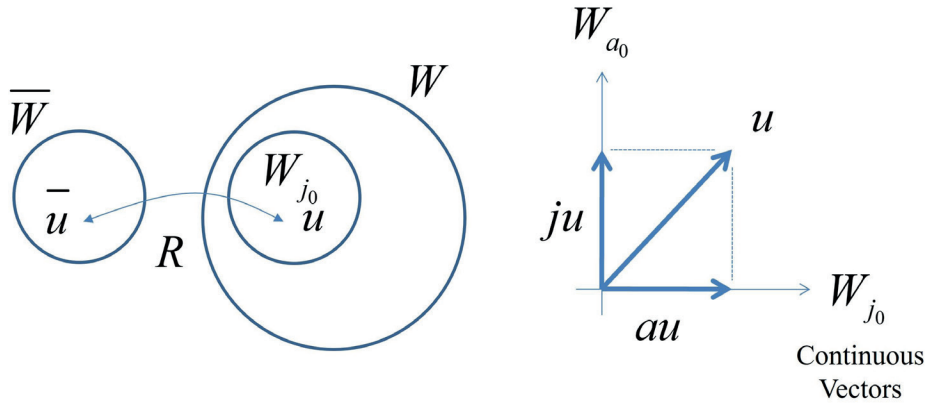


Figure 4. Natural injection R: $\overline{W} \to W_{j0}$, vector space of continuous derived vectors W_{j0} (jump-zero) and average zero W_{a0}.

3.3.2 Vector Space of Derived "Zero Average" Vectors

Let be the orthogonal complement of the vector space of continuous derived vectors $W_{a_0} \subset W$ or "zero average" vectors. In this way: $W = W_{j_0} \oplus W_{a_0}$.

The average $a : W \to W_{j_0}$ and jump $j : W \to W_{a_0}$ operators are introduced such that they satisfy the property $j + a = I$. The average operator applied to $u \in X$ is defined as:

$$(au)(p,\alpha) = \frac{1}{m(p)} \sum_{(p,\alpha)\in Z(p)} u(p,\alpha). \tag{6}$$

While the jump operator applied to $u \in X$ is calculated using the property:

$$ju = u - au. \tag{7}$$

If $u \in W_{a_0}$ then $au = 0$ and consequently $u = ju$. On the other hand, if $u \in W_{j_0}$ then $ju = 0$ and $u = au$.

3.4 Inner Products

3.4.1 Inner Product of the Vector Space of Original Vectors

Let $\bar{u},\bar{v} \in \overline{W}$. At each original node $p \in \overline{X}$ the standard Euclidean inner product is used, here denoted by $\bar{u}(p) \bigodot \bar{v}(p) \equiv \sum_{i=1}^{d} \bar{u}(p,i) \bigodot \bar{v}(p,i)$. Then the inner product of the original vector space is defined as:

$$\bar{u} \cdot \bar{v} \equiv \sum_{p\in\bar{X}} \bar{u}(p) \odot \bar{v}(p) \quad \text{where} \quad \bar{u}(p) \odot \bar{v}(p) \equiv \sum_{i=1}^{d} \bar{u}(p,i) \cdot \bar{v}(p,i) \tag{8}$$

3.4.2 Inner Product of the Derived Vector Space

Let $u,v \in W$. At each original node $(p, \alpha) \in X$ the standard Euclidean inner product is used, $u(p,\alpha) \odot v(p,\alpha) \equiv \sum_{i=1}^{d} u(p,\alpha,i)v(p,\alpha,i)$. Then the inner product of the derived vector space is defined as:

$$u \cdot v \equiv \sum_{(p,\alpha)\in X} u(p,\alpha) \odot v(p,\alpha)$$

where

$$u(p,\alpha) \odot v(p,\alpha) \equiv \sum_{i=1}^{d} u(p,\alpha,i)v(p,\alpha,i) \tag{9}$$

3.4.3 Equalities between Inner Products

Let the matrix of multiplicities of the original nodes $\overline{m}$ be a diagonal matrix such that:

$$\bar{m} = [\bar{m}_{pq}] \quad \text{where: } \bar{m}_{pq} = m(p)\delta_{pq}, \tag{10}$$

and let the matrix of multiplicities of derived nodes m, a diagonal matrix where:

$$\bar{m} = [\bar{m}_{(p,\alpha)(q,\beta)}] \quad \text{where: } m_{(p,\alpha)(q,\beta)} = m(p)\delta_{(p,\alpha)(q,\beta)}, \tag{11}$$

the equality between inner products is stated in the following theorem.

Theorem 1 (Equality between Inner Products:). If $\bar{u}$, $\bar{v} \in \overline{X}$, $u,v \in X$, $u = R\,\bar{u}$ *and* $v = R\,\bar{v}$ *then the following statement holds true*:

$$\text{a) } u \cdot v = (\overline{m\,u}) \cdot \bar{v} = \overline{m\,u} \cdot (\bar{v}), \tag{12}$$

$$\text{b) } (m^{-1}u) \cdot v = u \cdot (m^{-1}v) \cdot \bar{v} = \bar{u} \cdot \bar{v}. \tag{13}$$

The proof is given in **Appendix 1**.

3.5 System Matrix in the Derived Vector Space

Let the matrix of the original system be:

$$\overline{M} = [\overline{M}_{\mathrm{pq}}], \tag{14}$$

which can be considered as a transformation in the vector space of original vectors $\overline{M} : \overline{W} \to \overline{W}$.

Let $m(p,q)$ the pairwise multiplicity of derived vector nodes p and q, defined as:

$$m(p,q) = |\gamma_{pq}| \quad \text{where:} \quad \gamma_{pq} = \left\{\gamma \mid p \in \bar{X}^{\gamma} \wedge q \in \bar{X}^{\gamma}\right\}, \tag{15}$$

that is, the multiplicity for each pair of derived nodes p and q signifies the number of subdomains to which both nodes in the pair belong p y q simultaneously.

Then, let the matrix $\bar{M}^{\gamma}$: $\overline{W} \to \overline{W}$ associated with the subdomain $\bar{\Omega}_{\gamma}$ defined as:

$$\bar{M} = \left[\bar{M}_{pq}\right] \quad \text{where:} \quad \bar{M}^{\gamma}_{pq} = \begin{cases} \frac{\bar{M}_{pq}}{m(p,q)} & \text{if } p \in \bar{X}^{\gamma} \wedge q \in \bar{X}^{\gamma} \\ 0 & \text{if } p \notin \bar{X}^{\gamma} \vee q \notin \bar{X}^{\gamma} \end{cases}. \tag{16}$$

Let the matrix A^{γ}:$W \rightarrow W$ associated with the subdomain $\bar{\Omega}_{\gamma}$ defined as:

$$A^{\gamma} = \left[A^{\gamma}_{(p,\alpha)(q,\beta)}\right] \quad \text{where:} \quad A^{\gamma}_{(p,\alpha)(q,\beta)} = \bar{M}^{\gamma}_{pq}\delta_{\alpha\gamma}\delta_{\beta\gamma} \tag{17}$$

Finally, let the matrix A^{D}:$W \rightarrow W$ the matrix of the system of the associated problem in the derived vector space defined as:

$$A^{D} = \sum_{\gamma=1}^{E} A^{\gamma}. \tag{18}$$

Which can be considered as transformation of the derived vector space A^{D}:$W \rightarrow W$.

From the way the matrices A^{γ}:$W \rightarrow W$ are constructed, the matrix A^{D} is a block diagonal matrix (diagonal matrix of submatrices per subdomain). Please refer to **Appendix 2** titled "Construction of the Matrix A^{D}" where is shown in detail this construction.

3.6 Fundamental Theorem of DVS framework

The theorem is based on the following statement, which asserts the equality between inner products weighted by matrices.

Theorem 2 (Equalities between inner products weighted by matrices). *If* $\overline{u}, \overline{v} \in \overline{X}$, $u,v \in X$, $u = R\overline{u}$ *and* $v = R\overline{v}$ *and for definition of* $A^{D} = \sum_{\gamma=1}^{E} A^{\gamma}$ then it is fulfilled:

$$\text{a) } \bar{M} = \sum_{\gamma=1}^{E} \bar{M}^{\gamma}, \tag{19}$$

$$\text{b) } v \cdot A^{\gamma} u = \bar{v} \cdot \bar{M}^{\gamma} \bar{u}, \tag{20}$$

$$\text{c) } v \cdot A^{D} u = \bar{v} \cdot \bar{M} \bar{u}. \tag{21}$$

The proof is given in **Appendix 1**.

The following theorem establishes the equivalence between the solutions of the standard discretization of the original problem (2) and of the DVS discretization associated with the original problem but posed in the derived vector space (22). Note that such solutions are assumed to exist.

Theorem 3 (Fundamental Theorem). *Let the original problem* $\mathcal{L}u = f_{\Omega}$, *its standard discretization* $\overline{M}\,\overline{u} = \bar{f}$ *defined in the vector space of original vectors* $\overline{X}$, *and the considerations* $\overline{u} \in \overline{X}$, $u,v \in X$, *such that* $u = R\overline{u}$ *and* $f = m^{-1} R\bar{f}$.

Let the problem associated with the standard discretization of the original problem, defined within the derived vector space, be termed as the DVS discretization of the original problem:

$$aA^{D}au = af \quad \& \quad ju = 0. \tag{22}$$

Then the solution of the standard discretization of the original problem and the solution of the DVS discretization of the original problem posed in the space of derived vectors, if they exist, are equivalent, meaning they have the same solution.

$$\overline{M}\,\overline{u} = \overline{f} \Leftrightarrow \{aA^{D}au = af \quad \& \quad ju = 0\}. \tag{23}$$

Proof. Let $v = R\overline{v}$. Let the optimization problem be posed in the original vector space $\overline{X}$: $\overline{v}$ ="arg" min $\overline{J}(\overline{v})$ where $\bar{J}(\bar{v}) = \frac{1}{2}\bar{v} \cdot \bar{M}\bar{v} - \bar{v} \cdot \bar{f}$. Let the optimization problem formulated in the derived vector space X: u ="arg" min $J(v)$, where $J(v) = \frac{1}{2} v \cdot aA^{D}av - v \cdot af$.

While it will not be shown that these functionals have optimal values, it will be shown that $\overline{J}(\overline{v}) = J(v)$ when $v = R\overline{v}$. This implies that if there are optimal values for each of them, they must be equal. That is:

$$\frac{1}{2}\bar{v} \cdot \bar{M}\bar{v} - \bar{v} \cdot \bar{f} = \frac{1}{2} v \cdot aA^{D}av - v \cdot af.$$

Step 1: Since a it is symmetrical:

$$\frac{1}{2} v \cdot aA^{D}av - v \cdot af = \frac{1}{2} av \cdot A^{D}av - av \cdot f.$$

Step 2: Since $jv = 0$, then $av = v$:

$$\frac{1}{2} av \cdot A^{D}av - av \cdot f = \frac{1}{2} v \cdot A^{D}v - v \cdot f$$

Step 3: Considering the premises $u = R\overline{u}$, $v = R\overline{v}$, $f = m^{-1} R\bar{f}$:

$$\frac{1}{2} v \cdot A^{D}av - v \cdot f = \frac{1}{2} R\bar{v} \cdot A^{D} R\bar{v} - R\bar{v} \cdot m^{-1} R\bar{f}.$$

Step 4: By theorem 2,c) in the first summand and by theorem 1,b) in the second summand:

$$\frac{1}{2} R\bar{v} \cdot A^{D} R\bar{v} - R\bar{v} \cdot m^{-1} R\bar{f} = \frac{1}{2}\bar{v} \cdot \bar{M}\bar{v} - \bar{v} \cdot \bar{f}.$$

Finally, since $\overline{J}(\overline{v}) = J(v)$ then their optimal solution will be the same.

3.7 DVS Basic Problem

Before outlining the basic problem, it is necessary to first establish some considerations regarding the average operator a.

3.7.1 Considerations on the Average Operator

Before formulating the basic problem, two initial consider-

ations are introduced. Initially, the following factorization of the average operator a:

$$a = a^{\Delta} a^{\pi} = a^{\pi} a^{\Delta}$$

where

$$a^{\Delta} = \begin{bmatrix} I_{II} & 0 & 0 \\ 0 & I_{\pi\pi} & 0 \\ 0 & 0 & a_{\Delta\Delta} \end{bmatrix}, \quad a^{\pi} = \begin{bmatrix} I_{II} & 0 & 0 \\ 0 & a_{\pi\pi} & 0 \\ 0 & 0 & I_{\Delta\Delta} \end{bmatrix} \tag{24}$$

where $a_{\Delta\Delta}$ is the restriction of the matrix a to the set of dual nodes $\Delta\times\Delta$ (average operator that only affects dual nodes Δ), $a_{\pi\pi}$ is the restriction of the matrix to the set of primal nodes $\pi\times\pi$ (average operator that only affects primal nodes π). It is important to note that a_{II}, the restriction of the matrix to the set of interior nodes $I\times I$ is an identity matrix: $a_{II}=I_{II}$ (the average operator does not affect interior nodes I).

Second Consideration. The following assertion about the vector of independent terms of (22) is also introduced: $a\overline{f}=\bar{f}$. The above is due to the fact that $f\in W_{j0}$ it is a continuous derived vector, since by construction $f=m^{-1}R\tilde{f}$ it is the result of applying the injection $R:\bar{W}\rightarrow W_{j_0}$ of the original vector $\overline{f}\in\overline{W}$.

3.7.2 DVS Basic Problem

Therefore, beginning with the DVS discretization of the original problem (22) and taking into account the considerations made: $a=a^{\Delta}a^{\pi}=a^{\pi}a^{\Delta}$ and $a\,f=f$, have that $a^{\Delta}a^{\pi}A^{D}a^{\pi}a^{\Delta}u=f$.

Here, the continuity conditions across the interior boundary are introduced using the primal nodes, which are defined by the matrix $A=a^{\pi}A^{D}a^{\pi}$.

Finally, let the Basic DVS Problem be:

$$a^{\Delta}A\,a^{\pi}u=f \qquad \& \qquad ju=0 \qquad \text{where:} \qquad A=a^{\pi}A^{D}a^{\pi}. \tag{25}$$

3.7.3 Solution Strategy

The basic DVS problem (25) allows a structured solution through three nested levels. At the first level (outer level), the problem is solved globally for all dual nodes Δ within the vector space of derived vectors $W(\Delta)$. This involves computing the Schur complement over dual nodes while considering the node set $X=\Delta\cup\Pi$. Here, the system matrix is associated with $a^{\Delta}A\,a^{\Delta}$.

At the second level (middle level), the problem is solved globally for all primal nodes π within the vector space of derived vectors $W(\pi)$. Primal nodes are crucial for enforcing continuity conditions across the interior boundary. This involves computing the Schur complement over primal nodes π while considering the node set $\Pi=\pi\cup I$. Here, the system matrix is associated with $a^{\pi}A^{D}a^{\pi}$.

At the third level (inner level), the problem is solved locally within each subdomain for all interior nodes I within the vector space of derived vectors . Here, the matrices of the local systems are associated with A^{γ}.

3.7.4 Generalized Schur Complement

The matrix $A=a^{\pi}A^{D}a^{\pi}$ can be divided into four separate parts $A=A_{\Pi\Pi}+A_{\Pi\Delta}+A_{\Delta\Pi}+A_{\Delta\Delta}$ if the set X is partitioned into two subsets $X=\Delta\cup\Pi$. Alternatively, the matrix A can be represented by submatrices as follows:

$$A=\begin{bmatrix} \widetilde{A}_{\Pi\Pi} & \widetilde{A}_{\Pi\Delta} \\ \widetilde{A}_{\Delta\Pi} & \widetilde{A}_{\Delta\Delta} \end{bmatrix} \tag{26}$$

where $\tilde{A}_{\Pi\Delta}$ is a restriction of the matrix $A_{\Delta\Delta}$ to the set of nodes $\Pi\times\Delta$, and similarly for the other submatrices. Although this ordered matrix A is not block diagonal, each of its component submatrices, such as $\tilde{A}_{\Pi\Pi}$, $\tilde{A}_{\Pi\Delta}$, $\tilde{A}_{\Delta\Pi}$, $\tilde{A}_{\Delta\Delta}$, maintains this property. Therefore, its Schur complement $\sigma^{\Pi}_{\Delta\Delta}(A)$ over the dual nodes Δ, considering the node set $\Delta\bigcup\Pi$, forms a block diagonal matrix:

$$S=\underline{\underline{\sigma^{\Pi}_{\Delta\Delta}}}\{A\}=\widetilde{A}_{\Delta\Delta}-\widetilde{A}_{\Delta\Pi}\left(\widetilde{A}_{\Pi\Pi}\right)^{-1}\widetilde{A}_{\Pi\Delta} \tag{27}$$

Please refer to **Appendix 3** titled "Generalized Schur Complement". It's important to note that both matrices S and A^{D} are block diagonal matrices. This characteristic is crucial for enabling the computational parallelization of DVS algorithms, where each processor handles a distinct subdomain independently. The construction of matrix S is shown in detail in the Appendix 3.

4. DVS-Schur Algorithm and Operator S

4.1 DVS-Schur Algorithm

The DVS-Schur algorithm solves the system of equations resulting from computing the Schur complement over the dual nodes of the DVS Basic Problem (25), represented in submatrix notation as:

$$\begin{bmatrix} \widetilde{I}_{\Pi\Pi} & 0 \\ 0 & a_{\Delta\Delta} \end{bmatrix}\begin{bmatrix} \widetilde{A}_{\Pi\Pi} & \widetilde{A}_{\Pi\Delta} \\ \widetilde{A}_{\Delta\Pi} & \widetilde{A}_{\Delta\Delta} \end{bmatrix}\begin{bmatrix} I_{\Pi\Pi} & 0 \\ 0 & a_{\Delta\Delta} \end{bmatrix}\begin{bmatrix} \widetilde{u}_{\Pi} \\ \widetilde{u}_{\Delta} \end{bmatrix}=\begin{bmatrix} \widetilde{f}_{\Pi} \\ \widetilde{f}_{\Delta} \end{bmatrix}. \tag{28}$$

Then its Schur complement over dual nodes Δ considering the set of nodes $\Pi \bigcup \Delta$ is:

$$a_{\Delta\Delta} S a_{\Delta\Delta} \widetilde{u}_{\Delta} = \widehat{f}_{\Delta}^{\Pi} \quad \& \quad j_{\Delta\Delta} \widetilde{u}_{\Delta} = 0 \tag{29}$$

The DVS-Schur algorithm solves the system of equations (29). As previously indicated, the solution is structured into three nested levels. Figure 6 provides an overview of these three solution levels within the DVS-Schur algorithm.

4.2 First level: Global Iterative Solver on Dual nodes

At the outermost level, the first tier, the system of equations (29) is solved using a global iterative solver applied to all dual nodes Δ. Given the matrix's non-symmetric nature in this context, the Stabilized Biconjugate Gradient Method, BiCGstab algorithm 7.7 in Saad (2003), is employed by Van Der Vorst (1992) and Trefethen and Bau (1997). After solving this system of equations, the solution for the dual nodes Δ is projected onto the interior and primal nodes $\Pi = \pi \bigcup I$.

The algorithm is as follows:

Algorithm 1 First level Schur

Require: $\widehat{f}_{\Delta}^{\Pi}$

Ensure: u

1: $\widetilde{u}_{\Delta} \leftarrow Global\_BiCGstab\left\{a_{\Delta\Delta} S a_{\Delta\Delta}, \widehat{f}_{\Delta}^{\Pi}\right\}$
2: $\widetilde{u}_{\Pi} \leftarrow Proy_{\Delta}^{\Pi}\{\widetilde{u}_{\Delta}\}$
3: $u \leftarrow Join\_vectors\{\widetilde{u}_{\Pi}, \widetilde{u}_{\Delta}\}$

Since $A = a^{\pi} A^{D} a^{\pi}$ then:

$$S = \widetilde{A}_{\Delta\Delta}^{D} - \widetilde{A}_{\Delta\Pi}^{D} a_{\pi\pi}^{\Pi} \left(\widetilde{A}_{\Pi\Pi}\right)^{-1} a_{\pi\pi}^{\Pi} \widetilde{A}_{\Pi\Delta}, \tag{30}$$

$$\widehat{f}_{\Delta}^{\Pi} = \tilde{f}_{\Delta} - a_{\Delta\Delta} \widetilde{A}_{\Delta\Pi}^{D} a_{\pi\pi}^{\Pi} \left(\widetilde{A}_{\Pi\Pi}\right)^{-1} \tilde{f}_{\Pi}, \tag{31}$$

$$Proy_{\Delta}^{\Pi}\{\widetilde{u}_{\Delta}\} = \left(\widetilde{A}_{\Pi\Pi}\right)^{-1} \left(\tilde{f}_{\Pi} - a_{\pi\pi}^{\Pi} \widetilde{A}_{\Pi\Delta}^{D} a_{\Delta\Delta} \widetilde{u}_{\Delta}\right). \tag{32}$$

At this initial stage, the operator S must be applied to a vector $w_{\Delta} \in W(\Delta)$. The details of the operator S are explained at the second level. Note that $\boldsymbol{a}_{\pi\pi}^{\Pi}$ it is an extension of $a_{\pi\pi}$ the set of nodes $\Pi \times \Pi$ (conformable average operator that only affects the primal nodes π), which means:

$$a_{\pi\pi}^{\Pi} = \begin{bmatrix} I_{II} & 0 \\ 0 & a_{\pi\pi} \end{bmatrix}. \tag{33}$$

4.3 Second level: Operator S, Global Iterative Solver on Primal nodes

In the operator $S = \widetilde{A}_{\Delta\Delta}^{D} - \widetilde{A}_{\Delta\Pi}^{D} a_{\pi\pi}^{\Pi} \left(\widetilde{A}_{\Pi\Pi}\right)^{-1} a_{\pi\pi}^{\Pi} \widetilde{A}_{\Pi\Delta}$ and eqs. (30), (31) and (32) it is required to apply the operator $\left(\widetilde{A}_{\Pi\Pi}\right)^{-1}$ to a vector $w_{\Pi} \in W(\Pi)$. The latter is calculated using the following system of equations:

$$\tilde{v}_{\Pi} = \left(\widetilde{A}_{\Pi\Pi}\right)^{-1} \widetilde{w}_{\Pi} \quad \Rightarrow \quad \widetilde{A}_{\Pi\Pi} \tilde{v}_{\Pi} = \widetilde{w}_{\Pi}. \tag{34}$$

Which in submatrix notation is:

$$\begin{bmatrix} \widetilde{A}_{II} & \widetilde{A}_{I\pi} \\ \widetilde{A}_{\pi I} & \widetilde{A}_{\pi\pi} \end{bmatrix} \begin{bmatrix} \tilde{v}_{I} \\ \tilde{v}_{\pi} \end{bmatrix} = \begin{bmatrix} \widetilde{w}_{I} \\ \widetilde{w}_{\pi} \end{bmatrix}. \tag{35}$$

Next, the Schur complement is applied over primal nodes π, taking into account the partition of the node set $\Pi = \pi \bigcup I$, is:

$$\underset{=\pi\pi}{\sigma^{I}} \left\{\widetilde{A}_{\Pi\Pi}\right\} \tilde{v}_{\pi} = \widehat{w}_{\pi}^{I} \quad \& \quad j_{\pi\pi} \tilde{v}_{\pi} = 0 \tag{36}$$

This system of equations is solved using a global iterative solver on all primal nodes π. Once again, BiCGstab (Saad 2003) is utilized for this purpose. Upon solving these equations, the solution for the primal nodes π is then projected onto the internal nodes I.

The algorithm is the following:

Algorithm 2 Second level Schur

Require: $\widetilde{w}_{\Pi}$

Ensure: $\widetilde{v}_{\Pi}$

1: $\widetilde{w}_{\pi}, \widetilde{w}_{I} \leftarrow Split\_vector\{\widetilde{w}_{\Pi}\}$
2: $\widetilde{v}_{\pi} \leftarrow BiCGstab\left\{\sigma_{\pi\pi}\left\{\widetilde{A}_{\Pi\Pi}\right\}, \widehat{w}_{\pi}^{I}\right\}$
3: $\widetilde{v}_{I} \leftarrow Proy_{\pi}^{I}\{\widetilde{w}_{\pi}\}$
4: $\widetilde{v}_{\Pi} \leftarrow Join\_vectors\{\widetilde{v}_{\pi}, \widetilde{v}_{I}\}$

Given the condition that $A = a^{\pi} A^{D} a^{\pi}$ holds, we have:

$$\underset{=\pi\pi}{\sigma}\left\{\widetilde{A}_{\Pi\Pi}\right\} = a_{\pi\pi}\left(\widetilde{A}_{\pi\pi}^{D} - \widetilde{A}_{\pi I}^{D}\left(\widetilde{A}_{II}\right)^{-1} \widetilde{A}_{I\pi}\right) a_{\pi\pi} \tag{37}$$

$$\widehat{w}_{\pi}^{I} = \widetilde{w}_{\pi} - a_{\pi\pi} \widetilde{A}_{\pi I}^{D}\left(\widetilde{A}_{II}\right)^{-1} \widetilde{w}_{I} \tag{38}$$

$$Proy_{\pi}^{I}\{\tilde{v}_{\pi}\} = \left(\widetilde{A}_{II}\right)^{-1} \left(\widetilde{w}_{I} - \widetilde{A}_{I\pi}^{D} a_{\pi\pi} \tilde{v}_{\pi}\right) \tag{39}$$

At this second level, it is required to apply the operator $\left(\widetilde{A}_{II}\right)^{-1}$ to a vector $w_{I} \in W(I)$. The details of the operator $\left(\widetilde{A}_{II}\right)^{-1}$ are explained in the third level.

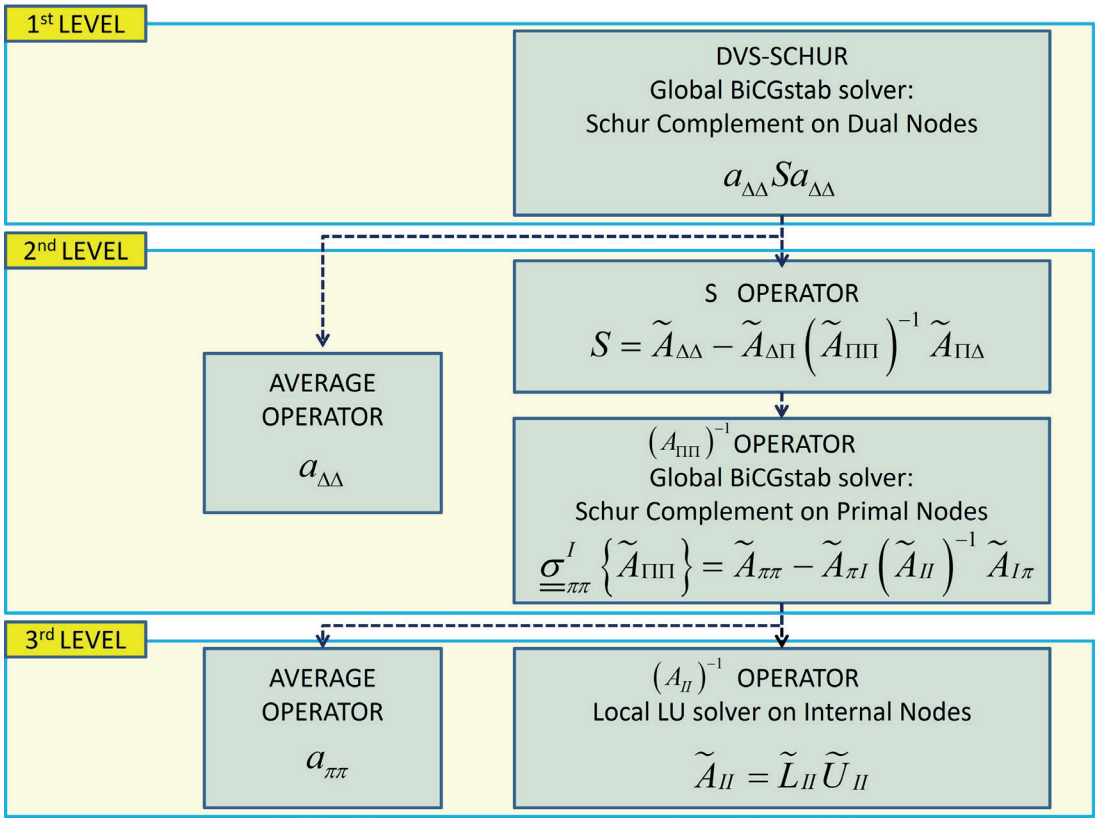


Figure 5. UML activity diagram summarizing the DVS-Schur Algorithm and its three solution levels. The second level shows the Operator *S*.

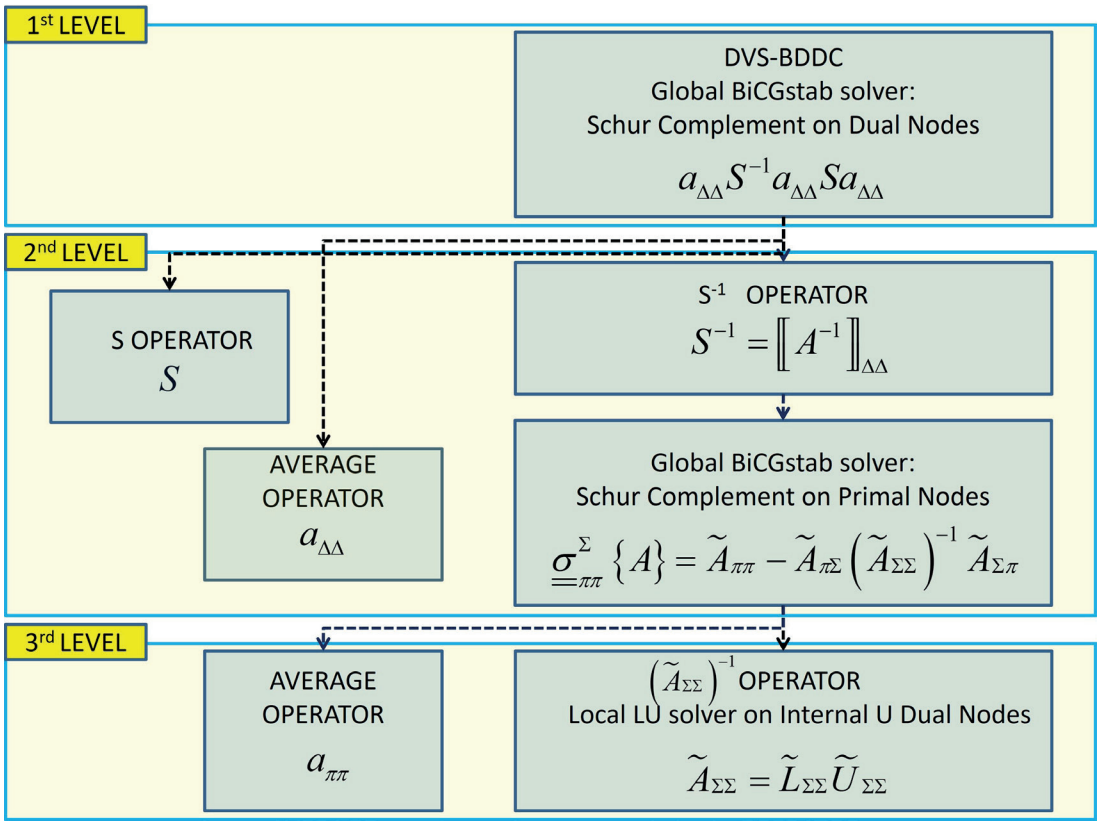


Figure 6. UML activity diagram representing the DVS-BDDC algorithm and its three solution levels. The second level summarizes the operator S^{-1}. Figure 6 shows the summary of the operator *S*.

4.4 Third level: Local Solver on Internal nodes

The operator $\left(\widetilde{A}_{II}\right)^{-1}$ needs to be applied to a vector in eqs. (37), (38) and (39). The latter is determined by the following system of equations:

$$\tilde{v}_I = \left(\widetilde{A}_{II}\right)^{-1} \tilde{w}_I \Rightarrow \widetilde{A}_{II}\tilde{v}_I = \tilde{w}_I \tag{40}$$

Under the condition that $A = a^{\pi}A^{D}a^{\pi}$, then $\widetilde{A}_{II} = \widetilde{A}^{D}_{II}$. This system of equations is solved locally within each subdomain using a direct method. In this work, LU decomposition is employed, along with appropriate forward or backward substitution (Burden *et al.*, 2016).

Algorithm 3 Third level DVS-Schur

Require: $\widetilde{A}^{D}_{II}$

Ensure: $\widetilde{v}_I$

1: $L_{II}, U_{II} \leftarrow LU\_Decomposition\left\{\widetilde{A}^{D}_{II}\right\}$
2: $y \leftarrow Forward\_Sustitution\{L_{II}, \widetilde{w}_I\}$
3: $\widetilde{v}_I \leftarrow Backward\_Sustitution\{U_{II}, y\}$

The structure of the DVS-Schur algorithm is presented below using a UML (Unified Modeling Language) activity diagram. This representation clearly shows the three solution levels and highlights their interactions, including the use of the Schur complement operator S at the intermediate level. This representation emphasizes the logical structure and dependencies of the procedure, including parallel and conditional operations.

5. DVS-BDDC Algorithm and Operator S^{-1}

5.1 DVS-BDDC Algorithm

By preconditioning the DVS-Schur algorithm (29) with aS^{-1}, we obtain the DVS-BDDC algorithm:

$$a_{\Delta\Delta}S^{-1}a_{\Delta\Delta}Sa_{\Delta\Delta}\tilde{u}_{\Delta} = a_{\Delta\Delta}S^{-1}\tilde{f}^{\Pi}_{\Delta} \quad \& \quad j_{\Delta\Delta}\tilde{u}_{\Delta} = 0. \tag{41}$$

As previously discussed, this system is resolved through three nested levels. Figure 7 presents a summary of the three solution tiers within the DVS-BDDC algorithm.

5.2 First level: Global Iterative Solver on Dual nodes

As in section 4.2, BiCGstab, algorithm 7.7 in (Saad 2003) is used. The algorithm is as follows:

Algorithm 4 First level DVS-BDDC

Require: $a_{\Delta\Delta}S^{-1}a_{\Delta\Delta}Sa_{\Delta\Delta}, a_{\Delta\Delta}S^{-1}\widehat{f}^{\Pi}_{\Delta}$

Ensure: u

1: $\widetilde{u}_{\Delta} \leftarrow Global\_BiCGstab\left\{a_{\Delta\Delta}S^{-1}a_{\Delta\Delta}Sa_{\Delta\Delta}, a_{\Delta\Delta}S^{-1}\widehat{f}^{\Pi}_{\Delta}\right\}$
2: $\widetilde{u}_{\Pi} \leftarrow Proy^{\Pi}_{\Delta}\{\widetilde{u}_{\Delta}\}$
3: $u \leftarrow Join\_vectors\{\widetilde{u}_{\Pi}, \widetilde{u}_{\Delta}\}$

The terms $\widehat{f}^{\Pi}_{\Delta}$ and $Proy^{\Pi}_{\Delta}\{\widetilde{u}_{\Delta}\}$ are defined in eqs. (31) and (32) respectively.

At this initial level, it is necessary to apply both the operator S and the operator S^{-1} to a vector $w_{\Delta} \in W(\Delta)$. The details of the operator S, eq. (30), are elaborated in section 4.3, while the details of the operator S^{-1} are explained at the second level.

5.3 Second level: Operator S^{-1}, Global Iterative Solver on Primal nodes

To calculate the application of the operator S^{-1} on the vector $w_{\Delta} \in W(\Delta)$ in eq. (41) the following lemma is used.

Lemma 4 (Toselli and Widlund (2005), section 4.3:). *Considering that* $w = \begin{bmatrix} 0_{\Pi} \\ w_{\Delta} \end{bmatrix}$ *then it is fulfilled:*

$$v_{\Delta} = S^{-1}w_{\Delta} = [\![A^{-1}w]\!]_{\Delta}, \tag{42}$$

where $v_{\Delta} = [\![v]\!]_{\Delta}$ *represents the restriction operator that confines a vector to dual nodes* Δ *(elements corresponding to nodes other than dual nodes* Δ *are discarded).*

Please refer to **Appendix 1** for detailed proof of this result.

The above is calculated using the following system of equations:

$$v = A^{-1}\, w \quad \Rightarrow \quad Av = w, \tag{43}$$

which in submatrix notation is:

$$\begin{bmatrix} \widetilde{A}_{\Sigma\Sigma} & \widetilde{A}_{\Sigma\pi} \\ \widetilde{A}_{\pi\Sigma} & \widetilde{A}_{\pi\pi} \end{bmatrix} \begin{bmatrix} \tilde{v}_{\Sigma} \\ \tilde{v}_{\pi} \end{bmatrix} = \begin{bmatrix} \tilde{w}_{\Sigma} \\ \tilde{w}_{\pi} \end{bmatrix}. \tag{44}$$

Therefore, its Schur complement over primal nodes π, considering the partition of the node set $X = \pi \cup \Sigma$, is:

$$\underset{=\pi\pi}{\sigma^{\Sigma}}\{A\}\tilde{v}_{\pi} = \widehat{w}^{\Sigma}_{\pi} \quad \& \quad j_{\pi\pi}\tilde{v}_{\pi} = 0, \tag{45}$$

This set of equations is resolved using a global iterative solver applied to all primal nodes π. BiCGstab (Saad 2003) is employed once again for this purpose. After solving these equations, the solution for the primal nodes is projected onto the nodes Σ.

DVS_Schur Algorithm	Operators	Algorithmic Complexity of Computations	Algorithmic Complexity of Communications
Level 3: Local Backward &Forward Substitution Solver for Internal Nodes	$\left(A^D_{\Sigma\Sigma}\right)^{-1}$	$O\left(\frac{n_\Delta^2}{p^2}+\frac{n_I^2}{p^2}\right)$	without communications
Level 2: Global BiCGstab Solver for Primal Nodes	$S^{\sim 1}$	$O\left(n_\pi \frac{n_\Delta^2}{p^2}+n_\pi \frac{n_I^2}{p^2}\right)$	$O\left(\frac{n_\pi^2}{p}+n_\pi \log(p)\right)$
Level1: Global BiCGstab Solver for Dual Nodes	$a_{\Delta\Delta}S^{\sim 1}a_{\Delta\Delta}Sa_{\Delta\Delta}$	$O\left(n_\pi \frac{n_\Delta^3}{p^2}+n_\pi n_\Delta \frac{n_I^2}{p^2}\right)$	$O\left(\frac{n_\Delta^2}{p}+n_\Delta \log(p)\right)$
LU Decomposition	$A^D_{\Sigma\Sigma}$	$O\left(\frac{n_\Delta^3}{p^3}+\frac{n_I^3}{p^3}\right)$	without communications

Figure 7. Details of the algorithmic complexity of the DVS-Schur algorithm for each of its sections. The notation $\mathcal{O}(f(n))$ means that the execution time is proportional to $f(n)$.

The algorithm proceeds as outlined below:

Algorithm 5 Second level DVS-BDDC

Require: $\widetilde{0}_I, \widetilde{w}_\Delta$

Ensure: $\widetilde{v}_\Delta, \widetilde{v}_I i$

1: $\widetilde{w}_\Sigma \leftarrow Join\_vectors\left\{\widetilde{0}_I, \widetilde{w}_\Delta\right\}$

2: $\widetilde{v}_\pi \leftarrow Global\_BiCGstab\left\{\underline{\sigma}^\Sigma_{=\pi\pi}\{A\}, \widehat{w}^\Sigma_\pi\right\}$

3: $\widetilde{v}_\Sigma \leftarrow Proy^\Sigma_\pi\{\widetilde{v}_\pi\}$

4: $\widetilde{v}_\Delta, \widetilde{v}_I \leftarrow Split\_vector\{\widetilde{v}_\Sigma\}$

▷ /* discard $\widetilde{v}_I$ */

Since $A = a^\pi A^D a^\pi$, we have that:

$$\sigma^\Sigma_{\pi\pi}\{A\} = a_{\pi\pi}\left(\widetilde{A}^D_{\pi\pi} - \widetilde{A}^D_{\pi\Sigma}\left(\widetilde{A}_{\Sigma\Sigma}\right)^{-1}\widetilde{A}_{\Sigma\pi}\right)a_{\pi\pi}, \tag{46}$$

$$\widehat{w}^\Sigma_\pi = \widetilde{w}_\pi - a_{\pi\pi}\widetilde{A}^D_{\pi\Sigma}\left(\widetilde{A}_{\Sigma\Sigma}\right)^{-1}\widetilde{w}_\Sigma, \tag{47}$$

$$Proy^\Sigma_\pi\{\tilde{v}_\pi\} = \left(\widetilde{A}_{\Sigma\Sigma}\right)^{-1}\left(\widetilde{w}_\Sigma - \widetilde{A}^D_{\Sigma\pi}a_{\pi\pi}\tilde{v}_\pi\right). \tag{48}$$

At this second level, it is required to apply the operator $(\tilde{A}_{\Sigma I})^{-1}$ to a vector $w_\Sigma \in W(I)$. The details of the operator $(\tilde{A}_{\Sigma I})^{-1}$ are explained in the third level.

5.4 Third level: Local Solver on Internal nodes

The operator $(\tilde{A}_{\Sigma\Sigma})^{-1}$ needs to be applied to a vector in eqs. (46), (47) and (48). The result of this application is determined by the following system of equations:

$$\tilde{v}_\Sigma = \left(\widetilde{A}_{\Sigma\Sigma}\right)^{-1}\widetilde{w}_\Sigma \Rightarrow \widetilde{A}_{\Sigma\Sigma}\tilde{v}_\Sigma = \widetilde{w}_\Sigma \tag{49}$$

Under the condition that $A = a^\pi A^D a^\pi$, then $\widetilde{A}_{\Sigma\Sigma} = \widetilde{A}^D_{\Sigma\Sigma}$. This system of equations is resolved locally within each subdomain using a direct method. Specifically, LU decomposition is employed, along with appropriate forward or backward substitution as described in Burden *et al.* (2016).

Algorithm 6 Third level DVS-BDDC

Require: $\widetilde{A}^D_{\Sigma\Sigma}$

Ensure: $\widetilde{v}_\Sigma$

1: $L_{\Sigma\Sigma}, U_{\Sigma\Sigma} \leftarrow LU\_Decomposition\left\{\widetilde{A}^D_{\Sigma\Sigma}\right\}$

2: $y \leftarrow Forward\_Sustitution\{L_{\Sigma\Sigma}, \widetilde{w}_\Sigma\}$

3: $\widetilde{v}_\Sigma \leftarrow Backward\_Sustitution\{U_{\Sigma\Sigma}, y\}$

The structure of the DVS-BDDC algorithm is illustrated below using a UML activity diagram. This representation shows the different solution levels and the main operators involved in the procedure.

6. Algorithmic Complexity

The DVS-Schur algorithm consists of two main components. The first step perform local LU factorizations of the matrices A_{II}^{D} in parallel. Each subdomain computation is assigned to one of the p available processors. The algorithmic complexity, AC, of this operation is $\mathcal{O}\left(n_I^3/p^3\right)$ (Cormen *et al.*, 2009; Kleinberg and Tardos, 2006 and Herrera-Revilla *et al.*, 2020). This process is executed without communication overhead. The second part entails employing Global BiCGstab for Dual Nodes using the operator $a_{\Delta\Delta}\, S^{-1} a_{\Delta\Delta}$, which has an algorithmic complexity of $\mathcal{O}\left(n_\Delta^2/p + n_\pi n_\Delta n_I^2/p^2\right)$for computations and $\mathcal{O}\left(n_\Delta^2/p + n_\Delta \log(p)\right)$ for communications. Further details regarding the algorithmic complexity of the DVS-Schur algorithm are illustrated in Figure 7.

In contrast, the DVS-BDDC algorithm is comprised of two main components. The first involves performing local LU factorization of matrices $A_{\Sigma\Sigma}^{D}$ in parallel, which has an algorithmic complexity, AC, for computations of $\mathcal{O}\left(n_\Sigma^3/p^3\right)$. This process operates without communication requirements. The second part involves employing Global BiCGstab for Dual Nodes using the operator $a_{\Delta\Delta}S^{-1}a_{\Delta\Delta}Sa_{\Delta\Delta}$, which has an algorithmic complexity of $\mathcal{O}\left(n_\pi n_\Delta^3/p^2 + n_\pi n_\Delta n_I^2/p^2\right)$ for computations and $\mathcal{O}\left(n_\Delta^2/p + n_\Delta \log(p)\right)$ for communications. Further specifics regarding the algorithmic complexity of the DVS-BDDC algorithm are shown in Figure 8.

The DVS-Schur and DVS-BDDC algorithms operate in two distinct cases. In the first case, when the number of interior nodes *I* per subdomain exceeds the number of dual nodes Δ the algorithmic complexity (AC) of the BiCGstab global solver for Dual nodes in DVS-Schur is $\mathcal{O}\left(n_I^2\right)$, and for DVS-BDDC is $\mathcal{O}\left(n_I^2\right)$. This situation typically arises with fewer subdomains (processors). In the second case, when the number of dual nodes Δ is greater than the number of internal nodes *I* per subdomain, the AC of the BiCGstab global solver for Dual nodes in DVS-Schur is $\mathcal{O}\left(n_\Delta^2\right)$, and for DVS-BDDC is $\mathcal{O}\left(n_\Delta^3\right)$. This scenario is more common with a larger number of subdomains (processors). Considering the algorithmic complexity, it is evident that the first case is more efficient than the second. Additionally, communications increase accordingly, $\mathcal{O}\left(n_\Delta^2/p + n_\Delta \log(p)\right)$, further supporting this conclusion.

DVS_Schur Algorithm	Operators	Algorithmic Complexity of Computations	Algorithmic Complexity of Communications
Level 3: Local Backward &Forward Substitution Solver for Internal Nodes	$\left(A_{II}^{D}\right)^{-1}$	$O\left(\frac{n_I^2}{p^2}\right)$	without communications
Level 2: Global BiCGstab Solver for Primal Nodes	S	$O\left(\frac{n_\Delta}{p} + n_\pi \frac{n_I^2}{p^2}\right)$	$O\left(\frac{n_\pi^2}{p} + n_\pi \log(p)\right)$
Level1: Global BiCGstab Solver for Dual Nodes	$a_{\Delta\Delta}Sa_{\Delta\Delta}$	$O\left(\frac{n_\Delta^2}{p} + n_\Delta n_\pi \frac{n_I^2}{p^2}\right)$	$O\left(\frac{n_\Delta^2}{p} + n_\Delta \log(p)\right)$
LU Decomposition	A_{II}^{D}	$O\left(\frac{n_I^3}{p^3}\right)$	without communications

Figure 8. Details of the algorithmic complexity of the DVS-BDDC algorithm for each of its sections.

7. Numerical Experiments

The numerical experiments were performed on the cluster of the "Laboratorio Universitario de Cómputo de Alto Rendimiento" (LUCAR), hosted at IIMAS, UNAM ("Laboratorio Universitario de Cómputo de Alto Rendimiento" (s/f)). The cluster consisted of 5 nodes, each equipped with 4 Intel Xeon E5-4657L v2 @2.40 GHz CPUs. Each CPU featured 12 cores, capable of executing 24 processes effectively in parallel with Hyper-Threading technology. Thus, the cluster had a total of up to 240 physical cores, capable of handling up to 480 processes effectively in parallel. In terms of memory, 3 nodes were equipped with 256 GB each (64 GB per CPU), and 2 nodes had 128 GB each (32 GB per CPU).

Due to varying definitions, the term "processor" in this study is interpreted as follows ("Introduction to Parallel Computing Tutorial" (s/f)): if a physical core has the capability to effectively handle two processes concurrently, it is abstractly considered as two logical processors. Therefore, each logical processor executes a single process. Within the DVS framework, a logical processor (or simply processor) is responsible for processing a single subdomain.

The numerical experiments focused on solving a general second-order partial differential equation:

$$-\nabla\cdot(a\nabla u) + \nabla\cdot(bu) + cu = f, \tag{50}$$

where a is a positive definite symmetric matrix, b is a vector and c is a scalar. This equation holds significance in Earth Sciences as it represents the steady state of the transport equation (Herrera and Pinder 2012).

To perform the LU factorization, we employ the SciPy `splu` function, which implements the Supernodal LU method (SciPy n.d.).

Problem 1 is defined in $\Omega = [0,1]^N \subset \mathbb{R}^N$, with homogeneous Dirichlet conditions, that is $u = 0$ on $\partial\Omega$. The coefficients are given by:

$$\begin{aligned} a &= I_{N\times N}, \\ b &= [b_i]_{n\times 1} \quad \text{where } b_i = 10 \text{ for all } i, \\ c &= 1, \end{aligned} \tag{51}$$

and the exact solution is:

$$u(x_1,\ldots,x_N) = \prod_{i=1}^{N} \sin(\pi x_i). \tag{52}$$

First, an example in 2 dimensions is provided, $N = 2$. Figure 10 displays the graphical representation of the solution (52).

Table 1 presents a table that corresponds to the Amdahl analysis (Wilkinson and Allen 2005) of the DVS-Schur algorithm, detailing execution times t_p for p processors, speedups $s = t_1/t_p$, and efficiencies $e = s/p$, considering 10 million degrees of freedom (DOF).

Table 2 displays a table corresponding to the Amdahl analysis of the DVS-BDDC algorithm. Figure 10(a) illustrates the speedup graph, demonstrating superlinearity $s > p$, while Figure 10(b) depicts the efficiency graph, also demonstrating superlinearity $e > 100\%$.

There are two distinct factors contributing to the superlinearity observed in the speedup of the DVS-BDDC algorithm.

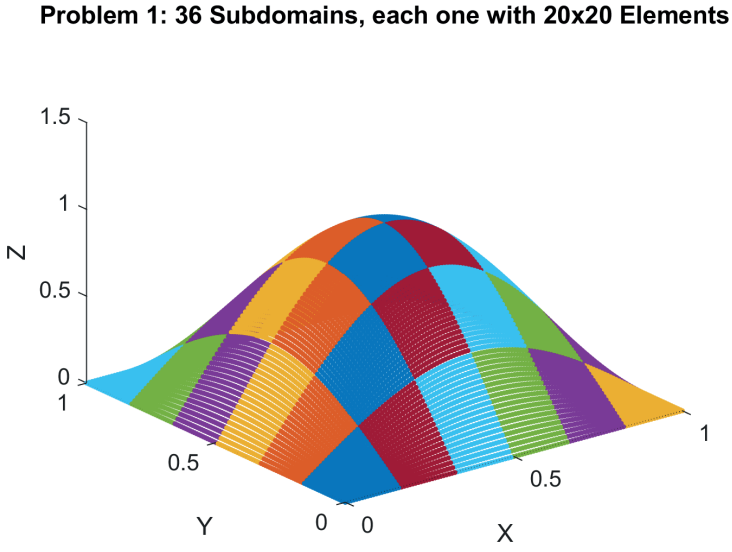


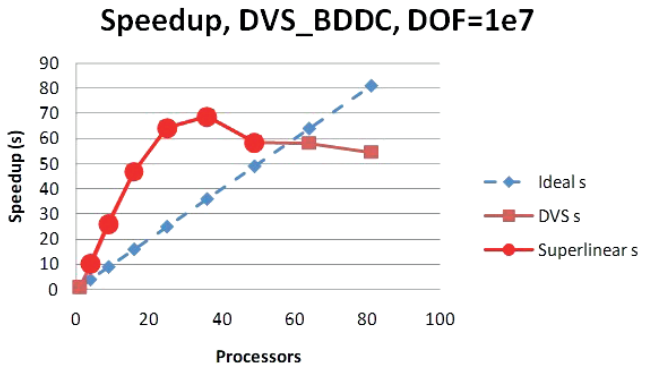
Figure 9. Graph of the solution of Problem 1, eq. (52), for two dimensions, using 36 subdomains.

Table 1. Execution times, speedups (Amdahl speedup) and efficiencies for Problem 1, for 2 dimensions, using the DVS-Schur algorithm with 10 million degrees of freedom (DOF = $1e7$). Due to computational resource limitations, the LU execution time for 1 subdomain was estimated with a quadratic fit based on interior DOF per subdomain; and the BiCGstab execution time for 1 subdomain was estimated with the average of the linear fits based on Dual DOF and Primal DOF for $p < 36$.

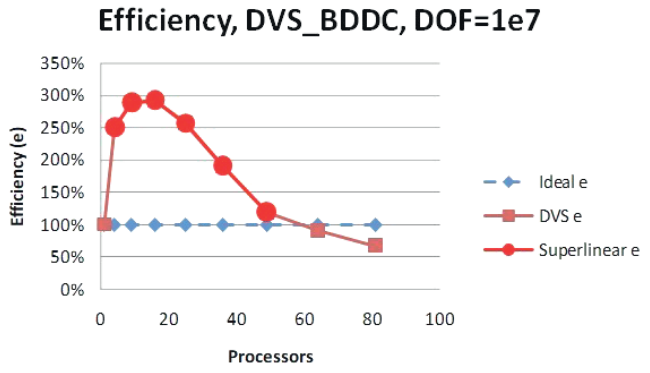
Subdo-mains	Elements by Subdomain	DoF	Dual DoF	Primal DoF	Internal DoF by Subdo-mian	Iterations (Level 1)	LU Elapsed Time	BiCGstab Elapsed Time	DVS (total) Elapsed Time	Speedup	Eficiency
1	3162 × 3162	9998244			9998244		2114.02	2401.40	4515.41	1.0	100.0%
2 × 2	1581 × 1581	9998244	25296	16	2499561	124	175.90	1046.35	1222.25	3.7	92.4%
3 × 3	1054 × 1054	9998244	37944	36	1110916	133	50.85	498.97	549.82	8.2	91.3%
4 × 4	791 × 791	10010896	50624	64	625681	146	20.79	292.82	313.61	14.4	90.0%
5 × 5	632 × 632	9985600	63200	100	399424	170	11.32	279.68	291.00	15.5	62.1%
6 × 6	527 × 527	9998244	75888	144	277729	188	7.57	252.20	259.77	17.4	48.3%
7 × 7	452 × 452	10010896	88592	196	204304	235	5.43	343.34	348.77	12.9	26.4%
8 × 8	395 × 395	9985600	101120	256	156025	207	4.77	295.79	300.56	15.0	23.5%
9 × 9	351 × 351	9979281	113724	324	123201	218	4.65	307.99	312.63	14.4	17.8%

Table 2. Execution times, speedups (Amdahl speedup) and efficiencies for Problem 1, for two dimensions, using the DVS-BDDC algorithm with 10 million degrees of freedom (DOF = $1e7$). The same considerations as in Table 1 are considered.

Subdo-mains	Elements by Subdomain	DoF	Dual DoF	Primal DoF	Internal DoF by Subdo-mian	Iterations (Level 1)	LU Elapsed Time	BiCGstab Elapsed Time	DVS (total) Elapsed Time	Speedup	Eficiency
1	3162 × 3162	9998244			9998244		4230.27	169.37	4399.63	1.0	100.0%
2 × 2	1581 × 1581	9998244	25296	16	2499561	4	351.60	87.55	439.15	10.0	250.5%
3 × 3	1054 × 1054	9998244	37944	36	1110916	6	100.48	68.95	169.43	26.0	288.5%
4 × 4	791 × 791	10010896	50624	64	625681	8	42.88	51.22	94.11	46.8	292.2%
5 × 5	632 × 632	9985600	63200	100	399424	8	24.13	44.51	68.65	64.1	256.4%
6 × 6	527 × 527	9998244	75888	144	277729	10	15.08	48.98	64.06	68.7	190.8%
7 × 7	452 × 452	10010896	88592	196	204304	10	11.05	64.53	75.58	58.2	118.8%
8 × 8	395 × 395	9985600	101120	256	156025	11	9.58	66.13	75.70	58.1	90.8%
9 × 9	351 × 351	9979281	113724	324	123201	10	9.12	71.65	80.77	54.5	67.2%



(a)Acceleration graph for the DVS-BDDC algorithm with DOF = 10^7. Note the superlinearity $s_{DVS} > s_{ideal}$, where $s_{ideal} = p$.



(b)Efficiency graph for the DVS-BDDC algorithm with DOF = 10^7. Note the superlinearity in efficiency, where $s_{DVS} > s_{ideal}$ and s_{ideal} =100%.

Figure 10. Acceleration and efficiency graphs for the DVS-BDDC algorithm with 10^7 degrees of freedom.

The first factor is the superlinear speedup of the LU factorization phase. Figure 11 provides an analysis of the superlinearity in speedup across each section of the algorithm.

The second factor is the preconditioner $a_{\Delta\Delta}S^{-1}$ used in the DVS-BDDC algorithm, as the DVS-Schur algorithm (which lacks a preconditioner) does not exhibit superlinear speedup. Figure 12 compares the number of iterations during the execution of the global BiCGstab on dual nodes (level 1 solver). While DVS-Schur involves more iterations that are computationally lighter, DVS-BDDC features fewer but more efficient computationally heavy iterations. Figure 13 compares the execution times of both algorithms, highlighting that DVS-BDDC is more efficient than DVS-Schur.

From this perspective, DVS algorithms are seen not just as Domain Decomposition Methods (DDM), but also as parallel preconditioners due to their ability to accelerate solution speeds in a superlinear manner.

However, there are two factors that counteract superlinear acceleration. The first factor is the serial fraction f_s of the algorithm (Wilkinson and Allen (2005)), which is defined as $(f_s)^{-1} = \lim_{p\to\infty} s(p)$ and measures the non-parallelizable part of the algorithm. For DVS-BDDC, the serial fraction is $f_s = 0.02$; whereas for DVS-Schur, it is $f_s = 0.07$. The second factor is the algorithmic complexity of communications, $\mathcal{O}(\log p)$.

Problem 2 is defined in $\Omega = [0,1]^N \subset \mathbb{R}^N$, with homogeneous Dirichlet conditions, that is $u = 0$ on $\partial\Omega$. The coefficients are given by:

$$\begin{aligned} &a = I_{N\times N}, \\ &b = [b_i]_{n\times 1} \quad \text{where } b_i = V = 10 \text{ for all } i, \\ &c = 0. \end{aligned} \tag{53}$$

and the exact solution is:

$$u(x_1,\ldots,x_N) = \prod_{i=1}^{N}\left(\frac{e^{Vx_i} - e^{V}}{1 - e^{V}}\right). \tag{54}$$

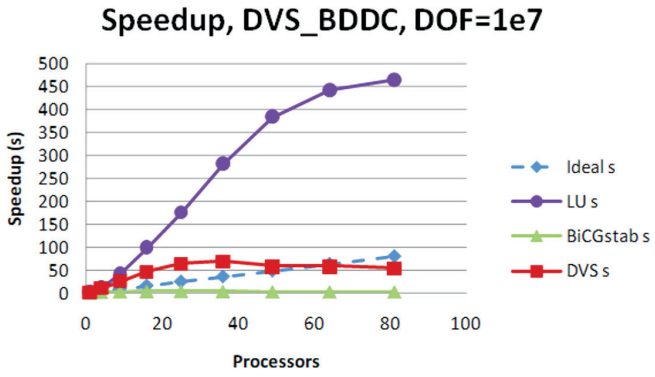


Figure 11. Acceleration graph of the sections of the DVS-BDDC algorithm. Note the superlinear acceleration of the LU factorization section.

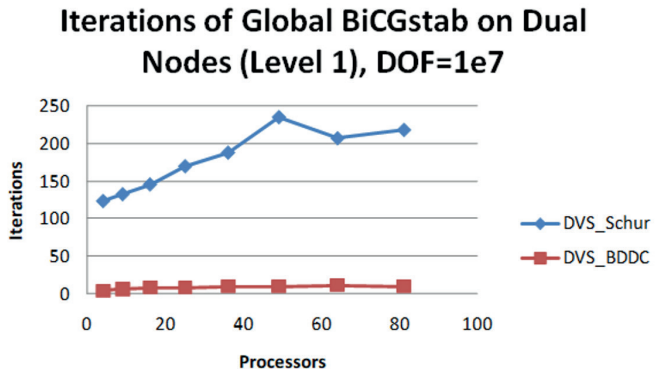


Figure 12. Iterations of the global BiCGstab on dual nodes (first level resolver) in the DVS algorithms.

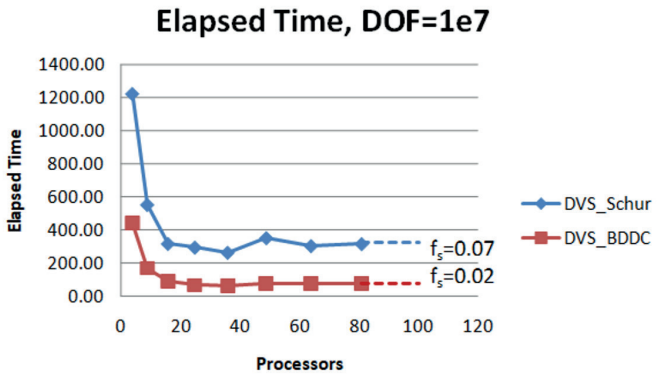


Figure 13. Comparison of execution time in DVS algorithms.

A graph of the solution for $N = 2$, eq. (54), is shown in Figure 14. Table 3 shows a table corresponding to the DVS-BDDC algorithm with execution times from 10 million DOF to 100 million DOF (with increments of 10 million) (Klawonn *et al.*, 2015).

Figure 15 depicts the convergence analysis, where first-degree polynomial Finite Element Method (FEM) was utilized, and the convergence is characterized by $\mathcal{O}\left(h^2\right)$, where h is the element size.

Table 4 displays a table detailing the execution times of the DVS-BDDC algorithm. The analysis covers scenarios with 10 million degrees of freedom (DOF) in two dimensions employing 441 processors, and in three dimensions using 343 processors.

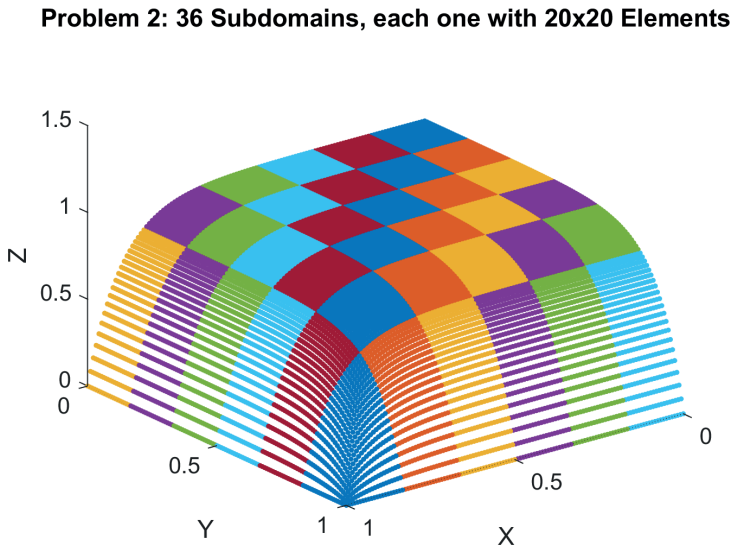


Figure 14. Graph of the solution of Problem 2, eq. (54), for two dimensions, using 36 subdomains.

Table 3. Execution times for Problem 2, for two dimensions, using the DVS-BDDC algorithm with up to 100 million degrees of freedom (DOF = $1e8$)

Subdo-mains	Elements by Subdo-main	DoF	Dual DoF	Primal DoF	Internal DoF by Subdo-mian	Iterations (Level 1)	LU Elapsed Time	BiCGstab Elapsed Time	DVS (total) Elapsed Time	Error on Dual Nodes
16 × 16	198 × 198	10036224	202752	1024	39204	15	2.10	92.16	94.26	4.0401E-07
16 × 16	280 × 280	20070400	286720	1024	78400	17	5.31	183.86	189.18	2.0212E-07
16 × 16	342 × 342	29942784	350208	1024	116964	16	9.68	262.87	272.55	1.3401E-07
16 × 16	395 × 395	39942400	404480	1024	156025	17	14.05	374.25	388.29	9.8472E-08
16 × 16	442 × 442	50013184	452608	1024	195364	17	20.26	506.24	526.49	9.2479E-08
16 × 16	484 × 484	59969536	495616	1024	234256	17	25.87	585.14	611.01	5.9626E-08
16 × 16	523 × 523	70023424	535552	1024	273529	18	32.73	751.61	784.33	5.9696E-08
16 × 16	559 × 559	79995136	572416	1024	312481	17	38.37	826.61	864.98	5.5073E-08
16 × 16	593 × 593	90022144	607232	1024	351649	17	44.92	955.57	1000.49	4.1407E-08
16 × 16	625 × 625	100000000	640000	1024	390625	17	32.89	706.22	739.10	4.3466E-08

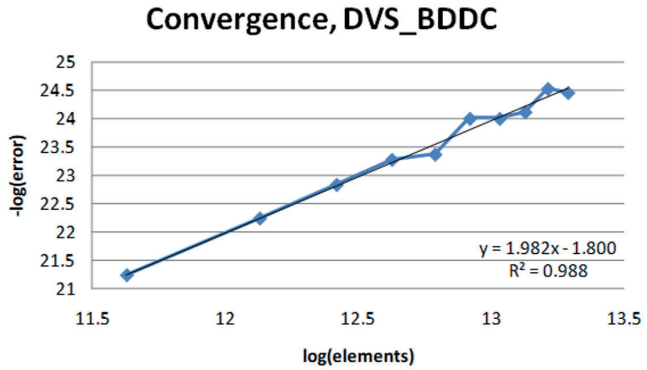


Figure 15. Convergence graph. Since FEM with linear polynomials was used, the order of convergence is $\mathcal{O}\left(h^2\right)$, where h is the element size.

Table 4. Execution times, for two and three dimensions, using the DVS-BDDC algorithm with 10 million degrees of freedom (DOF = $1e7$).

Problem N=2	Subdo-mains	Elements by Subdomain	DoF	Dual DoF	Primal DoF	Internal DoF by Subdo-mian	Iterations (Level 1)	LU Elapsed Time	BiCGstab Elapsed Time	DVS (total) Elapsed Time	Error on Dual Nodes
Problem 1	21 × 21	151 × 151	10055241	266364	1764	22801	14	0.92	81.47	82.39	1.30E-07
Problem 2	21 × 21	151 × 151	10055241	266364	1764	22801	16	0.92	83.62	84.55	4.08E-07

Problem N=3	Subdo-mains	Elements by Subdomain	DoF	Dual DoF	Primal DoF	Internal DoF by Subdo-mian	Iterations (Level 1)	LU Elapsed Time	BiCGstab Elapsed Time	DVS (total) Elapsed Time	Error on Dual Nodes
Problem 1	7 × 7 × 7	31 × 31 × 31	10218313	1977738	127596	29791	9	126.16	547.69	673.86	4.38E-05
Problem 2	7 × 7 × 7	32 × 31 × 31	10218313	1977738	127596	29791	9	126.18	618.74	744.93	1.00E-04

8. Conclusions

The main goal of this research was to expand the applicability of the Domain Decomposition Method within the Derived Vector Space (DVS framework), originally devised for symmetric matrices by Herrera in 2014, to handle non-symmetric matrices. This advancement was achieved by utilizing the Biconjugate Stabilized Gradient (BiCGstab) method as a global solver.

The DVS framework represents a non-standard Substructuring Domain Decomposition Method (DDM) that does not require Lagrange multipliers. It operates with non-overlapping subdomains and non-overlapping nodes. The discretized problem is tackled within an extended vector space known as the Derived Vector Space (DVS), where the system matrix adopts a block diagonal form. Each block corresponds to a submatrix associated with a single subdomain, enabling straightforward and efficient parallelization of numerical algorithms.

After establishing the foundational principles of the DVS framework, a Fundamental Theorem is proposed, establishing equivalence between discretized problems in the standard vector space and those in the derived vector space.

A basic DVS problem structure is outlined, enabling a structured solution in three nested levels. At the outermost level, the Schur complement system is solved on dual nodes using a global BiCGstab solver. In an intermediate level, the Schur complement system is addressed on primal nodes with a global BiCGstab solver. Finally, at the innermost level, local systems on interior nodes within each subdomain are solved using LU factorization and Gaussian substitution.

The DVS-Schur and DVS-BDDC algorithms (the preconditioned version of the former) are thoroughly explained. Based on the analysis of their algorithmic complexities (AC), they operate in two distinct cases. In the first case, when the number of internal nodes per subdomain exceeds the number of dual nodes, both algorithms exhibit an AC of the global BiCGstab solver on dual nodes (first level) are $\mathcal{O}\left(n_I^2\right)$. In the second case, when the number of dual nodes surpasses the number of interior nodes per subdomain, the ACs of the first level solver are $\mathcal{O}\left(n_\Delta^2\right)$ and $\mathcal{O}\left(n_\Delta^3\right)$, respectively. Generally, the AC of the LU factorization stands at $\mathcal{O}\left(n_I^3\right)$, while the AC associated with communications is $\mathcal{O}(\frac{n_\Delta^2}{p} + n_\Delta \log p)$.

Numerical examples in 2D and 3D illustrate the solution of the general second-order differential equation associated with the steady-state transport model relevant in Geosciences. Acceleration and efficiency analyses are conducted using Amdahl's law. The DVS-BDDC algorithm demonstrates superlinear acceleration particularly when internal nodes dominate dual nodes, attributed to the superlinear acceleration of the LU factorization section and the preconditioner $a_{\Delta\Delta}S^{-1}$. Overall, DVS-BDDC proves more efficient than DVS-Schur despite its more complex calculations, establishing DDM-DVS algorithms as effective parallel preconditioners. However, superlinear speedup is mitigated by the serial fraction of the algorithm and the AC of communications.

Additionally, a convergence analysis is performed, highlighting successful resolutions of problems involving up to 100 million degrees of freedom (DOF) and up to 441 processors.

Future work aims to incorporate a time-marching algorithm into the DVS framework to address time-dependent problems and extend its applicability to nonlinear problems.

9. Acknowledgments

The authors express gratitude for the support received from the *Posgrado en Ciencias de la Tierra*, IIMAS and IGEF, all of which are affiliated with UNAM. We also acknowledge the scholarship provided by CONACYT to Alicia Margarita de la Mora Cebada, which supported her doctoral studies.

Additionally, we would like to thank engineer Adrián Durán Chavesti for his support in managing the computing nodes of the *Laboratorio Universitario de Cómputo de Alto Rendimiento* (LUCAR) which were used in this research. We also gratefully acknowledge the computing time granted by the project LANCAD-UNAM-DGTIC-460 on the Miztli supercomputer.

We are grateful to the Reviewer and the associate editors for their constructive comments and suggestions, which helped improve the quality of this manuscript.

Finally, we thank Dr. Felipe Angeles for insightful discussions and valuable feedback throughout the development of this work."

10. References

Barney, B., & Frederick, D. (n.d.). *Introduction to parallel computing tutorial.* Lawrence Livermore National Laboratory.

Burden, R. L., Faires, J. D., & Burden, M. (2016). *Numerical analysis* (10th ed.). Cengage.

Carrillo, A. (2013). *Métodos de descomposición de dominio en el espacio de vectores derivados y su implementación computacional en paralelo*. [Tesis de Doctorado]. Universidad Nacional Autónoma de México.

Carrillo-Ledesma, A., Herrera, I., & de la Cruz, L. M. (2013). Parallel algorithms for computational models of geophysical systems. *Geofísica Internacional*, *52*(3), 293–309. doi: **https://doi.org/10.1016/S0016-7169(13)71478-8**

Contreras, I. (2016). *Procesamiento en paralelo de sistemas de EDP's*. [Tesis de Doctorado]. Universidad Nacional Autónoma de México.

Coordinación de la Investigación Científica. (2024). *Laboratorio universitario de cómputo de alto rendimiento*. Coordinación de la Investigación Científica.

Cormen, T. H., Leiserson, C. Eric., Rivest, R. L. & Stein, Clifford. (2009).

Introduction to algorithms. PHI Learning Private Limited.

De la Mora, A. (2021). *Método de elemento finito con funciones óptimas: Caso de estudio: Ecuación de elasticidad*. [Tesis de Doctorado]. Universidad Nacional Autónoma de México, Instituto de Geofísica.

Dohrmann, C. R. (2003). A Preconditioner for substructuring based on constrained energy minimization. *SIAM Journal on scientific computing*, *25*(1), 246–258. doi: **https://doi.org/10.1137/S1064827502412887**

Dolean, Victorita., Jolivet, Pierre., & Nataf, F. (2015). *An introduction to domain decomposition methods : algorithms, theory, and parallel implementation*. Society for industrial and applied mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104).

Farhat, C., Lesoinne, M., LeTallec, P., Pierson, K., & Rixen, D. (2001). FETI-DP: a dual–primal unified FETI method—part I: A faster alternative to the two-level FETI method. *International Journal for Numerical Methods in Engineering*, *50*(7), 1523–1544. doi: **https://doi.org/10.1002/nme.76**

Farhat, C., Lesoinne, M., & Pierson, K. (2000). A scalable dual-primal domain decomposition method. *Numerical linear algebra with applications*, *7*(7–8), 687–714. doi: **https://doi.org/10.1002/1099-1506(200010/12)7:7/8<687::AID-NLA219>3.0.CO;2-S**

Herrera, I. (1985). Unified formulation of numerical methods. I. Green's formulas for operators in discontinuous fields. *Numerical methods for partial differential equations*, *1*(1), 25–44. doi: **https://doi.org/10.1002/num.1690010105**

Herrera, I. (2000). Trefftz method: A general theory. *Numerical methods for partial differential equations*, *16*(6), 561–580. doi: **https://doi.org/10.1002/1098-2426(200011)16:6<561::AID-NUM4>3.0.CO;2-V**

Herrera, I. (2007). Theory of differential equations in discontinuous piecewise-defined functions. *Numerical methods for partial differential equations*, *23*(3), 597–639. doi: **https://doi.org/10.1002/num.20182**

Herrera, I., & Contreras, I. (2016). An innovative tool for effectively applying highly parallelized hardware to problems of elasticity. *Geofísica Internacional*, *55*(1). doi: **https://doi.org/10.22201/igeof.00167169p.2016.55.1.1710**

Herrera, I., de la Cruz, L. M., & Rosas-Medina, A. (2014). Nonoverlapping discretization methods for partial differential equations. *Numerical methods for partial differential equations*, *30*(5), 1427–1454. doi: **https://doi.org/10.1002/num.21852**

Herrera, I., Keyes, D., Widlund, O., & Yates, R. (2003). Domain decomposition methods in science and engineering. *Fourteenth International Conference on Domain Decomposition Methods, Cocoyoc, Mexico*.

Herrera, I., & Pinder, G. F. (2012). *Mathematical modeling in science and engineering*. Wiley. doi: **https://doi.org/10.1002/9781118207239**

Herrera, I., & Rosas-Medina, A. A. (2013). The derived-vector space framework and four general purposes massively parallel DDM algorithms. *Engineering analysis with boundary elements*, *37*(3), 646–657. doi: **https://doi.org/10.1016/j.enganabound.2012.12.003**

Herrera, I., & Yates, R. A. (2009). Unified multipliers-free theory of dual-primal domain decomposition methods. *Numerical methods for partial differential equations*, *25*(3), 552–581. doi: **https://doi.org/10.1002/num.20359**

Herrera, I., & Yates, R. A. (2010). The multipliers-free domain decomposition methods. *Numerical methods for partial differential equations*, *26*(4), 874–905. doi: **https://doi.org/10.1002/num.20462**

Herrera, I., & Yates, R. A. (2011). Multipliers-free dual-primal domain decomposition methods for nonsymmetric matrices and their numerical testing. *Numerical methods for partial differential equations*, *27*(5), 1262–1289. doi: **https://doi.org/10.1002/num.20581**

Herrera, I., Yates, R., & Diaz, M. (2002). General theory of domain decomposition: Indirect methods. *Numerical methods for partial differential equations*, *18*(3), 296–322. doi: **https://doi.org/10.1002/num.10008**

Herrera, I., Yates, R., & Rubio, E. (2007). Collocation methods: More efficient procedures for applying collocation. *Advances in engineering software*, *38*(10), 657–667. doi: **https://doi.org/10.1016/j.advengsoft.2006.10.010**

Herrera, I. (2021). The enhanced derived-vector-space Approach to domain decomposition methods. *ArXiv*.

Herrera-Revilla, I. (2008). New formulation of iterative substructuring methods without Lagrange multipliers: Neumann–Neumann and FETI. *Numerical methods for partial differential equations*, *24*(3), 845–878. doi: **https://doi.org/10.1002/num.20293**

Herrera-Revilla, I., Contreras, I., & S. Herrera, G. (2020). The divide-and-conquer framework: a suitable setting for domain decomposition methods of the future. *Geofísica Internacional*, *59*(1), 27–37. doi: **https://doi.org/10.22201/igeof.00167169p.2020.59.1.2078**

Klawonn, A., Lanser, M., & Rheinbach, O. (2015). Toward extremely scalable nonlinear domain decomposition methods for elliptic partial differential equations. *SIAM Journal on scientific computing*, *37*(6), C667–C696. doi: **https://doi.org/10.1137/140997907**

Kleinberg, Jon., & Tardos, E. (2006). *Algorithm design*. Pearson/Addison-Wesley.

Mandel, J. (1993). Balancing domain decomposition. *Communications in numerical methods in engineering*, *9*(3), 233–241. doi: **https://doi.org/10.1002/cnm.1640090307**

Mandel, J., & Dohrmann, C. R. (2003). Convergence of a balancing domain decomposition by constraints and energy minimization. *Numerical linear algebra with applications*, *10*(7), 639–659. doi: **https://doi.org/10.1002/nla.341**

Mandel, J., Dohrmann, C. R., & Tezaur, R. (2005). An algebraic theory for primal and dual substructuring methods by constraints. *Applied numerical mathematics*, *54*(2), 167–193. doi: **https://doi.org/10.1016/j.apnum.2004.09.022**

Mathew, T. P. A. (2008). *Domain decomposition methods for the numerical solution of partial differential equations* (Vol. 61). Springer Berlin Heidelberg. doi: **https://doi.org/10.1007/978-3-540-77209-5**

Proceedings of Domain Decomposition Meetings. (n.d.). **https://www.ddm.org/Proceedings.php**

Quarteroni, Alfio., & Valli, A. (1999). *Domain decomposition methods*

for partial differential equations. Clarendon Press.

Rubio, E. (2008). *Métodos de elementos finitos con funciones óptimas*. [Tesis de Doctorado]. Universidad Nacional Autónoma de México.

Saad, Y. (2003). *Iterative methods for sparse linear systems*. Society for industrial and applied mathematics. doi: **https://doi.org/10.1137/1.9780898718003**

SciPy Community. (2008). *scipy.sparse.linalg.splu (Version 1.14.0)*.

Smith, B. F., Bjørstad, P. E., & Gropp, William. (2004). *Domain decomposition : parallel multilevel methods for elliptic partial differential equations*. Cambridge University Press.

Toselli, A., & Widlund, O. B. (2005). *Domain decomposition methods—Algorithms and theory* (Vol. 34). Springer Berlin Heidelberg. doi: **https://doi.org/10.1007/b137868**

Trefethen, L. N. ., & Bau, David. (1997). *Numerical linear algebra*. Society for industrial and applied mathematics.

Van Der Vorst, H. A. (1992). Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, *13*(2), 631–644. doi: **https://doi.org/10.1137/0913035**

Wilkinson, Barry., & Allen, C. Michael. (2005). *Parallel programming: techniques and applications using networked workstations and parallel computers*. Pearson/Prentice Hall.