UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSTGRADO EN CIENCIAS DE LA TIERRA
INSTITUTO DE GEOFÍSICA

“Métodos de Descomposición de Dominio en el Espacio de Vectores Derivados y su Implementación Computacional en Paralelo”

T E S I S

QUE PARA OPTAR POR EL GRADO DE
DOCTOR EN CIENCIAS DE LA TIERRA
(MODELACIÓN DE SISTEMAS TERRESTRES)

P R E S E N T A:

M.C. ANTONIO CARRILLO LEDESMA

DIRECTOR DE TESIS:
DR. ISMAEL HERRERA REVILLA
INSTITUTO DE GEOFÍSICA

México D.F. Marzo del 2013

Dedico el presente trabajo con todo cariño a:

- Mi madre *Alfonsina* por haber hecho posible esto, después de pensar que yo era incorregible.
- Mi esposa *Josefina* por todo su apoyo y tiempo cedido para realizar la presente.
- Mi hijo *José Antonio* por mostrarme lo que realmente es la vida y ceder tanto de su tiempo de juegos conmigo para poder materializar este proyecto.
- Toda mi *familia y amigos* presentes y a los ya ausentes por mostrarme el camino en vida y así poder lograr este trabajo.

Agradecimientos:

Quiero agradecer a todas las personas que hicieron posible que el presente trabajo llegará a su fin:

- Al director de tesis:
  Ismael Herrera Revilla
  por todo el apoyo, tiempo y
  sus invaluables enseñanzas
- A los sinodales:
  Francisco Sánchez Sesma
  Martín Díaz Viera
  Fabián García Nocetti
  Luis Miguel De La Cruz Salas
  por sus enseñanzas y comentarios
- A los investigadores:
  Alejandra Arciniega Ceballos
  Víctor Cruz Atienza
  por todo el apoyo proporcionado

Índice

1 Introducción

Los sistemas continuos —sistemas físicos macroscópicos (véase [1])—, tales como los yacimientos petroleros, la atmósfera, los campos electromagnéticos, los océanos, el aparato circulatorio de los seres humanos, la corteza terrestre y muchos otros sistemas de interés en Ciencia y en Ingeniería, al modelarse, estos contienen un gran número de grados de libertad[1].

Los modelos matemáticos de los sistemas continuos (véase [37] y [10]) son sistemas de ecuaciones diferenciales, las cuales son parciales —con valores iniciales y condiciones de frontera— para casi todos los sistemas de mayor interés en la Ciencia y la Ingeniería. Salvo por los problemas más sencillos, no es posible obtener por métodos analíticos las soluciones de tales ecuaciones, que son las que permiten predecir el comportamiento de los sistemas continuos y realizar las simulaciones requeridas.

La capacidad para formular los modelos matemáticos de sistemas continuos complicados y de gran diversidad, es sin duda una contribución fundamental para el avance de la Ciencia y sus aplicaciones, tal contribución quedaría incompleta y, debido a ello, sería poco fecunda, si no se hubiera desarrollado simultáneamente su complemento esencial: los métodos numéricos y la computación electrónica.

Sin embargo, la solución numérica y las simulaciones computacionales de problemas concomitantes en Ciencias e Ingenierías han llevado al límite nuestra actual capacidad de predicción, por la gran cantidad de grados de libertad que necesitan nuestros modelos para tratar de representar a la realidad.

Con el desarrollo de nuevas herramientas numéricas y computacionales, la diversidad y complejidad de problemas que pueden ser tratados de forma satisfactoria y eficiente es impresionante. Pero hay que destacar, que todavía hay una gama de problemas que hasta la fecha no es posible resolver satisfactoriamente o con la precisión deseada —como la predicción climática a largo plazo o simulación de recuperación mejorada en yacimientos petroleros, entre otros—.

1.1 Antecedentes

La solución numérica de ecuaciones diferenciales parciales por los esquemas tradicionales —tipo Diferencias Finitas, Volumen Finito y Elemento Finito— reducen el problema a la generación y solución de un —cada vez más grande— sistema algebraico de ecuaciones (véase [3]). La factorización directa de sistemas de gran escala $O(10^6)$ con toda su eficacia, no es, en general una opción viable, y el uso de métodos iterativos básicos —tales como el método de gradiente conjugado o residual mínimo generalizado— resultan en una convergencia bastante lenta con respecto a otras formas de discretización como son los métodos de descomposición de dominio (véase [6] y [8]).

El desarrollo de métodos numéricos para sistemas algebraicos grandes, es central en el desarrollo de códigos eficientes para la solución de problemas en

[1] El número de grados de libertad en un sistema físico se refiere al número mínimo de números reales que es necesario especificar para determinar completamente el estado físico.

Ciencia e Ingeniería, actualmente cuando se necesita implementar una solución computacional, se dispone de bibliotecas optimizadas[2] para la solución de sistemas lineales que pueden correr en ambientes secuenciales y/o paralelos. Estas bibliotecas implementan métodos algebraicos robustos para muchos problemas prácticos, pero sus discretizaciones no pueden ser construidas por sólo técnicas algebraicas simples, tales como aproximaciones a la inversa o factorización incompleta.

En la actualidad, los sistemas computacionales paralelos son ubicuos. En ellos es posible encontrar más de una unidad de procesamiento, conocidas como Núcleo (Core). El número de Cores es creciente conforme avanza la tecnología, esto tiene una gran importancia en el desarrollo eficiente de algoritmos que resuelvan sistemas algebraicos en implementaciones paralelas. Actualmente la gran mayoría de los algoritmos desarrollados son algoritmos secuenciales y su implantación en equipos paralelos no es óptima, pero es una práctica común usar diversas técnicas de seudoparalelización —a veces mediante la distribución de una gran matriz en la memoria de los múltiples Cores y otras mediante el uso de directivas de compilación—, pero la eficiencia resultante es pobre y no escalable a equipos masivamente paralelos por la gran cantidad de comunicación involucrada en la solución.

Para hacer eficiente la solución de sistemas de ecuaciones diferenciales parciales, se introdujeron los métodos de descomposición de dominio que toman en cuenta la ecuación diferencial parcial y su discretización, permitiendo una alta eficiencia computacional en diversas arquitecturas paralelas (véase [6] y [8]). La idea básica detrás de los métodos de descomposición de dominio es que en lugar de resolver un enorme problema sobre un dominio, puede ser conveniente —o necesario— resolver múltiples problemas de tamaño menor sobre un solo subdominio un cierto número de veces. Mucho del trabajo en la descomposición de dominio se relaciona con la selección de subproblemas que aseguren que la razón de convergencia del nuevo método iterativo sea rápida. En otras palabras, los métodos de descomposición de dominio proveen precondicionadores a priori que puedan acelerarse por métodos en el espacio de Krylov (véase [19]).

1.2 Métodos de Descomposición de Dominio

La descomposición de dominio generalmente se refiere a la separación de una ecuación diferencial parcial o una aproximación de ella dentro de problemas acoplados sobre subdominios pequeños formando una partición del dominio original. Esta descomposición puede hacerse a nivel continuo, donde diferentes modelos físicos, pueden ser usados en diferentes regiones, o a nivel discreto, donde puede ser conveniente el empleo de diferentes métodos de aproximación en diferentes regiones, o en la solución del sistema algebraico asociado a la aproximación de la ecuación diferencial parcial —estos tres aspectos están íntimamente interconectados en la práctica (véase [19])—.

[2] Como pueden ser las bibliotecas ATLAS —http://math-atlas.sourceforge.net— y HYPRE —http://acts.nersc.gov/hypre/— entre muchas otras.

Los método de descomposición de dominio —Domain Decomposition Methods (DDM)— se basan en la suposición de que dado un dominio $\Omega \subset \mathbb{R}^n$, se puede particionar en E subdominios $\Omega_i, i = 1, 2, ..., E$; tales que $\Omega = \left(\bigcup_{i=1}^{E} \Omega_i \right)$, entre los cuales puede o no existir traslape (véase [3] y [19]). Entonces, el problema es reformulado en términos de cada subdominio —mediante el uso de algún método de discretización— obteniendo una familia de subproblemas de tamaño reducido independientes entre sí, y que están acoplados a través de la solución en la interfase —que es desconocida— de los subdominios.

De esta manera, se puede clasificar de forma burda a los métodos de descomposición de dominio (véase [6]), como aquellos en que: existe traslape entre los subdominios y en los que no existe traslape. A la primera clase pertenece el método de Schwarz —en el cual el tamaño del traslape es importante en la convergencia del método— y a los de la segunda clase pertenecen los métodos del tipo subestructuración —en el cual los subdominios sólo tienen en común a los nodos de la interfase o frontera interior—.

Desde hace ya algún tiempo, la comunidad internacional[3] inicio el estudio intensivo de los métodos de descomposición de dominio, la atención se ha desplazado (véase [9]) de los métodos con traslape en el dominio (véase [59]) a los métodos sin traslape en el dominio (véase [52], [53], [54], [55] y [56]), ya que estos últimos son más efectivos para una gran variedad de problemas de la Ciencia y la Ingeniería.

Los métodos de descomposición de dominio sin traslape son un paradigma natural usado por la comunidad de modeladores (véase [1]). Los sistemas físicos son descompuestos en dos o más subdominios contiguos basados en consideraciones fenomenológicas o computacionales. Esta descomposición se refleja en la Ingeniería de Software del código correspondiente, además, el uso de la programación orientada a objetos, permite dividir en niveles la semántica de los sistemas complejos, tratando así con las partes, más manejables que el todo, permitiendo una implementación, extensión y mantenimiento sencillo (véase [17]).

Tomando en cuenta que los métodos de descomposición de dominio sin traslape son fácilmente implementados para su uso en computadoras paralelas mediante técnicas de programación orientada a objetos —porqué el algoritmo del método es paralelo–, además, con los continuos avances en cómputo, en particular, en la computación en paralelo mediante equipos de cómputo de alto desempeño y/o Clusters parecen ser el mecanismo más efectivo para incrementar la capacidad y velocidad de resolución de varios tipos de problemas de interés en Ciencias e Ingenierías usando métodos de descomposición de dominio (véase [53], [54], [55], [57] y [58]).

La implementación de los métodos de descomposición de dominio permite

[3] Ello se refleja en las más de 19 conferencias internacionales de Métodos Descomposición de Dominio (véase [18]) y de las cuales se han publicado 14 libros que recopilan los trabajos más relevantes de cada conferencia. Además de varios mini simposios de descomposición de dominio en congresos mundiales como es el World Congress on Computational Mechanics o el Iberian Latin American Congress on Computational Methods in Engineering.

utilizar de forma eficiente, las crecientes capacidades del cómputo en paralelo (véase [18] y [19]) —Grids[4] de decenas de Clusters, cada uno con cientos o miles de procesadores interconectados por red, con un creciente poder de cómputo medible en Peta Flops—, así como el uso de una amplia memoria —ya sea distribuida y/o compartida del orden de Tera Bytes—, permitiendo atacar una gran variedad de problemas que sin estas técnicas es imposible hacerlo de manera flexible y eficiente. Pero hay que notar que existe una amplia gama de problemas que se necesitan resolver, los cuales superan la capacidad de cómputo actual, ya sea por el tiempo requerido para su solución, por el consumo excesivo de memoria o ambos.

Así, los métodos de descomposición de dominio que introducen desde la formulación matemática del problema una separación natural de las tareas a realizar y simplifican considerablemente la transmisión de información entre los subdominios (véase [19]), en conjunción con la programación orientada a objetos y el cómputo en paralelo forman una amalgama poderosa. La cual permite construir aplicaciones que coadyuven en la solución una gran gama de problemas concomitantes en Ciencias e Ingenierías que requieren hacer uso de una gran cantidad de grados de libertad.

Por otro lado, la lista de los métodos de descomposición de dominio y el tipo de problemas que pueden ser atacados por estos, es grande y está en constante evolución (véase [18] y [19]), ya que se trata de encontrar un equilibrio entre la complejidad del método —aunada a la propia complejidad del modelo—, la eficiencia en el consumo de los recursos computacionales y la precisión esperada en la solución encontrada por los diversos métodos y las arquitecturas paralelas en la que se implante.

Dos de los esquemas más comúnmente usados (véase [18], [42], [43], [53], [54], [55], [57], [58] y [30]) en los métodos de descomposición de dominio sin traslapes son Finite Element Tearing and Interconnect Dual-Primal (FETI-DP) y Balancing Domain Decomposition by Constraints (BDDC). Aquí, FETI es sinónimo de Finite Element Tearing and Interconnect de Farhat (véase [20] y [21]); y FETI-DP es la versión Dual-Primal de FETI (véase [22] a [24]). BDD es el método de Balancing Domain Decomposition de Mandel (véase [25] y [26]), mientras que BDDC es BDD con restricciones (véase [27] al [29]). Ambos: FETI-DP y BDDC inician de la ecuación diferencial parcial y en ellos los grados de libertad están asociados con las funciones base usadas.

En el marco de los métodos de descomposición de dominio sin traslape se distinguen dos categorías: Los esquemas duales —es el caso del método Finite Element Tearing and Interconnect (FETI) y sus variantes— los cuales usan multiplicadores de Lagrange; y esquemas primales —es el caso del método Balancing Domain Decomposition (BDD) y sus variantes— que tratan el problema

[4]Bajo el Macroproyecto: Tecnologías para la Universidad de la Información y la Computación de la UNAM, se interconectaron cuatro Cluster heterogéneos —dos en la Facultad de Ciencias, uno en el Instituto de Geofísica y otro en el Instituto de Matemáticas Aplicadas y Sistemas— con redes no dedicadas y en ellos se probo una versión de los códigos, en los cuales se vio que es factible el uso en Grids y si estos cuentan con una red dedicada —de alta velocidad— su eficiencia puede llegar a ser alta.

sin el recurso de los multiplicadores de Lagran

Idealmente, los métodos de descomposición buscan satisfacer lo que llamamos el paradigma-DDM: "el cual construye la solución global por la resolución de problemas locales, exclusivamente". Para logar esto, es esencial desconectar los problemas en los sudominios. En FETI-DP, tal desconexión es lograda por la formulación del método en el espacio producto que esta contenido en el espacio de funciones discontinuas. Sin embargo, FETI-DP usa una formulación indirecta basada en los multiplicadores de Lagrange. BBDC usa en lugar una formulación directa, pero no trabaja en el espacio de funciones discontinuas. Otro hecho difícil a superar por los métodos, es que los algoritmos competitivos necesitan incorporar restricciones que eviten la total desconexión de los subdominios.

En este trabajo se ha desarrollado una metodología integradora de dos de los métodos ampliamente usados —FETI-DP y BDDC— y se generan otros nuevos (véase [36], [39]). A este esquema, lo hemos llamado el espacio de vectores derivados —Derived Vectors Space (DVS)—, el cual es un esquema primal similar a la formulación BDD, donde una significativa diferencia entre el esquema BDD y DVS es que en este último, el problema es transformado en otro, definido en el espacio de vectores derivados, el cual es un espacio producto, conteniendo funciones discontinuas donde todo el trabajo del método es realizado, el cual proporciona un marco unificado para los métodos de descomposición de dominio sin traslape y es usado para formular y discutir en general y de forma sistemática la teoría de DDM para problemas simétricos, no simétricos e indefinidos.

En la formulación BDD por otro lado, el espacio original de funciones continuas nunca es abandonado completamente y constantemente se regresa a los grados de libertad asociados con el espacio de funciones continuas pertenecientes a las subestructuras, el cual en su formulación juega el rol del espacio producto.

1.3 El Método de Descomposición de Dominio en el Espacio de Vectores Derivados

En el presente trabajo se da una perspectiva general de algunas de las más importantes formulaciones algebraicas de métodos de descomposición de dominio sin traslape. Dos de los esquemas más comúnmente usados son (véase [18], [42], [43], [53], [54], [55], [57] y [58]): BDDC y FETI-DP. Los cuales fueron puestos en el marco primal —véase secciones (5.1) y (5.2)— del espacio de vectores derivados —Derived Vectors Space (DVS)—. El cual permite una efectiva y sintética presentación de ambos métodos (véase [36] y [38]): Formulaciones primal y dual.

Esto simplifica los algoritmos, los que se sintetizan en un breve conjunto de formulaciones matriciales muy generales que son aplicables a matrices simétricas, no simétricas e indefinidas cuando ellas provienen de la discretización de ecuaciones parciales o sistemas de tales ecuaciones.

El espacio de vectores derivados constituye un espacio de Hilbert con respecto al adecuado producto interior —el producto interior Euclidiano— y mediante la utilización de la formulación DVS, se saca provecho de la estructura del espacio de Hilbert, obteniendo de esta manera una gran simplicidad para el algoritmo

en el espacio de funciones definidas por tramos y es usado para establecer una clara correspondencia entre los problemas a nivel continuo y aquellos obtenidos después de la discretización.

En particular, usando el esquema DVS, se deriva de forma simple y explícita un conjunto de ocho fórmulas matriciales aplicables a matrices simétricas, no simétricas e indefinidas generadas a partir de la discretización de los sistemas de ecuaciones diferenciales parciales para un desarrollo simplificado del código computacional de modelos gobernados por una sola ecuación diferencial parcial o sistemas de estas ecuaciones, teniendo un amplio campo de aplicación a problemas prácticos. Estas formulaciones matriciales explícitas son usadas directamente en el desarrollo de código computacional.

De las ocho fórmulas matriciales explícitas, cuatro son no precondicionadas y cuatro precondicionadas. De ellas, las que tienen interés práctico son por supuesto las formulaciones precondicionadas, pero se incluyen las no precondicionadas, porque de ellas se deriva el entendimiento teórico de las precondicionadas, además de ser el camino más directo para el desarrollo del código computacional, al permitir empezar con una formulación simple y posteriormente agregar el precondicionador para obtener la formulación precondicionada requerida.

De las cuatro fórmulas matriciales precondicionadas, dos corresponden a los algoritmos FETI-DP y BDDC, de las otras dos no se tiene contraparte reportada en la literatura de métodos de descomposición de dominio, aunque la efectividad de su desempeño es del mismo orden de los métodos BDDC o FETI-DP.

Nótese que, todo el desarrollo de la metodología ha sido hecho en el espacio vectorial sujeto a restricciones y por lo tanto, todos los algoritmos aquí presentados son algoritmos sujetos a restricciones.

Idealmente, los métodos DDM intentan producir algoritmos tal que "la solución global es obtenida por la resolución de problemas locales definidos de manera separada en cada subdominio de la malla gruesa de la descomposición"; en este trabajo tal condición será referida como el "paradigma DDM". Cuando el paradigma DDM es totalmente satisfecho la paralelización de los métodos DVS puede ser lograda de forma eficiente al asignar cada subdominio a un procesador diferente.

Algunas características importantes de la metodología desarrollada son:

1. Las formulaciones Dual y Primal son de dos de los métodos comúnmente usados —BDDC y FETI-DP— han sido derivados de una manera unificada.

2. El esquema DVS incluye formulaciones algebraicas para matrices simétricas, no simétricas e indefinidas —i.e. no positivas y no negativas definidas— . Además se detallan las condiciones que tales matrices deben de satisfacer para que los algoritmos generales sean aplicables.

3. El esquema DVS permite aplicar técnicas de descomposición de dominio directamente a la matriz que es obtenida después de que la ecuación diferencial —o sistema de tales ecuaciones— ha sido discretizada. La

aplicación de tales procedimientos no requiere del conocimiento acerca de la ecuación diferencial que originó la matriz.

4. El esquema DVS puede ser aplicado independientemente del procedimiento de discretización usado para obtenerlo; ya que la matriz que es obtenida inmediatamente después de la discretización no esta definida en el espacio de vectores derivados. La teoría provee una fórmula para derivar otra matriz en términos del problema formulado en el espacio de vectores derivados (véase [34]).

5. Como es común en los métodos de descomposición de dominio, la matriz global nunca se construye; a este respecto, una manera simple de definir el espacio vectorial derivado es presentada en este trabajo.

El esquema DVS así obtenido, es innovador en varios aspectos y capaz de englobar a dos de los métodos de descomposición de dominio más usados. Una significativa innovación es que los algoritmos desarrollados en este esquema son igualmente aplicables a matrices simétricas y no simétricas (véase [35]). En el presente trabajo se muestra que la solución de matrices no simétricas presenta eficiencia comparable a las simétricas.

1.4 Objetivos de la Tesis

Los objetivos del presente trabajo son agrupados en objetivos generales, objetivos particulares y objetivos de la implementación, los cuales se detallan a continuación.

Objetivos Generales Los objetivos generales del presente trabajo son:

- Presentar el esquema del método de descomposición de dominio en el espacio de vectores derivados (véase [36], [39] y [38]), el cual permite aplicar técnicas de descomposición de dominio directamente al sistema de matrices que son obtenidas después de que la ecuación diferencial o sistema de tales ecuaciones han sido discretizadas. La aplicación de tales procedimientos no requiere del conocimiento acerca de la ecuación diferencial que originó las matrices.

- Mostrar como la teoría provee una fórmula para derivar la matriz en términos del problema formulado en el espacio de vectores derivados, ya que la matriz que es obtenida inmediatamente después de la discretización no está definida en el espacio de vectores derivados. Como es común en los métodos de descomposición de dominio, tal matriz nunca se construye. A este respecto, una manera simple de definir el espacio de vectores derivados es presentada en este trabajo.

- Mostrar como el esquema DVS es igualmente aplicable a matrices simétricas, no simétricas e indefinidas.

- Mostrar la eficiencia computacional de los algoritmos desarrollados en problemas que generan matrices simétricas y no simétricas, además de mostrar que la implementación en paralelo es escalable.

Objetivos Particulares Los objetivos particulares de este trabajo son:

- Mostrar los ocho algoritmos iterativos básicos de descomposición de dominio sin traslape que se han obtenido, de los cuales cuatro son formulaciones primales y los otros cuatro son formulaciones duales.
- Mostrar como dos de las fórmulas precondicionadas corresponden a los algoritmos FETI-DP y BDDC.
- Mostrar en el caso de matrices simétricas y no simétricas (véase [34] y [35]), en particular para problemas de Advección-Difusión (véase [29] y [50]) que la eficiencia numérica de los algoritmos precondicionados están en el mismo orden que los reportados en la literatura.

Objetivos de la Implementación La implementación de los códigos de los métodos de descomposición de dominio en el espacio de vectores derivados en su forma secuencial y paralela tiene como objetivo primario el resolver un grupo de ecuaciones por todos los métodos desarrollados que satisfagan:

- El código sea independiente de la geometría del dominio.
- El código sea independiente de la dimensión del problema.
- El código soporte ecuaciones escalares y vectoriales.
- El código utilice diferentes métodos de solución del sistema lineal asociado al complemento de Schur local.
- El Algoritmo global sea débilmente acoplado a los subdominios.
- El desarrollo del código sea orientado a objetos para simplificar la implementación y permitir tener un solo código para la parte secuencial y paralela con un mínimo de cambios.
- La Implementación sea eficiente y escalable en paralelo.

1.5 Infraestructura Computacional Usada

El modelo computacional generado, está contenido en un programa de cómputo bajo el paradigma de orientación a objetos, programado en el lenguaje C++ en su forma secuencial y en su forma paralela en C++ y la interfaz de paso de mensajes (MPI) bajo el esquema Maestro-Esclavo.

Hay que notar que, el paradigma de programación orientada a objetos sacrifica algo de eficiencia computacional por requerir mayor manejo de recursos computacionales al momento de la ejecución. Pero en contraste, permite mayor

flexibilidad a la hora de adaptar los códigos a nuevas especificaciones. Adicionalmente, disminuye notoriamente el tiempo invertido en el mantenimiento y búsqueda de errores dentro del código. Esto tiene especial interés cuando se piensa en la cantidad de meses invertidos en la programación comparada con los segundos consumidos en la ejecución.

Para desarrollar estos códigos, se realizó una jerarquía de clases para cada uno de los distintos componentes del sistema de descomposición de dominio en base a clases abstractas, las cuales reducen la complejidad del esquema DVS, permitiendo usarlo tanto en forma secuencial como paralela redefiniendo sólo algunos comportamientos.

La programación, depuración y puesta a punto de los códigos fue hecha en el siguiente equipo:

- Notebook Intel dual Core a 1.7 GHtz con 2 GB de RAM en Linux Debian, haciendo uso del compilador C++ GNU y MPICH para el paso de mensajes.
- PC Intel Quad Core a 2.4 GHtz con 3 GB de RAM en Linux Debian, haciendo uso del compilador C++ GNU y MPICH para el paso de mensajes.

Las pruebas de rendimiento de los distintos programas se realizaron en equipos multiCore y Clusters a los que se tuvo acceso y que están montados en la Universidad Nacional Autónoma de México, en las pruebas de análisis de rendimiento se usó el siguiente equipo:

- Cluster homogéneo Kanbalam de 1024 Cores AMD Opteron a 2.6 GHtz de 64 bits, cada 4 Cores con 8 GB de RAM interconectados con un switch de 10 Gbps ubicado en el Departamento de Supercómputo de la D.G.C.T.I.C de la UNAM.
- Cluster homogéneo Olintlali de 108 Cores emulando 216 hilos, Intel Xeon a 2.67 GHtz, 9 nodos con 6 Cores y 8 hilos de ejecución con 48 GB RAM interconectados con GIGE 10/100/1000 Gb/s, a cargo del Dr. Ismael Herrera Revilla del Departamento de Recursos Naturales del Instituto de Geofísica de la UNAM.
- Cluster homogéneo Pohualli de 104 Cores Intel Xeon a 2.33 GHtz de 64 bits, cada 8 Cores cuentan con 32 GB de RAM interconectados con un switch de 1 Gbps, a cargo del Dr. Víctor Cruz Atienza del Departamento de Sismología del Instituto de Geofísica de la UNAM.
- Cluster homogéneo IO de 22 Cores Intel Xeon a 2.8 GHtz de 32 bits, cada 2 Cores con 1 GB de RAM interconectados con un switch de 100 Mbps, a cargo de la Dra. Alejandra Arciniega Ceballos del Departamento de Vulcanología del Instituto de Geofísica de la UNAM.
- PC de alto rendimiento Antipolis de 8 Cores Intel Xeon a 2.33 GHtz de 64 bits y 32 GB de RAM, a cargo del Dr. Víctor Cruz Atienza del Departamento de Sismología del Instituto de Geofísica de la UNAM.

1.6 Organización de la Tesis

Para poder cumplir con los objetivos planteados del presente trabajo, se inicia describiendo en el capítulo dos, las formulaciones Dirichlet-Dirichlet y Neumann-Neumann a nivel continuo que son la base de las formulaciones en el espacio de vectores derivados; en el capítulo tres se desarrolla el marco teórico del espacio de vectores derivados; para que conjuntando el material del capítulo dos y tres, en el capítulo cuatro se deriven las ocho formulaciones Dirichlet-Dirichlet y Neumann-Neumann en el marco del espacio de vectores derivados, de los cuales cuatro son formulaciones primales y cuatro son formulaciones duales donde dos corresponden a los algoritmos Finite Element Tearing and Interconnect Dual-Primal (FETI-DP) y Balancing Domain Decomposition with Constraints (BDDC).

En el capítulo cinco, se muestra como los esquemas FETI-DP y BDDC son puestos en términos del esquema de descomposición de dominio en el espacio de vectores derivados desarrollado, además se muestran sus principales semejanzas y diferencias.

En el capítulo seis, se muestran los detalles de la formulación numérica del esquema de descomposición de dominio en el espacio de vectores derivados, en particular se deriva la forma de construir las matrices involucradas en el método y los detalles de la implementación numérica, así como la implementación de los operadores virtuales involucrados en el esquema.

En el capítulo siete, se muestra la implementación computacional en el esquema Maestro-Esclavo como una forma de codificación orientada a objetos de los métodos de descomposición de dominio en el espacio de vectores derivados tanto para generar el código secuencial como el paralelo en el lenguaje de programación C++ y algunas consideraciones sobre su desempeño computacional.

En el capítulo ocho, se realiza el análisis de rendimiento para mostrar la eficiencia de los códigos desarrollados, por una parte para manipular sistemas simétricos y no simétricos; y por otra ver su escalamiento al usar cientos de Cores en la ejecución de algunas pruebas en equipos paralelos como es el Cluster Kanbalam de la UNAM.

Por último, en el capítulo nueve se dan las conclusiones de los logros alcanzados en el presente trabajo y se esboza lo que se considera pueden ser sus perspectivas.

En el apéndice A, se muestran varias consideraciones sobre la implementación de los métodos de descomposición de dominio en el espacio de vectores derivados, en especial sobre el caso en que la formulación se basa en la creación de las matrices locales a partir de algún método de discretización local por subdominio como pueden ser los métodos de Elemento Finito, Diferencias Finitas o Volumen Finito, así como el manejo de ecuaciones vectoriales o escalares por el sistema desarrollado. Para terminar este apéndice se revisa con detalle el método de descomposición de dominio de subestructuración y su implementación computacional, ya que este es la base de todos y cada uno de los métodos de descomposición de dominio tratados en este trabajo.

En el apéndice B, se muestra como hacer una eficiente implementación para la solución de grandes sistemas de ecuaciones lineales por medio de los méto-

dos directos e iterativos, así como las estructuras óptimas de las matrices al codificarse en el lenguaje de programación C++ tanto para su implementación secuencial como en paralelo.

1.7 Agradecimientos

El presente trabajo es posible gracias al apoyo de múltiples personas que me acompañaron a lo largo del desarrollo del mismo, sin el cual no podría haber concluido este objetivo en mi vida.

- Primeramente, quiero agradecer las incontables horas de clases, asesoría y seminarios que con tanto gusto y dedicación me proporcionó mi tutor, el Dr. Ismael Herrera Revilla, a lo largo de toda mi estancia en la que realice estudios de Maestría y Doctorado en el Instituto de Geofísica de la UNAM.

- A mis tres cotutores el Dr. Arón Jazcilevich Diamant, Dr. Martín Díaz Viera y el Dr. Robert Yates Smith que me acompañaron a lo largo de mis estudios de maestría y doctorado en el Instituto de Geofísica y me ayudaron a definir, hacer y ampliar el presente trabajo. En especial al Dr. Yates, por las cientos de horas dedicadas en resolver mis dudas y ayudarme con ideas innovadoras sobre el arduo proceso de análisis, diseño e implementación de los códigos computacionales tanto en su forma secuencial como paralela, en sus múltiples versiones desarrolladas a lo largo de estos años.

- A la Dra. Alejandra Arciniega Ceballos y al Dr. Víctor Cruz Atienza por todo el apoyo prestado a lo largo de tantos años, tanto en asesorías como con el acceso a los equipos de alto rendimiento y Clusters que tienen a su cargo; los cuales fueron invaluables para poder concluir el presente trabajo.

- Al encargado del Cluster Kanbalam de la D.G.C.T.I.C.-UNAM, M.C. José Luís Gordillo por todo el apoyo proporcionado para lograr concluir las corridas en dicho equipo y por resolver los problemas técnicos que se presentaron con tanta rapidez y expertez.

- A los compañeros del Grupo de Modelación Matemática y Computacional del Instituto de Geofísica de la UNAM —investigadores y compañeros de estudio—, y muy en especial a los alumnos de los cursos que impartí a lo largo de mi estancia en el Instituto en especial a Alberto Rosas Medina, a los cuales les agradezco infinitamente el haberme permitido ser su colega, pues fue con ellos y a través de nuestra interacción que se afianzaron mis conocimientos del área de Modelación Matemática y Computacional en sus múltiples facetas.

2 Formulaciones Dirichlet-Dirichlet y Neumann-Neumann a Nivel Continuo

Para poner la metodología desarrollada en una adecuada perspectiva, se inicia por revisar algunos conceptos elementales sobre las formulaciones mencionadas en el título de esta sección. También se toman algunas herramientas presentadas del trabajo "Theory of differential equations in discontinuous picewise-defined functions" (véase [31]).

El problema que se considera aquí[5] es: "Encontrar $u \in H^2(\Omega)$, tal que

$$\begin{cases} \mathcal{L}u = f_\Omega, \text{ en } \Omega \\ u = 0, \text{ sobre } \partial\Omega \end{cases} \quad „ \tag{2.1}$$

así, bajo las adecuadas condiciones (véase [6]), la existencia y la unicidad de la solución de este problema está garantizada. Este problema puede también formularse en espacio de funciones definidas por tramos (véase [31]).

Sin pérdida de generalidad, sea el dominio Ω descompuesto en dos subdominios Ω_1 y Ω_2, como se muestra en la figura:

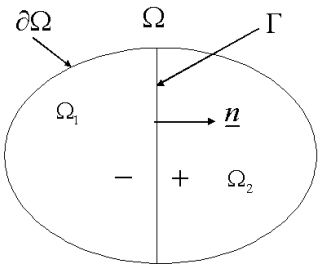

Figura 1: Partición del dominio Ω en dos subdominios Ω_1 y Ω_2.

Supóngase que el operador diferencial $\mathcal{L}$ es de segundo orden y se considera el espacio $H^2(\Omega_1) \oplus H^2(\Omega_2)$ de funciones discontinuas definidas por tramos (véase [31]). Una función en tal espacio es definida independientemente en cada uno de los dos subdominios y sus restricciones a Ω_1 y Ω_2 pertenecen a $H^2(\Omega_1)$ y $H^2(\Omega_2)$ respectivamente. Generalmente la discontinuidad a través de la interfase Γ tiene un salto no cero de la función misma y de sus derivadas normales. La notación para el 'salto' y el 'promedio' a través de Γ esta dado por

$$[\![u]\!] \equiv u_+ - u_- \qquad \text{y} \qquad \dot{\widehat{u}} = \frac{1}{2}(u_+ + u_-), \text{ sobre } \Gamma \tag{2.2}$$

respectivamente.

[5] La notación que se usa es la estándar para los espacios de Sobolev (véase [2]).

Nótese que, el espacio $H^2(\Omega)$ es un subespacio de $H^2(\Omega_1) \oplus H^2(\Omega_2)$. En efecto, sea $u \in H^2(\Omega_1) \oplus H^2(\Omega_2)$, entonces $u \in H^2(\Omega)$ si y sólo si

$$[\![u]\!] = \left[\!\left[\frac{\partial u}{\partial n}\right]\!\right] = 0, \text{ sobre } \Gamma. \tag{2.3}$$

Por lo tanto, una formulación del problema dado en la Ec.(2.1) es: "Buscar a $u \in H^2(\Omega_1) \oplus H^2(\Omega_2)$, tal que

$$\begin{cases} \mathcal{L}u = f_\Omega, \text{ en } \Omega \\ [\![u]\!] = 0 \text{ y } \left[\!\left[\frac{\partial u}{\partial n}\right]\!\right] = 0, \text{ sobre } \Gamma \quad \text{". } \\ u = 0, \text{ sobre } \partial\Omega \end{cases} \tag{2.4}$$

Se debe observar que, cuando el valor de la solución u es conocida sobre Γ, entonces u puede ser obtenida en cualquier lugar en Ω por la resolución de dos problemas Dirichlet con valor en la frontera, uno en cada uno de los subdominios Ω_1 y Ω_2 —el título de Dirichlet-Dirichlet para el procedimiento es por el hecho de resolver este tipo de problemas—.

De manera similar, cuando los valores de la derivada normal $\frac{\partial u}{\partial n}$ son conocidos sobre Γ, entonces u puede ser obtenida en cualquier lugar en Ω por la resolución de dos problemas Neumann con valores en la frontera, uno para cada uno de los subdominios Ω_1 y Ω_2 —el título de Neumann-Neumann para el procedimiento es por el hecho de resolver este tipo de problemas—.

Estas observaciones son las bases de los dos enfoques de los métodos de descomposición de dominio que se consideran en esta sección: Los métodos Dirichlet-Dirichlet y Neumann-Neumann.

2.1 El Problema no Precondicionado Dirichlet-Dirichlet

Para el problema no precondicionado Dirichlet-Dirichlet, se toma u_Γ como la restricción a Γ de la solución u de la Ec.(2.4), entonces u es la única solución de los siguientes dos problemas Dirichlet

$$\begin{cases} \mathcal{L}u = f_\Omega, \text{ en } \Omega_\alpha, \alpha = 1, 2 \\ u = u_\Gamma, \text{ sobre } \Gamma \\ u = 0, \text{ sobre } \partial\Omega \end{cases} . \tag{2.5}$$

El enfoque Dirichlet-Dirichlet al método de descomposición de dominio consiste en buscar para la función u_Γ, esencialmente una elección de sucesiones de funciones de prueba: $u_\Gamma^0, u_\Gamma^1, ..., u_\Gamma^n, ...$, hasta que se encuentre la función buscada.

A este respecto, una primera pregunta es: ¿cómo se reconoce cuando la función de prueba es satisfactoria?. Se sabe que cuando u_Γ es la restricción a Γ de la solución del problema, la solución que es obtenida por la resolución de los dos problemas con valores en la frontera de la Ec.(2.5) satisfacen las condiciones de salto indicadas en la Ec.(2.4). Ahora, en el caso del enfoque Dirichlet-Dirichlet, el salto de la función se nulifica necesariamente, ya que

$$[\![u]\!] = u_+ - u_- = u_\Gamma - u_\Gamma = 0 \tag{2.6}$$

sin embargo, por lo general la condición

$$\left[\!\left[\frac{\partial u}{\partial n}\right]\!\right] = 0 \tag{2.7}$$

no es satisfecha. La selección de la función de prueba u_Γ será satisfactoria, si y sólo si, la Ec.(2.7) se satisface.

Si se escribe la solución del problema u de la Ec.(2.4) como

$$u = u_p + v \tag{2.8}$$

donde u_p satisface

$$\begin{cases} \mathcal{L}u_p = f_\Omega, \text{ en } \Omega_\alpha, \alpha = 1,2 \\ u_p = 0, \text{ sobre } \Gamma \\ u_p = 0, \text{ sobre } \partial\Omega \end{cases} \tag{2.9}$$

y por lo tanto v satisface

$$\begin{cases} \mathcal{L}v = 0, \text{ en } \Omega_\alpha, \alpha = 1,2 \\ v = u_\Gamma, \text{ sobre } \Gamma \\ v = 0, \text{ sobre } \partial\Omega \end{cases} \tag{2.10}$$

entonces, la elección de la función de prueba u_Γ será satisfactoria, si y sólo si

$$\left[\!\left[\frac{\partial v}{\partial n}\right]\!\right] = -\left[\!\left[\frac{\partial u_p}{\partial n}\right]\!\right] \tag{2.11}$$

desde este punto de vista, la función $v \in H^2(\Omega_1) \oplus H^2(\Omega_2)$. Así, se busca una función tal que

$$\begin{cases} \mathcal{L}v = 0, \text{ en } \Omega_\alpha, \alpha = 1,2 \\ [\![v]\!] = 0, \left[\!\left[\frac{\partial v}{\partial n}\right]\!\right] = -\left[\!\left[\frac{\partial u_p}{\partial n}\right]\!\right], \text{ sobre } \Gamma \quad . \\ v = 0, \text{ sobre } \partial\Omega \end{cases} \tag{2.12}$$

Esta condición puede expresarse teniendo en mente el operador de Steklov-Poincaré[6] τ, el cual para cualquier función u definida sobre Γ, se genera otra función definida sobre Γ, a saber (véase [3])

$$\tau(u_\Gamma) \equiv \left[\!\left[\frac{\partial v}{\partial n}\right]\!\right], \text{ sobre } \Gamma \tag{2.13}$$

donde v satisface la Ec.(2.10). Entonces la Ec.(2.12) es equivalente a

$$\tau(u_\Gamma) \equiv -\left[\!\left[\frac{\partial u_p}{\partial n}\right]\!\right], \text{ sobre } \Gamma. \tag{2.14}$$

[6] El operador de Steklov-Poincaré generalmente se define como la ecuación para la traza de la solución exacta u sobre Γ (véase [49]).

2.2 El Problema no Precondicionado Neumann-Neumann

Para el caso del problema no precondicionado Neumann-Neumann, se toma a u como la solución de la Ec.(2.4) y sea $q_\Gamma \equiv \frac{\partial u}{\partial n}$ sobre Γ, entonces u es la única solución de los siguientes dos problemas Neumann

$$\begin{cases} \mathcal{L}u = f_\Omega, \text{ en } \Omega_\alpha, \alpha = 1,2 \\ \frac{\partial u}{\partial n} = q_\Gamma \text{ sobre } \Gamma \\ u = 0 \text{ sobre } \partial\Omega \end{cases} . \tag{2.15}$$

El título del enfoque Neumann-Neumann proviene de el hecho que la Ec.(2.15) implica resolver un problema Neumann en el subdominio Ω_1 y otro problema Neumann en Ω_2. Este enfoque consiste en buscar una función q_Γ —independientemente del valor de q_Γ—, ya que, cualquier solución de la Ec.(2.15) satisface

$$\left[\!\left[\frac{\partial u}{\partial n}\right]\!\right] = \left(\frac{\partial u}{\partial n}\right)_+ - \left(\frac{\partial u}{\partial n}\right)_- = q_\Gamma - q_\Gamma = 0 \tag{2.16}$$

sin embargo, por lo general, la condición

$$[\![u]\!] = 0 \tag{2.17}$$

no es satisfecha. La selección de la función de prueba u_Γ será satisfactoria, si y sólo si, la Ec.(2.17) se satisface.

Si se toma nuevamente una solución $u = u_p + v$ como en la Ec.(2.8), donde u_p ahora satisface

$$\begin{cases} \mathcal{L}u_p = f_\Omega, \text{ en } \Omega_\alpha, \alpha = 1,2 \\ \frac{\partial u_p}{\partial n} = 0, \text{ sobre } \Gamma \\ u_p = 0, \text{ sobre } \partial\Omega \end{cases} \tag{2.18}$$

y por lo tanto v satisface

$$\begin{cases} \mathcal{L}v = 0, \text{ en } \Omega_\alpha, \alpha = 1,2 \\ \frac{\partial v}{\partial n} = q_\Gamma, \text{ sobre } \Gamma \\ v = 0, \text{ sobre } \partial\Omega \end{cases} \tag{2.19}$$

entonces, la función $v \in H^2(\Omega_1) \oplus H^2(\Omega_2)$ es caracterizada por

$$\begin{cases} \mathcal{L}v = 0, \text{ en } \Omega_\alpha, \alpha = 1,2 \\ [\![v]\!] = [\![u_p]\!], \left[\!\left[\frac{\partial v}{\partial n}\right]\!\right] = 0, \text{ sobre } \Gamma \\ v = 0, \text{ sobre } \partial\Omega \end{cases} . \tag{2.20}$$

Para la formulación Neumann-Neumann existe una contraparte —el operador μ— al operador de Steklov-Poincaré τ, dada cualquier función q_Γ, definida sobre Γ, se define $\mu(q_\Gamma)$, esta será una función definida sobre Γ dada por

$$\mu(q_\Gamma) \equiv [\![v]\!], \text{ sobre } \Gamma \tag{2.21}$$

aquí, v satisface la Ec.(2.19). Entonces, la Ec.(2.20) es equivalente a

$$\mu(q_\Gamma) \equiv -[\![u_p]\!], \text{ sobre } \Gamma. \tag{2.22}$$

3 Marco Teórico del Espacio de Vectores Derivados

En el marco de los métodos de descomposición de dominio sin traslape se distinguen dos categorías (véase [18], [42], [43], [53], [54] y [58]): Los esquemas duales —como es el caso del método Finite Element Tearing and Interconnect (FETI) y sus variantes— los cuales usan multiplicadores de Lagrange; y los esquemas primales —como es el caso del método Balancing Domain Decomposition (BDD) y sus variantes— que tratan el problema sin el recurso de los multiplicadores de Lagrange.

En este trabajo se ha derivado un sistema unificador (véase [36] y [38]), el esquema del espacio de vectores derivados (DVS), este es un esquema primal similar a la formulación BDD. Donde una significativa diferencia entre nuestro esquema y BDD es que en la formulación DVS el problema es transformado en otro definido en el espacio de vectores derivados, el cual es un espacio producto conteniendo funciones discontinuas, es en este espacio en el cual todo el trabajo del método es realizado.

Por otro lado, en la formulación BDD, el espacio original de funciones continuas nunca es abandonado completamente y constantemente se regresa a los grados de libertad asociados con el espacio de funciones continuas pertenecientes a las subestructuras, el cual en su formulación juega el rol del espacio producto.

Aunque el marco DVS es un esquema primal, las formulaciones duales pueden ser acomodadas en él; esta característica permite unificar en este terreno a ambos esquemas, duales y primales; en particular a BDDC y FETI-DP. También el espacio DVS constituye un espacio de Hilbert con respecto al adecuado producto interior —el producto interior Euclidiano— y mediante la utilización de la formulación DVS, se saca provecho de la estructura del espacio de Hilbert obteniendo de esta manera una gran simplicidad para el algoritmo definido en el espacio de funciones definidas por tramos y es usado para establecer una clara correspondencia entre los problemas a nivel continuo y aquellos obtenidos después de la discretización.

3.1 Discretización del Dominio para los Métodos Duales y Primales

Dos de los enfoques más comúnmente usados (véase [18], [40], [41], [42], [43], [53], [54], [55], [57] y [58]) en los métodos de descomposición de dominio son FETI-DP y BDDC. Ambos, inicializan con la ecuación diferencial parcial y de ella, los grados de libertad son asociados con las funciones de base usadas. BDDC es un método directo, i.e. es un método que no hace uso de Multiplicadores de Lagrange, mientras que FETI-DP trabaja en el espacio producto mediante el uso de los Multiplicadores de Lagrange.

En los métodos FETI-DP y BDDC, el dominio Ω en el cual se define el problema, es subdividido en E subdominios 'sin traslape' Ω_α, $\alpha = 1, 2, ..., E$, es

decir

$$\Omega_\alpha \cap \Omega_\beta = \varnothing \quad \forall \alpha \neq \beta \quad \text{y} \quad \overline{\Omega} = \bigcup_{\alpha=1}^{E} \overline{\Omega}_\alpha \tag{3.1}$$

y al conjunto

$$\Gamma = \bigcup_{\alpha=1}^{E} \Gamma_\alpha, \quad \text{si} \quad \Gamma_\alpha = \partial\Omega_\alpha \backslash \partial\Omega \tag{3.2}$$

se le llama la frontera interior del dominio Ω. La notación $\partial\Omega$ y $\partial\Omega_\alpha$, $\alpha = 1, ..., E$ es tomada de la frontera del dominio Ω y la frontera del subdominio Ω_i respectivamente. Claramente

$$\partial\Omega \subset \bigcup_{\alpha=1}^{E} \partial\Omega_\alpha \quad \text{y} \quad \Omega = \left(\bigcup_{\alpha=1}^{E} \Omega_\alpha \right) \bigcup \Gamma. \tag{3.3}$$

Se denota por H al máximo diámetro $H_\alpha = Diam(\Omega_\alpha)$ de cada Ω_α que satisface $Diam(\Omega_\alpha) \leq H$ para cada $\alpha = 1, 2, ..., E$, además, cada subdominio Ω_α es descompuesto en un mallado fino $\mathcal{T}_h$ de K subdominios mediante una triangulación Ω_j de modo que este sea conforme, se denota por h al máximo diámetro $h_j = Diam(\Omega_j)$ de cada Ω_α que satisface $Diam(\Omega_j) \leq h$ para cada $j = 1, 2, ..., K$ de cada $\alpha = 1, 2, ..., E$.

Para iniciar la discusión, se considera un ejemplo particular, de un conjunto de veinticinco nodos de una descomposición del dominio Ω 'sin traslape', se asume que la numeración de los nodos es de izquierda a derecha y de arriba hacía abajo, véase figura (2)

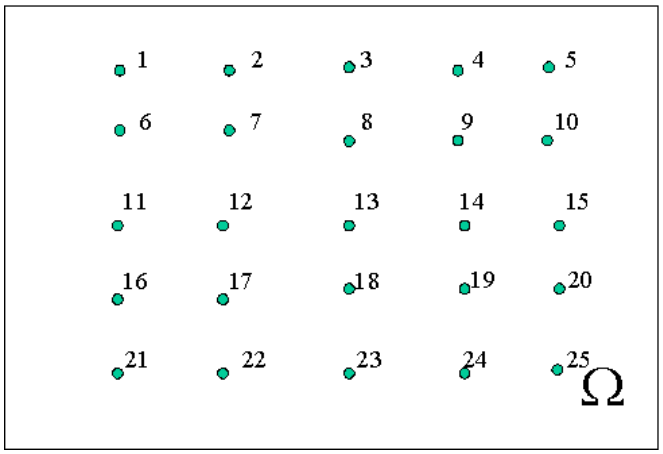

Figura 2: Conjunto de nodos originales

En este caso el conjunto de nodos originales, se particiona usando una malla gruesa de cuatro subdominios Ω_α $\alpha = 1, 2, 3, 4$, véase figura (3).

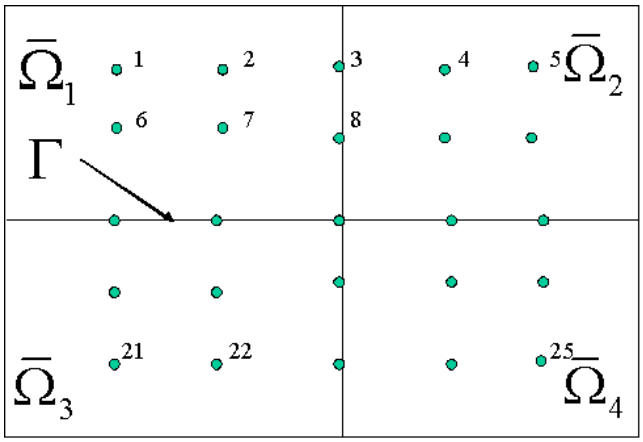

Figura 3: Nodos en una descomposición gruesa de cuatro subdominios

Entonces se tiene un conjunto de nodos y un conjunto de subdominios, los cuales se enumeran usando un conjunto de índices

$$\hat{N} \equiv \{1, 2, \ldots, 25\} \qquad \text{y} \qquad E \equiv \{1, 2, 3, 4\} \tag{3.4}$$

respectivamente. Entonces, el conjunto de 'nodos originales' correspondiente a tal descomposición de dominio 'sin traslape', en realidad tiene un 'traslape', esto es por que la familia de los cuatros subconjuntos

$$\begin{aligned} \hat{N}^1 &= \{1, 2, 3, 6, 7, 8, 11, 12, 13\} \\ \hat{N}^2 &= \{3, 4, 5, 8, 9, 10, 13, 14, 15\} \\ \hat{N}^3 &= \{11, 12, 13, 16, 17, 18, 21, 22, 23\} \\ \hat{N}^4 &= \{13, 14, 15, 18, 19, 20, 23, 24, 25\} \end{aligned} \tag{3.5}$$

no son disjuntos. En efecto, por ejemplo

$$\hat{N}^1 \cap \hat{N}^2 = \{3, 8, 13\}\,. \tag{3.6}$$

Con la idea de obtener una 'verdadera descomposición de dominio sin traslape', se reemplaza el conjunto de 'nodos originales' por otro conjunto, el conjunto de los 'nodos derivados' en los cuales se satisfaga la condición de que sea una verdadera descomposición de dominio sin traslapes.

3.2 Una Verdadera Descomposición de Dominio sin Traslapes

A la luz de lo visto en la sección anterior, es ventajoso trabajar con una descomposición de dominio sin traslape alguno del conjunto de nodos y, para este fin, se reemplaza el conjunto $\hat{N}$ de 'nodos originales' por otro conjunto de 'nodos

derivados', los cuales son denotados por X^α, donde cada nodo en la frontera interior Γ se parte en un número que llamaremos la multiplicidad del nodo y es definida como el número de subdominios que comparten al nodo. Cada nodo derivado es definido por un par de números: (p, α), donde p corresponde a un nodo perteneciente a $\overline{\Omega}_p$, i.e. un 'nodo derivado' es el par de números (p, α) tal que $p \in \hat{N}^\alpha$, véase figura (4).

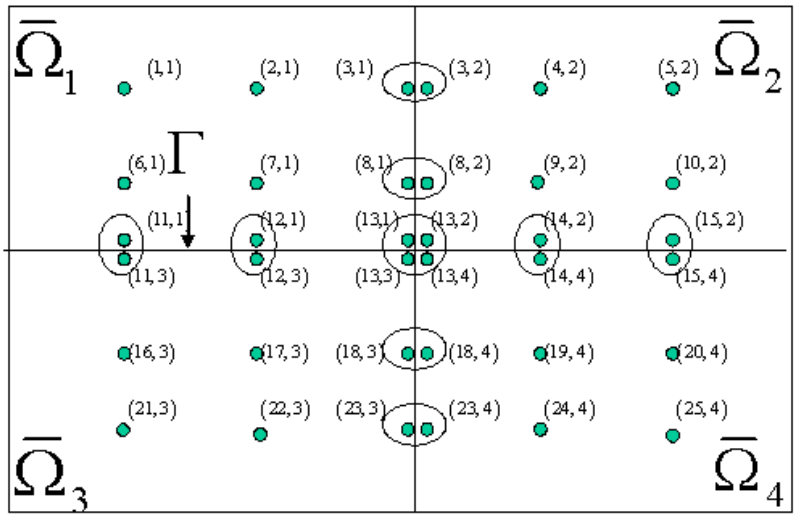

Figura 4: Mitosis de los nodos en la frontera interior

Se denota con X el conjunto total de nodos derivados; nótese que el total de nodos derivados para el ejemplo de la sección anterior es de 36, mientras el número de nodos originales es de 25.

Entonces, se define X^α como el conjunto de nodos derivados que puede ser escrito como

$$X^\alpha = \left\{ (p, \alpha) \mid \alpha \in \hat{E} \quad \text{y} \quad p \in \hat{N}^\alpha \right\}. \tag{3.7}$$

Para el ejemplo referido, tomando α sucesivamente como $1, 2, 3$ y 4, se tiene la familia de cuatro subconjuntos,

$$\left\{ X^1, X^2, X^3, X^4 \right\} \tag{3.8}$$

la cual es una descomposición de dominio sin traslape véase figura (5), donde el conjunto X satisface

$$X = \bigcup_{\alpha=1}^{4} X^\alpha \quad \text{y} \quad X^\alpha \cap X^\beta = \varnothing \quad \text{cuando} \quad \alpha \neq \beta. \tag{3.9}$$

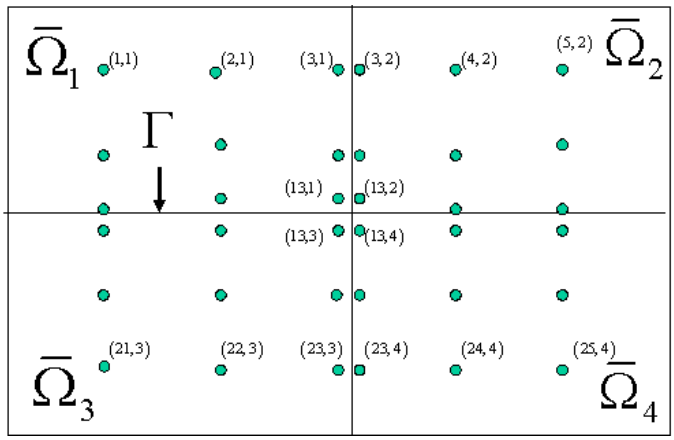

Figura 5: Conjunto de nodos derivados

Por supuesto, la cardinalidad de los subdominio —el número de nodos de cada uno de los subdominios es 36/4— es igual a 9.

Dada la discusión anterior, y para el desarrollo de una teoría general, sea el conjunto de nodos-índice y de subdominio-índice original definidos como

$$\hat{N} = \{1, ..., n\} \quad \text{y} \quad \hat{E} = \{1, ..., E\} \tag{3.10}$$

respectivamente, donde n es el número total de nodos que se obtienen de la versión discretizada de la ecuación diferencial que se quiere resolver —generalmente, los nodos de frontera no son incluidos—.

Se define al conjunto de 'nodos derivados' por el par de números (p, α), tal que $p \in \hat{N}^{\alpha}$. Entonces, el conjunto de todos los nodos derivados, satisface

$$X^{\alpha} \equiv \left\{ (p, \alpha) \mid \alpha \in \hat{E} \quad \text{y} \quad p \in \hat{N}^{\alpha} \right\} \tag{3.11}$$

y para evitar repeticiones, ya que en lo sucesivo se hará uso extensivo de los nodos derivados, la notación (p, α) se reserva para el par tal que $(p, \alpha) \in X$.

Nótese que la cardinalidad de X es siempre mayor que la cardinalidad de $\hat{N}$, excepto en el caso trivial cuando $E = 1$. En lo que sigue los nodos derivados serán referidos, simplemente como nodos.

Por otro lado, dado cualquier nodo original, $p \in \hat{N}$, se define el conjunto $Z(p) \subset X$, como el conjunto de los nodos derivados de p que puede ser escrito como (p, α), para algún $1 \leq \alpha \leq E$. Además, dado cualquier nodo $(p, \alpha) \in X$, su multiplicidad es definida por la cardinalidad del conjunto $Z(p) \subset X$ y es denotada como $m(p)$.

Los nodos derivados son clasificados como 'nodos internos' o de 'frontera interna', dependiendo si su multiplicidad es uno o más grande que uno. Los

subconjuntos $I \subset \overline{\Omega}$ y $\Gamma \subset \overline{\Omega}$ son el conjunto de nodos internos y de frontera interna respectivamente, que satisfacen la siguiente relación

$$X = I \cup \Gamma \quad \text{y} \quad I \cap \Gamma = \varnothing \tag{3.12}$$

nótese que para el ejemplo mostrado en la figura 3, la cardinalidad de Γ es 20.

Para cada $\alpha = 1, 2, \ldots, E$, se define

$$\Gamma_\alpha = \Gamma \cap X^\alpha \tag{3.13}$$

entonces, la familia de subconjuntos $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_E\}$ es una partición de Γ, en el sentido de que

$$\Gamma \equiv \bigcup_{\alpha=1}^{E} \Gamma_\alpha \quad \text{mientras} \quad \Gamma_\alpha \cap \Gamma_\beta = \varnothing \quad \text{cuando} \quad \alpha \neq \beta. \tag{3.14}$$

En este desarrollo, Γ es descompuesto dentro de dos subconjuntos

$$\pi \subset \Gamma \subset X \quad \text{y} \quad \Delta \subset \Gamma \subset X \tag{3.15}$$

los nodos pertenecientes al primer conjunto π son llamados 'primales', mientras que si estos pertenecen al segundo conjunto Δ son llamados 'duales', donde

$$\Delta \equiv \Gamma - \pi$$

entonces

$$\Gamma = \pi \cup \Delta \quad \text{mientras} \quad \pi \cap \Delta = \varnothing. \tag{3.16}$$

Además, se define

$$\Pi \equiv I \cup \pi$$

en este caso, cada una de las familias de los subconjuntos $\{I, \pi, \Delta\}$ y $\{\Pi, \Delta\}$ son descomposiciones sin traslape de $\overline{\Omega}$, es decir,

$$X = I \cup \pi \cup \Delta \quad \text{mientras que} \quad I \cap \pi = \pi \cap \Delta = \Delta \cap I = \varnothing \tag{3.17}$$

y

$$X = \Pi \cup \Delta \quad \text{mientras que} \quad \Pi \cap \Delta = \varnothing. \tag{3.18}$$

Nótese que todos los conjuntos considerados hasta ahora son dimensionalmente finitos. Por lo tanto, cualquier función con valores en los reales[7] definida en cualquier conjunto define unívocamente un vector dimensionalmente finito.

Para el planteamiento de los métodos Dual-Primal, los nodos de la interfase son clasificados dentro de nodos Primales y Duales. Entonces se define:

- $\hat{N}_I \subset \hat{N}$ como el conjunto de nodos interiores.
- $\hat{N}_\Gamma \subset \hat{N}$ como el conjunto de nodos de la interfase.

[7] Cuando se consideren sistemas de ecuaciones, como en los problemas de elasticidad, tales funciones son vectoriales.

- $\hat{N}_\pi \subset \hat{N}$ como el conjunto de nodos primales.
- $\hat{N}_\Delta \subset \hat{N}$ como el conjunto de nodos duales.

El conjunto $\hat{N}_\pi \subset \hat{N}_\Gamma$ es escogido arbitrariamente y entonces $\hat{N}_\Delta$ es definido como $\hat{N}_\Delta \equiv \hat{N}_\Gamma - \hat{N}_\pi$. Cada una de las siguientes dos familias de nodos son disjuntas:

$$\left\{\hat{N}_I, \hat{N}_\Gamma\right\} \quad \text{y} \quad \left\{\hat{N}_I, \hat{N}_\pi, \hat{N}_\Delta\right\}.$$

Además, estos conjuntos de nodos satisfacen las siguientes relaciones:

$$\hat{N} = \hat{N}_I \cup \hat{N}_\Gamma = \hat{N}_I \cup \hat{N}_\pi \cup \hat{N}_\Delta \quad \text{y} \quad \hat{N}_\Gamma = \hat{N}_\pi \cup \hat{N}_\Delta \tag{3.19}$$

adicionalmente se define los siguientes conjuntos de X :

- $I \equiv \left\{(p,\alpha) \mid p \in \hat{N}_I\right\}.$
- $\Gamma \equiv \left\{(p,\alpha) \mid p \in \hat{N}_\Gamma\right\}.$
- $\pi \equiv \left\{(p,\alpha) \mid p \in \hat{N}_\pi\right\}.$
- $\Delta \equiv \left\{(p,\alpha) \mid p \in \hat{N}_\Delta\right\}.$

En vista de todo lo anterior, la familia de subconjuntos $\left\{X^1, ..., X^E\right\}$ es una descomposición de dominio del conjunto de nodos derivados en donde no se tiene ningún traslape (véase [36] y [38]), es decir

$$X = \bigcup_{\alpha=1}^{E} X^\alpha \quad \text{y} \quad X^\alpha \cap X^\beta = \varnothing \quad \text{cuando} \quad \alpha \neq \beta. \tag{3.20}$$

3.3 El Problema Original

El esquema DVS puede ser aplicado al sistema matricial que es obtenido después de la discretización, este procedimiento es independiente del método de discretización usado —puede ser el método de Elemento Finito, Diferencias Finitas o cualquier otro—. El procedimiento requiere de algunas suposiciones que deben de ser satisfechas, las cuales se explican a continuación (véase [36] y [38]). Tales suposiciones están dadas en términos del sistema matricial y dos conceptos adicionales: los nodos originales y una familia de subconjuntos de tales nodos, los cuales están asociados con la partición del dominio.

Para ilustrar como estos conceptos son introducidos, se considera la formulación variacional de la versión discretizada general de un problema de valores en la frontera, el cual consiste en encontrar $\hat{u} \in V$, tal que

$$a\left(\hat{u}, v\right) = \left(g, v\right), \quad \forall v \in V \tag{3.21}$$

aquí, V es un espacio lineal de dimensión finita de funciones real valuadas[8] definidas en cierto dominio espacial Ω, mientras $g \in V$ es una función dada.

Sea $\hat{N} = \{1, ..., n\}$ el conjunto de índices, el cual es el número de nodos usados en la discretización, y sea $\{\varphi_1, ..., \varphi_n\} \subset V$ una base de V, tal que para cada $i \in \hat{N}, \varphi_i = 1$ en el nodo i y cero en cualquier otro nodo. Entonces, ya que $\hat{u} \in V$, entonces

$$\hat{u} = \sum \widehat{u}_i \varphi_i \tag{3.22}$$

aquí, $\widehat{u}_i$ es el valor de $\hat{u}$ en el nodo i. Sea $\underline{\widehat{u}}$ y $\underline{\widehat{f}}$ vectores[9] $\underline{\widehat{u}} \equiv \left(\widehat{u}_1, ..., \widehat{u}_n\right)$ y $\underline{\widehat{f}} \equiv \left(\widehat{f}_1, ..., \widehat{f}_n\right)$, donde

$$\widehat{f}_i \equiv (g, \varphi_i), \quad i \in \hat{N}. \tag{3.23}$$

La formulación variacional de la Ec.(3.21) es equivalente a

$$\underline{\underline{\widehat{A}}}\,\underline{\widehat{u}} = \underline{\widehat{f}} \tag{3.24}$$

donde la matriz $\underline{\underline{\widehat{A}}}$, será referida como la 'matriz original' y esta es definida mediante

$$\underline{\underline{\widehat{A}}} \equiv \left(\widehat{A}_{ij}\right) \tag{3.25}$$

con

$$\widehat{A}_{ij} \equiv \tilde{a}\left(\varphi_i, \varphi_j\right) \qquad i, j = 1, ..., n \tag{3.26}$$

después de que el problema ha sido discretizado —$\tilde{a}\left(\varphi_i, \varphi_j\right)$ son las funciones base usadas en la discretización—, donde se supone que el dominio Ω ha sido particionado mediante un conjunto de subdominios no traslapados $\{\Omega_1, ..., \Omega_E\}$; más precisamente, para cada $\alpha = 1, ..., E$; Ω_α es abierto y

$$\Omega_\alpha \cap \Omega_\beta = \varnothing \quad \forall \alpha \neq \beta \quad \text{y} \quad \Omega = \bigcup_{\alpha=1}^{E} \overline{\Omega}_\alpha \tag{3.27}$$

donde $\overline{\Omega}_\alpha$ es la clausura estándar del conjunto Ω_α.

El conjunto de subdominios-índices es denotado por $\hat{E} = \{1, ..., E\}$. Por otro lado $\hat{N}^\alpha$, $\alpha = 1, ..., E$, denota a los nodos originales que corresponden a los nodos pertenecientes a $\overline{\Omega}_\alpha$. Como es usual, los nodos son clasificados en 'interiores' y 'nodos de interfase'; un nodo es interior, si pertenece sólo a la clausura de una subpartición del dominio, y es un nodo de la interfase cuando pertenece a más de uno.

Las funciones real valuadas definidas en $\hat{N} = \{1, ..., n\}$ constituyen un espacio lineal el cual será denotado por $\widehat{W}$ y referido como el 'espacio vectorial

[8] La teoría aquí presentada, con ligeras modificaciones, trabaja también en el caso de que las funciones de V sean vectoriales.

[9] Hablando estrictamente estos deberían ser vectores columna, sin embargo, cuando se incorporan en el texto, se escriben como vectores renglón para ahorrar espacio de escritura.

original'. Los vectores $\underline{\widehat{u}} \in \widehat{W}$ se escriben como $\underline{\widehat{u}} = \left(\widehat{u}_1, ..., \widehat{u}_n\right)$, donde $\widehat{u}_i$ para $i = 1, ..., n$, son las componentes de un vector $\underline{\widehat{u}}$.

Entonces, el 'problema original' consiste en "Dada $\underline{\widehat{f}} \in \widehat{W}$, encontrar a $\underline{\widehat{u}} \in \widehat{W}$ tal que la Ec.(3.24) se satisfaga".

A lo largo de todo el desarrollo de esta metodología, la matriz original $\underline{\underline{\widehat{A}}}$ se asume como no singular —esto define una biyección de $\widehat{W}$ sobre sí misma—, para definir las condiciones sobre las cuales el esquema DVS es aplicable para matrices indefinidas y/o no simétricas (véase [34]), se asume lo siguiente, (axioma): "Sean los índices $i \in \hat{N}^\alpha$ y $j \in \hat{N}^\beta$ de nodos interiores originales, entonces

$$\widehat{A}_{ij} = 0 \quad \text{siempre y cuando} \quad \alpha \neq \beta". \tag{3.28}$$

Así, para cada $\alpha = 1, ..., E$ se define el subespacio de vectores $\widehat{W}^\alpha \subset \widehat{W}$, el cual es constituido por los vectores que tienen la propiedad que para cada $i \notin \hat{N}^\alpha$, esta i-ésima componente se nulifica. Usando esta notación se define el espacio producto W por

$$W \equiv \prod_{\alpha=1}^{E} \widehat{W}^\alpha = \widehat{W}^1 \times ... \times \widehat{W}^E. \tag{3.29}$$

3.4 El Espacio de Vectores Derivado

Usando la notación de la sección anterior, en la cual se definió el 'problema original' dado por la Ec.(3.24), este es un problema formulado en el espacio vectorial original $\widehat{W}$, en el desarrollo que sigue se transforma en un problema que será reformulado en el espacio W, el cual es un espacio definido sobre las funciones discontinuas.

Para ello, se entiende por un 'vector derivado' una función real-valuada definida en el conjunto X de nodos derivados[10]. El conjunto de vectores derivados constituyen un espacio lineal, el cual será referido como el 'espacio de vectores derivados'. De manera análoga para cada subconjunto local de nodos derivados en el subespacio X^α existe un 'subespacio local de vectores derivados' W^α, el cual es definido con la condición de que los vectores de W^α se nulifiquen en cada nodo derivado que no pertenezca a X^α. Una manera formal de iniciar esta definición es

$$\underline{u} \in W^\alpha \subset W, \quad \text{si y sólo si} \quad \underline{u}\,(p, \beta) = 0 \quad \text{cuando} \quad \beta \neq \alpha. \tag{3.30}$$

Una importante diferencia entre los subespacios W^α y $\widehat{W}^\alpha$ puede se notada al observar que $W^\alpha \subset W$, mientras que $\widehat{W}^\alpha \subsetneq W$. En particular

$$W \equiv \prod_{\alpha=1}^{E} \widehat{W}^\alpha = W^1 \oplus ... \oplus W^E \tag{3.31}$$

[10] Para el tratamiento de sistemas de ecuaciones, tales como las involucradas en el tratamiento de la elasticidad lineal, tales funciones serán vectoriales.

en palabras: El espacio W es el producto de subespacios de la familia

$$\left\{\widehat{W}^1, ..., \widehat{W}^E\right\} \tag{3.32}$$

pero al mismo tiempo es la suma directa de la familia

$$\left\{W^1, ..., W^E\right\}. \tag{3.33}$$

En vista de la Ec.(3.31) es sencillo establecer una biyección —de hecho, un isomorfismo— entre el espacio de vectores derivados y el espacio producto. Por lo tanto, en lo que sigue, se identifica a ambos como uno solo.

Para cada par de vectores[11] $\underline{u} \in W$ y $\underline{w} \in W$, el 'producto interior Euclidiano' es definido por

$$\underline{u} \cdot \underline{w} = \sum_{(p,\alpha)\in X} \underline{u}(p,\alpha)\, \underline{w}(p,\alpha). \tag{3.34}$$

Una importante propiedad es que el espacio de vectores derivados W, constituye un espacio de Hilbert dimensionalmente finito con respecto al producto interior Euclidiano. Nótese que el producto interior Euclidiano es independiente de la forma de la matriz original $\underline{\underline{\widehat{A}}}$; en particular esta puede ser simétrica, no simétrica o indefinida.

La inyección natural $R : \widehat{W} \to W$, de $\widehat{W}$ sobre W, es definida por la condición de que, para todo $\underline{\hat{u}} \in \widehat{W}$, se obtenga

$$\left(R\underline{\hat{u}}\right)(p,\alpha) = \underline{\hat{u}}(p), \quad \forall\,(p,\alpha) \in X. \tag{3.35}$$

La 'multiplicidad', $m(p)$ de cualquier nodo original $p \in \hat{N}$ es caracterizada por la propiedad (véase [35] y [34]),

$$\sum_{\alpha=1}^{E} \left(R\underline{\hat{u}}\right)(p,\alpha) = m(p)\underline{\hat{u}}(p). \tag{3.36}$$

Además, el espacio W puede ser descompuesto en dos subespacios ortogonales complementarios $W_{11} \subset W$ y $W_{12} \subset W$, tal que

$$W = W_{11} + W_{12} \quad \text{y} \quad \{0\} = W_{11} \cap W_{12} \tag{3.37}$$

donde, el subespacio $W_{12} \subset W$ es la inyección natural de $\widehat{W}$ dentro de W; i.e.

$$W_{12} \equiv R\widehat{W} \subset W \tag{3.38}$$

[11] En el caso vectorial —que surge en aplicaciones como en la teoría de elasticidad— cuando se trabajan sistemas de ecuaciones, i.e. cuando $\underline{u}(p,\alpha)$ es un vector, la Ec(3.34) es reemplazada por

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

donde, $\underline{u}(p,\alpha) \odot \underline{w}(p,\alpha)$ se identifica con el producto interior de vectores involucrados.

y $W_{11} \subset W$ es el complemento ortogonal con respecto al producto interior Euclidiano. Además se define la inversa de $R : \widehat{W} \to W$, cuando ésta es restringida a $W_{12} \subset W$ la cual siempre existe y es denotada por $R^{-1} : W_{12} \to \widehat{W}$.

Es costumbre el uso de la notación de suma directa

$$W = W_{12} \oplus W_{12} \tag{3.39}$$

cuando el par de condiciones de la Ec.(3.37) son satisfechas.

El 'subespacio de vectores continuos' es definido como el subespacio

$$W_{12} \subset W \tag{3.40}$$

mientras que el 'subespacio de vectores de promedio cero' es definido como el subespacio

$$W_{11} \subset W. \tag{3.41}$$

Dos matrices $\underline{\underline{a}} : W \to W$ y $\underline{\underline{j}} : W \to W$ son ahora introducidas; ellas son los operadores proyección con respecto al producto interior Euclidiano sobre W_{12} y W_{11} respectivamente. El primero será referido como el 'operador promedio' y el segundo como el 'operador salto' respectivamente.

En vista de la Ec.(3.37), cada vector $\underline{u} \in W$, puede ser escrito de forma única como la suma de un vector de promedio cero más un vector continuo (vector de salto cero), es decir

$$\underline{u} = \underline{u}_{11} + \underline{u}_{12} \text{ con } \begin{cases} \underline{u}_{11} \equiv \underline{\underline{j}}\underline{u} \in W_{11} \\ \underline{u}_{12} \equiv \underline{\underline{a}}\underline{u} \in W_{12} \end{cases} \tag{3.42}$$

los vectores $\underline{\underline{j}}\underline{u}$ y $\underline{\underline{a}}\underline{u}$ son llamados el 'salto' y el 'promedio' de $\underline{u}$ respectivamente.

Los subespacios lineales que se definen a continuación son elegidos conforme a la nomenclatura estándar. En particular W_I, W_Γ, W_π y W_Δ son definidos para imponer restricciones sobre sus miembros. Los vectores de:

- W_I se nulifica en cada nodo derivado que no es un nodo interno.
- W_Γ se nulifica en cada nodo derivado que no es un nodo de interfase.
- W_π se nulifica en cada nodo derivado que no es un nodo primal.
- W_Δ se nulifica en cada nodo derivado que no es un nodo dual.

Además

- $W_r \equiv W_I + \underline{\underline{a}}W_\pi + W_\Delta$.
- $W_\Pi \equiv W_I + \underline{\underline{a}}W_\pi$.

Nótese que, para cada una de las siguientes familias de subespacios

$$\{W_I, W_\Gamma\}, \{W_I, W_\pi, W_\Delta\}, \{W_\Pi, W_\Delta\} \tag{3.43}$$

son linealmente independientes. Y también satisfacen

$$W = W_I + W_\Gamma = W_I + W_\pi + W_\Delta \text{ y } W_r = W_\Pi + W_\Delta \tag{3.44}$$

la anterior definición de W_r es apropiada cuando se considera las formulaciones Dual-Primal. En el presente trabajo sólo se considera la restricción impuesta por el promedio en los nodos primales —otro ejemplo de restricción es tomar el salto sobre los nodos primales—. Otros tipos de restricciones requieren cambiar el término $\underline{\underline{a}} W_\pi$ por $\underline{\underline{a}}^r W_\pi$ donde $\underline{\underline{a}}^r$ es la proyección sobre el espacio restringido, el tipo de restricciones que pueden ser definidas están en función del problema particular a resolver (véase [29]).

3.5 Discretización Partiendo de la Construcción de la Matriz $\underline{\underline{A}}^t$

Una característica notable del enfoque DVS es que este inicia con la matriz que es obtenida después de que el problema se ha discretizado —a partir de una discretización por algún método como por ejemplo Elemento Finito o Diferencias Finitas— y para su aplicación no se requiere ninguna información acerca de la ecuación diferencial parcial de la cual se originó. Por supuesto, tal matriz no es definida en el espacio de vectores derivados y la teoría provee una fórmula para derivar la matriz en el espacio de vectores derivados (véase [36]); aquí se da un procedimiento para definir la matriz $\underline{\underline{A}}^t : W \to W$, la cual es usada para formular el problema en el espacio de vectores derivados.

Sea la matriz $\widehat{\underline{\underline{A}}} : \widehat{W} \to \widehat{W}$ dada por la Ec.(3.24) la cual puede ser escrita como

$$\widehat{\underline{\underline{A}}} \equiv \left(\widehat{A}_{pq} \right) \tag{3.45}$$

para cada par (p, q) tal que $p \in \hat{N}$ y $q \in \hat{N}$, y se define

$$\delta_{pq}^{\alpha} \equiv \begin{cases} 1, & \text{si } p, q \in \hat{N}^{\alpha} \\ 0, & \text{si } p \notin \hat{N}^{\alpha} \text{ ó } q \notin \hat{N}^{\alpha} \end{cases}, \alpha = 1, ..., E \tag{3.46}$$

junto con

$$m(p, q) \equiv \sum_{\alpha=1}^{N} \delta_{pq}^{\alpha} \tag{3.47}$$

y

$$s(p, q) \equiv \begin{cases} 1, & \text{cuando } m(p, q) = 0 \\ m(p, q), & \text{cuando } m(p, q) \neq 0 \end{cases} \tag{3.48}$$

la función $m(p, q)$ es llamada la 'multiplicidad' del par (p, q). Esto puede verse de la suposición básica dada por la Ec.(3.28), teniendo que

$$m(p, q) = 0 \Rightarrow \widehat{A}_{pq} = 0. \tag{3.49}$$

Definiendo ahora

$$\underline{\underline{\widehat{A}}}^{\alpha} \equiv \left(\widehat{A}_{pq}^{\alpha}\right) \text{ con } \widehat{A}_{pq}^{\alpha} = \frac{\widehat{A}_{pq}\delta_{pq}^{\alpha}}{s(p,q)} \tag{3.50}$$

se observa la identidad

$$\underline{\underline{\widehat{A}}} = \sum_{\gamma=1}^{E} \underline{\underline{\widehat{A}}}^{\gamma} \tag{3.51}$$

ya que se ha usado una verdadera descomposición del dominio sin traslape. Además, para cada $\gamma = 1, ..., E$, la matriz $\underline{\underline{A}}^{\gamma} : W \to W$ es definida por

$$\underline{\underline{A}}^{\gamma} \equiv \left(A_{(p,\alpha)(q,\beta)}^{\gamma}\right) \tag{3.52}$$

con

$$A_{(p,\alpha)(q,\beta)}^{\gamma} \equiv \delta_{\alpha\gamma}^{\gamma}\widehat{A}_{pq}^{\gamma}\delta_{\beta\gamma} \tag{3.53}$$

entonces la matriz $\underline{\underline{A}}^{t} : W \to W$ (t de total, no de transpuesta) es dada por

$$\underline{\underline{A}}^{t} \equiv \sum_{\gamma=1}^{E} \underline{\underline{A}}^{\gamma} \tag{3.54}$$

una propiedad fundamental de $\underline{\underline{A}}^{t}$ es que

$$\left(\underline{\underline{A}}^{t}\right)^{-1} = \sum_{\gamma=1}^{E} \left(\underline{\underline{A}}^{\gamma}\right)^{-1} \tag{3.55}$$

ya que las matrices $\underline{\underline{A}}^{\gamma}$, con $\gamma = 1, 2, ..., E$ son independientes entre si —al ser una verdadera descomposición del dominio, ver sección (3.2)—. Además, otra propiedad es que

$$R^{-1}\underline{\underline{aA}}R = R^{-1}\underline{\underline{aAa}}R = \widehat{A}. \tag{3.56}$$

Nótese que la matriz $\underline{\underline{A}}^{t}$ puede ser expresada en más de una manera. Algunas opciones útiles que se usan son (véase [34] y [35]):

$$\begin{aligned} \underline{\underline{A}}^{t} &= \begin{pmatrix} \underline{\underline{A}}_{\Pi\Pi}^{t} & \underline{\underline{A}}_{\Pi\Delta}^{t} \\ \underline{\underline{A}}_{\Delta\Pi}^{t} & \underline{\underline{A}}_{\Delta\Delta}^{t} \end{pmatrix} = \begin{pmatrix} \underline{\underline{A}}_{II}^{t} & \underline{\underline{A}}_{I\pi}^{t} & \underline{\underline{A}}_{I\Delta}^{t} \\ \underline{\underline{A}}_{\pi I}^{t} & \underline{\underline{A}}_{\pi\pi}^{t} & \underline{\underline{A}}_{\pi\Delta}^{t} \\ \underline{\underline{A}}_{\Delta I}^{t} & \underline{\underline{A}}_{\Delta\pi}^{t} & \underline{\underline{A}}_{\Delta\Delta}^{t} \end{pmatrix} \\ &= \begin{pmatrix} \underline{\underline{A}}^{1} & \underline{\underline{0}} & \cdots & \underline{\underline{0}} \\ \underline{\underline{0}} & \underline{\underline{A}}^{2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \underline{\underline{0}} \\ \underline{\underline{0}} & \cdots & \underline{\underline{0}} & \underline{\underline{A}}^{N} \end{pmatrix}. \end{aligned} \tag{3.57}$$

A la luz de la Ec.(3.55), implica que, si el complemento de Schur local

$$\underline{\underline{S}}^{\alpha} = \underline{\underline{A}}^{\alpha}_{\Gamma\Gamma} - \underline{\underline{A}}^{\alpha}_{\Gamma I}\left(\underline{\underline{A}}^{\alpha}_{II}\right)^{-1}\underline{\underline{A}}^{\alpha}_{I\Gamma} \tag{3.58}$$

de cada $\underline{\underline{A}}^{\alpha}$ existe y es invertible —véase sección (A.7) para la definición y su implementación del complemento de Schur en general—, entonces, la matriz 'total del complemento de Schur', $\underline{\underline{S}}^{t} : W_r \to W_r$, satisface

$$\underline{\underline{S}}^{t} = \sum_{\alpha=1}^{E} \underline{\underline{S}}^{\alpha} \tag{3.59}$$

y

$$\left(\underline{\underline{S}}^{t}\right)^{-1} = \sum_{\alpha=1}^{E} \left(\underline{\underline{S}}^{\alpha}\right)^{-1} \tag{3.60}$$

al ser una verdadera descomposición del dominio —ver sección (3.2)— y porque se satisfacen las Ecs.(3.54 y 3.55).

Por otro lado, si se define $\underline{u}^{'} \equiv R\widehat{\underline{u}}$, usando la Ec.(3.56) entonces el problema original Ec.(3.24), se transforma en uno equivalente a

$$\underline{\underline{a}}\underline{\underline{A}}^{t}\underline{u}^{'} = \underline{\overline{f}} \quad \text{con} \quad \underline{\underline{j}}\underline{u}^{'} = 0 \tag{3.61}$$

una vez que $\underline{u}^{'} \in W$ es obtenida, se puede recuperar $\widehat{\underline{u}} \in \widehat{W}$ usando

$$\widehat{\underline{u}} = R^{-1}\underline{u}^{'} \tag{3.62}$$

esta última es correcta cuando $\underline{u}^{'}$ es evaluada exactamente, pero en las aplicaciones numéricas, $\underline{u}^{'}$ sólo es una evaluación aproximada, por ello puede generar "inestabilidades", en el sentido de que una aproximación a $\underline{u}^{'}$, independientemente de cuan pequeño sea el error, puede no ser continua en cuyo caso $\widehat{u} = R^{-1}\underline{u}^{'}$ no esta definida. Por lo tanto es conveniente reemplazar la Ec.(3.62) por la siguiente versión estable

$$\widehat{\underline{u}} = R^{-1}\left(\underline{\underline{a}}\underline{u}^{'}\right). \tag{3.63}$$

Sea $\underline{\underline{a}}^{r} : W \to W_r$ el operador de proyección ortogonal de W a W_r y nótese que $\underline{\underline{a}} = \underline{\underline{a}}\underline{\underline{a}}^{r}$, entonces, se define

$$\underline{\underline{A}} \equiv \underline{\underline{a}}^{r}\underline{\underline{A}}^{t}\underline{\underline{a}}^{r}. \tag{3.64}$$

Por otro lado, se tiene que

$$\begin{aligned} \underline{\underline{A}} &\equiv \underline{\underline{a}}^{r}\underline{\underline{A}}^{t}\underline{\underline{a}}^{r} \equiv \begin{pmatrix} \underline{\underline{A}}_{\Pi\Pi} & \underline{\underline{A}}_{\Pi\Delta} \\ \underline{\underline{A}}_{\Delta\Pi} & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix} \\ &= \begin{pmatrix} \underline{\underline{A}}^{t}_{II} & \underline{\underline{A}}^{t}_{I\pi}\underline{\underline{a}}^{\pi} & \underline{\underline{A}}^{t}_{I\Delta} \\ \underline{\underline{a}}^{\pi}\underline{\underline{A}}^{t}_{\pi I} & \underline{\underline{a}}^{\pi}\underline{\underline{A}}^{t}_{\pi\pi}\underline{\underline{a}}^{\pi} & \underline{\underline{a}}^{\pi}\underline{\underline{A}}^{t}_{\pi\Delta} \\ \underline{\underline{A}}^{t}_{\Delta I} & \underline{\underline{A}}^{t}_{\Delta\pi}\underline{\underline{a}}^{\pi} & \underline{\underline{A}}^{t}_{\Delta\Delta} \end{pmatrix} \end{aligned} \tag{3.65}$$

la notación usada es tal que

$$\left\{ \begin{array}{ll} \underline{\underline{A}}_{\Pi\Pi} : W_\Pi \to W_\Pi, & \underline{\underline{A}}_{\Pi\Delta} : W_\Delta \to W_\Pi \\ \underline{\underline{A}}_{\Delta\Pi} : W_\Pi \to W_\Delta, & \underline{\underline{A}}_{\Delta\Delta} : W_\Delta \to W_\Delta \end{array} \right. \tag{3.66}$$

donde

$$\left\{ \begin{array}{ll} \underline{\underline{A}}_{\Pi\Pi}\underline{u} = \left(\underline{\underline{A}}\underline{u}_\Pi\right)_\Pi, & \underline{\underline{A}}_{\Delta\Pi}\underline{u} = \left(\underline{\underline{A}}\underline{u}_\Pi\right)_\Delta \\ \underline{\underline{A}}_{\Pi\Delta}\underline{u} = \left(\underline{\underline{A}}\underline{u}_\Delta\right)_\Pi, & \underline{\underline{A}}_{\Delta\Delta}\underline{u} = \left(\underline{\underline{A}}\underline{u}_\Delta\right)_\Delta \end{array} \right. \tag{3.67}$$

por otro lado, se tiene la inmersión natural en W_r, i.e.

$$\begin{aligned} \underline{\underline{A}}_{\Pi\Pi} &\equiv \begin{pmatrix} \underline{\underline{A}}_{\Pi\Pi} & 0 \\ 0 & 0 \end{pmatrix} \\ \underline{\underline{A}}_{\Pi\Delta} &\equiv \begin{pmatrix} 0 & \underline{\underline{A}}_{\Pi\Delta} \\ 0 & 0 \end{pmatrix} \\ \underline{\underline{A}}_{\Delta\Pi} &\equiv \begin{pmatrix} 0 & 0 \\ \underline{\underline{A}}_{\Delta\Pi} & 0 \end{pmatrix} \\ \underline{\underline{A}}_{\Delta\Delta} &\equiv \begin{pmatrix} 0 & 0 \\ 0 & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix}. \end{aligned} \tag{3.68}$$

Así, el 'Complemento de Schur Dual-Primal' está definido por

$$\underline{\underline{S}} = \underline{\underline{A}}_{\Delta\Delta} - \underline{\underline{A}}_{\Delta\Pi}\underline{\underline{A}}^{-1}_{\Pi\Pi}\underline{\underline{A}}_{\Pi\Delta} \tag{3.69}$$

que define al sistema virtual

$$\underline{\underline{S}}\underline{u}_\Gamma = \underline{f}_\Gamma. \tag{3.70}$$

Nótese que

$$\underline{\underline{A}}_{\Pi\Pi} = \begin{pmatrix} \underline{\underline{A}}_{II} & \underline{\underline{A}}_{I\pi} \\ \underline{\underline{A}}_{\pi I} & \underline{\underline{A}}_{\pi\pi} \end{pmatrix} \tag{3.71}$$

así, se definen a las matrices

$$\underline{\underline{A}}^\alpha_{II}, \underline{\underline{A}}^\alpha_{I\pi}, \underline{\underline{A}}^\alpha_{I\Delta}, \underline{\underline{A}}^\alpha_{\pi I}, \underline{\underline{A}}^\alpha_{\pi\pi}, \underline{\underline{A}}^\alpha_{\pi\Delta}, \underline{\underline{A}}^\alpha_{\Delta I}, \underline{\underline{A}}^\alpha_{\Delta\pi} \text{ y } \underline{\underline{A}}^\alpha_{\Delta\Delta} \tag{3.72}$$

las cuales son locales a cada subdominio. Usando esta metodología se puede construir cada componente de $\overset{\frown}{A}{}^\alpha_{pq}$tomando los nodos p y q según correspondan.

Por ejemplo para construir $\underline{\underline{A}}^\alpha_{\pi I}$, en el subdominio $\overline{\Omega}_\alpha$ se construye

$$\underline{\underline{A}}^\alpha_{\pi I} = \left(\overset{\frown}{A}{}^\alpha_{pq}\right) \tag{3.73}$$

donde los nodos $p \in \pi$ y $q \in I$ y ambos pertenecen al $\alpha-$ésimo subdominio de Ω.

3.6 El Problema General con Restricciones

Recordando lo visto en la sección (3.3) en lo referente a la definición del problema original Ec.(3.24). Entonces "Un vector $\underline{\hat{u}} \in \widehat{W}$ es solución del problema original, si y sólo si, $\underline{u}' = R\underline{\hat{u}} \in W_r \subset W$ satisface la expresión

$$\underline{\underline{aA}}\underline{u}' = \overline{\underline{f}} \quad \text{y} \quad \underline{\underline{j}}\underline{u}' = 0 \tag{3.74}$$

el vector

$$\overline{\underline{f}} \equiv \left(R\hat{\underline{f}}\right) \in W_{12} \subset W_r \tag{3.75}$$

puede ser escrito como

$$\overline{\underline{f}} \equiv \overline{\underline{f}}_{\Pi} + \overline{\underline{f}}_{\Delta} \tag{3.76}$$

con $\overline{\underline{f}}_{\Pi} \in W_{\Pi}$ y $\overline{\underline{f}}_{\Delta} \in W_{\Delta}$".

Este es el 'problema Dual-Primal formulado en el espacio de vectores derivados'; o simplemente, el problema Dual-Primal-DVS. Recordando, que este problema esta formulado en el espacio W_r del espacio de vectores derivados W, en el cual las restricciones ya han sido incorporadas. Por lo tanto todos los algoritmos que se desarrollan en las siguientes secciones incluyen tales restricciones; en particular, aquellos impuestos por el promedio de los nodos primales.

Un vector $\underline{u}' \in W$ satisface la Ec.(3.74) si y sólo si, satisface la Ec.(3.61). Para derivar este resultado se usa la propiedad de que cuando $\underline{u}' \in W$ y $\underline{\underline{j}}\underline{u}' = 0$, la siguiente relación se satisface

$$\underline{u}' \in W_r \quad \text{y} \quad \underline{\underline{a}}\left(\underline{\underline{a}}^r \underline{\underline{A}}^t \underline{\underline{a}}^r\right)\underline{u}' = \underline{\underline{aA}}^t \underline{u}'. \tag{3.77}$$

En las discusiones posteriores, la matriz $\underline{\underline{A}} : W_r \to W_r$ se asume invertible, en la mayoría de los casos, este hecho se puede garantizar cuando se toman un número suficientemente grande de nodos primales colocados adecuadamente.

Sea $\underline{u}' \in W_r$ una solución de la Ec.(3.74), entonces $\underline{u}' \in W_{12} \subset W$ necesariamente, ya que $\underline{\underline{j}}\underline{u}' = 0$ y se puede aplicar la inversa de la proyección natural obteniendo

$$\underline{\hat{u}} = \underline{\underline{R}}\underline{u}' \tag{3.78}$$

ya que este problema es formulado en el espacio de vectores derivados; en los algoritmos que se presentan en este trabajo (véase [36]), todas las operaciones son realizadas en dicho espacio; en particular, para realizar los cálculos, nunca se regresa al espacio vectorial original $\widehat{W}$, excepto al final cuando se aplica la Ec.(3.78).

4 Formulaciones Dirichlet-Dirichlet y Neumann-Neumann en el Marco del Espacio de Vectores Derivados

A nivel continuo, los algoritmos más estudiados son el Neumann-Neumann y el Dirichlet-Dirichlet (véase [19] y [3]) y estos fueron bosquejados en este trabajo en el capítulo 2. Durante el desarrollo del esquema del espacio de vectores derivados (DVS), se muestra una clara correspondencia entre el procedimiento a nivel continuo, y el procedimiento a nivel discreto (véase [48]). Usando tal correspondencia el resultado desarrollado puede ser resumido de una forma breve y efectiva, como sigue:

1. Algoritmos no Precondicionados

   - El Algoritmo del Complemento de Schur (la formulación Primal del problema Dirichlet-Dirichlet), sección (4.1.1).
   - La formulación Dual del Problema Neumann-Neumann, sección (4.1.2).
   - La formulación Primal del Problema Neumann-Neumann, sección (4.1.3).
   - La segunda formulación Dual del Problema Neumann-Neumann, sección (4.1.4).

2. Algoritmos Precondicionados

   - La Versión DVS del Algoritmo BDDC (la formulación Dirichlet-Dirichlet precondicionada), sección (4.2.1).
   - La Versión DVS del Algoritmo FETI-DP (la formulación Dual precondicionada del problema Neumann-Neumann), sección (4.2.2).
   - La formulación Primal Precondicionada del problema Neumann-Neumann, sección (4.2.3).
   - La segunda formulación Dual Precondicionada del problema Neumann-Neumann, sección (4.2.4).

Todos estos algoritmos están formulados en el espacio vectorial sujeto a restricciones, tal que todos estos son algoritmos con restricciones.

Para la discusión de todo el material de este capítulo se usa la notación de la sección (3.4), en especial la definición de $\underline{\underline{A}}$ en la Ec.(3.65)

$$\underline{\underline{A}} \equiv \begin{pmatrix} \underline{\underline{A}}_{\Pi\Pi} & \underline{\underline{A}}_{\Pi\Delta} \\ \underline{\underline{A}}_{\Delta\Pi} & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix} \tag{4.1}$$

y la dada por la Ec.(3.69) para la matriz del complemento de Schur

$$\underline{\underline{S}} = \underline{\underline{A}}_{\Delta\Delta} - \underline{\underline{A}}_{\Delta\Pi}\underline{\underline{A}}^{-1}_{\Pi\Pi}\underline{\underline{A}}_{\Pi\Delta}. \tag{4.2}$$

4.1 Algoritmos no Precondicionados

Pese a que los algoritmos no precondicionados carecen de utilidad práctica por su pobre desempeño computacional con respecto a los precondicionados, tienen una gran utilidad teórica y son el camino más sencillo para generar los algoritmos computacionales de los precondicionados, ya que estos permiten probar el marco computacional con algoritmos sencillos y luego sólo se precondicionan para tener el algoritmo final de interés. Se desarrollaron cuatro algoritmos no precondicionados (véase [36]), los cuales se detallan a continuación.

4.1.1 Algoritmo del Complemento de Schur

Sea $\underline{u} = \underline{u}' - \underline{\underline{A}}_{\Pi\Pi}^{-1}\overline{\underline{f}}_{\Pi}$, entonces la Ec.(3.74) del problema general con restricciones —véase sección (3.6)—

$$\underline{\underline{a}}\underline{\underline{A}}\underline{u}' = \overline{\underline{f}} \quad \text{y} \quad \underline{\underline{j}}\underline{u}' = 0 \tag{4.3}$$

es equivalente a: "Dada $\underline{f} = \underline{f}_{\Delta} \in \underline{\underline{a}}W_{\Delta}$, encontrar una $\underline{u}_{\Delta} \in W_{\Delta}$ tal que

$$\underline{\underline{a}}\underline{\underline{S}}\underline{u}_{\Delta} = \underline{f}_{\Delta} \quad \text{y} \quad \underline{\underline{j}}\underline{u}_{\Delta} = 0" \tag{4.4}$$

aquí, $\underline{u} = \underline{u}_{\Pi} + \underline{u}_{\Delta}$, $\underline{f} \equiv \overline{\underline{f}}_{\Delta} - \underline{\underline{A}}_{\Delta\Pi}\underline{\underline{A}}_{\Pi\Pi}^{-1}\overline{\underline{f}}_{\Pi}$ y $\underline{u}_{\Pi} \equiv \underline{\underline{A}}_{\Pi\Pi}^{-1}\underline{\underline{A}}_{\Pi\Delta}\underline{u}_{\Delta}$.

Además, nótese que la matriz del complemento de Schur $\underline{\underline{S}} : W_{\Delta} \to W_{\Delta}$ es invertible cuando $\underline{\underline{A}} : W_r \to W_r$ (véase [35] y [34]).

En el capítulo anterior se mostró la versión continua del problema Dirichlet-Dirichlet no precondicionado, de la cual la Ec.(4.4) es su versión discreta. Por lo tanto este algoritmo pudiera ser llamado 'el algoritmo Dirichlet-Dirichlet no precondicionado'. Sin embargo, en lo que sigue, el algoritmo correspondiente a la Ec.(4.4) será referido como el 'algoritmo del complemento de Schur' ya que es la variante de una de las más simples formas del método de subestructuración la cual se detalla en la sección (A.7).

4.1.2 Formulación Dual del Problema Neumann-Neumann

Para iniciar este desarrollo, se toma la identidad

$$\underline{\underline{a}}\underline{\underline{S}} + \underline{\underline{j}}\underline{\underline{S}} = \underline{\underline{S}} \tag{4.5}$$

esta última ecuación es clara ya que

$$\underline{\underline{a}} + \underline{\underline{j}} = \underline{\underline{I}}. \tag{4.6}$$

Usando las Ecs.(4.4 y 4.5) juntas, implican que

$$\underline{\underline{S}}\underline{u}_{\Delta} = \underline{\underline{a}}\underline{\underline{S}}\underline{u}_{\Delta} + \underline{\underline{j}}\underline{\underline{S}}\underline{u}_{\Delta} = \underline{f}_{\Delta} - \underline{\lambda}_{\Delta} \tag{4.7}$$

donde el vector $\underline{\lambda}_{\Delta}$ es definido por

$$\underline{\lambda}_{\Delta} = -\underline{\underline{j}}\underline{\underline{S}}\underline{u}_{\Delta}. \tag{4.8}$$

Por lo tanto, $\underline{\lambda}_\Delta \in \underline{\underline{j}} W_\Delta$. Entonces, el problema de encontrar $\underline{u}_\Delta$ ha sido transformado en uno, en que se debe encontrar 'el multiplicador de Lagrange $\underline{\lambda}_\Delta$', una vez encontrado $\underline{\lambda}_\Delta$ se puede aplicar $\underline{\underline{S}}^{-1}$ a la Ec.(4.7) para obtener

$$\underline{u}_\Delta = \underline{\underline{S}}^{-1}\left(\underline{f}_\Delta - \underline{\lambda}_\Delta\right). \tag{4.9}$$

Además, en la Ec.(4.9), $\underline{u}_\Delta \in \underline{\underline{a}} W_\Delta$, tal que

$$\underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{f}_\Delta - \underline{\lambda}_\Delta\right) = 0 \tag{4.10}$$

entonces, $\underline{\lambda}_\Delta \in W_\Delta$ satisface

$$\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\lambda}_\Delta = \underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \text{junto con} \quad \underline{\underline{a}}\underline{\lambda}_\Delta = 0 \tag{4.11}$$

por lo anterior, $\underline{\underline{j}}\underline{\underline{S}}\underline{u}_\Delta$ es la versión discreta de el promedio de la derivada normal (véase [35] y [34]).

Formulación usando Multiplicadores de Lagrange Para obtener la formulación de los Multiplicadores de Lagrange, se escribe

$$\left.\begin{array}{c} J(u) = \frac{1}{2}\underline{u}\cdot\underline{\underline{S}}\underline{u} - \underline{u}\cdot\underline{f} \to \min \\ \underline{\underline{j}}\underline{u} = 0 \end{array}\right\} \tag{4.12}$$

tomando la variación en la Ec.(4.12), se obtiene

$$\underline{\underline{S}}\underline{u} + \underline{\underline{j}}\underline{\lambda} = \underline{f} \quad \text{junto con} \quad \underline{\underline{j}}\underline{u} = 0 \tag{4.13}$$

para asegurar la unicidad de $\underline{\lambda}$, se impone la condición de que $\underline{\underline{a}}\underline{u} = 0$, tal que $\underline{\underline{j}}\underline{\lambda} = \underline{\lambda}$. Entonces la Ec.(4.13) implica

$$\underline{\underline{a}}\underline{\underline{S}}\underline{u} + \underline{\underline{j}}\underline{\underline{S}}\underline{u} + \underline{\lambda} = \underline{f} \tag{4.14}$$

multiplicando por $\underline{\underline{j}}$ y $\underline{\underline{a}}$ respectivamente, se obtiene

$$\underline{\lambda} = -\underline{\underline{j}}\underline{\underline{S}}\underline{u} \quad \text{y} \quad \underline{\underline{a}}\underline{\underline{S}}\underline{u} = \underline{f} \tag{4.15}$$

entonces la Ec.(4.8) es clara.

4.1.3 Formulación Primal del Problema Neumann-Neumann

Para iniciar este desarrollo, se toma la Ec.(4.4) del algoritmo del complemento de Schur —véase sección (4.1.1)— y multiplicando la primera igualdad por $\underline{\underline{S}}^{-1}$, y observando que $\underline{\underline{a}}\underline{f}_\Delta = \underline{f}_\Delta$ se obtiene

$$\underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\underline{S}}\underline{u}_\Delta = \underline{\underline{S}}^{-1}\underline{f}_\Delta = \underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{f}_\Delta = \underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\underline{S}}\left(\underline{\underline{S}}^{-1}\underline{f}_\Delta\right) \tag{4.16}$$

de este modo, la Ec.(4.4) puede ser transformada en

$$\underline{\underline{S}}^{-1}\underline{\underline{aS}}\left(\underline{u}_\Delta - \underline{\underline{S}}^{-1}\underline{f}_\Delta\right) = 0 \quad \text{y} \quad \underline{\underline{j}}\underline{u}_\Delta = 0 \tag{4.17}$$

o

$$\underline{\underline{aS}}\left(\underline{u}_\Delta - \underline{\underline{S}}^{-1}\underline{f}_\Delta\right) = 0 \quad \text{y} \quad \underline{\underline{j}}\underline{u}_\Delta = 0. \tag{4.18}$$

Si se define

$$\underline{v}_\Delta = \underline{\underline{S}}^{-1}\underline{f}_\Delta - \underline{u}_\Delta \tag{4.19}$$

entonces la Ec.(4.18) es transformada en

$$\underline{\underline{j}}\underline{v}_\Delta = \underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{\underline{aS}}\underline{v}_\Delta = 0 \tag{4.20}$$

la forma iterativa[12] de este algoritmo se obtiene al multiplicarlo por $\underline{\underline{S}}^{-1}$

$$\underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{v}_\Delta = \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{\underline{aS}}\underline{v}_\Delta = 0. \tag{4.21}$$

Si la solución de la Ec.(4.4) es conocida, entonces $\underline{v}_\Delta \in W_\Delta$, definida por la Ec.(4.19), satisface la Ec.(4.21); a la inversa, si $\underline{v}_\Delta \in W_\Delta$ satisface la Ec.(4.21) entonces

$$\underline{u}_\Delta \equiv \underline{\underline{S}}^{-1}\underline{f}_\Delta - \underline{v}_\Delta \tag{4.22}$$

es solución de la Ec.(4.4).

En lo sucesivo, el algoritmo iterativo definido por la Ec.(4.21) será referido como la 'formulación libre de multiplicadores de Lagrange del problema no precondicionado Neumann-Neumann'.

4.1.4 Segunda Formulación Dual del Problema Neumann-Neumann

En este caso, se toma como inicio la Ec.(4.11) —véase sección (4.1.2)—. Nótese que la siguiente identidad se satisface

$$\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\right) = \underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1} \tag{4.23}$$

entonces, se multiplica la primera igualdad en la Ec.(4.11) por $\underline{\underline{S}}$, obteniéndose

$$\begin{aligned} \underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\right)\underline{\lambda}_\Delta &= \underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\lambda}_\Delta = \underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\right)\underline{f}_\Delta \\ \text{junto con} \quad \underline{\underline{a}}\underline{\lambda}_\Delta &= 0 \end{aligned} \tag{4.24}$$

o

$$\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\left(\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\right)\underline{f}_\Delta - \underline{\lambda}_\Delta\right) \quad \text{junto con} \quad \underline{\underline{a}}\underline{\lambda}_\Delta = 0. \tag{4.25}$$

[12] Para que el algoritmo sea iterativo, se necesita que el dominio y contradominio sean el mismo espacio, que en este caso debe de ser W_Δ.

Si se multiplica la primera de estas igualdades por $\underline{\underline{S}}^{-1}$ y se define

$$\underline{\mu}_\Delta \equiv \underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_\Delta - \underline{\lambda}_\Delta \tag{4.26}$$

la Ec.(4.25) es transformada en

$$\underline{\underline{a}}\underline{\mu}_\Delta = \underline{\underline{a}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\mu}_\Delta = 0. \tag{4.27}$$

Nótese que esta última ecuación es equivalente a la Ec.(4.25) ya que $\underline{\underline{S}}^{-1}$ es no singular. Si la solución de la Ec.(4.11) es conocida, entonces $\underline{\mu}_\Delta \in W_\Delta$ definida por la Ec.(4.26) satisface la Ec.(4.27). A la inversa, si $\underline{\mu}_\Delta \in W_\Delta$ satisface la Ec.(4.27), entonces

$$\underline{\lambda}_\Delta \equiv \underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_\Delta - \underline{\mu}_\Delta \tag{4.28}$$

es solución de la Ec.(4.11).

Por otro lado, la Ec.(4.27) no define un algoritmo iterativo. Sin embargo, si se multiplica la Ec.(4.27) por $\underline{\underline{S}}$ se obtiene el siguiente algoritmo iterativo

$$\underline{\underline{\underline{S}}\underline{\underline{a}}}\underline{\mu}_\Delta = \underline{\underline{\underline{S}}\underline{\underline{a}}\underline{\underline{S}}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\mu}_\Delta = 0 \tag{4.29}$$

esta última ecuación provee una manera iterativa de aplicar la formulación con Multiplicadores de Lagrange. La igualdad $\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\mu}_\Delta = 0$ puede ser interpretada como una restricción; y en efecto, se puede ver que es equivalente a $\underline{\mu}_\Delta \in \underline{\underline{\underline{S}}\underline{\underline{a}}}W_\Delta$.

4.2 Algoritmos Precondicionados

Los algoritmos precondicionados son los que se implementan para resolver los problemas de interés en Ciencias e Ingenierías, ya que su ejecución en equipos paralelos —como Clusters— es eficiente y con buenas características de convergencia (véase [39]). Se desarrollaron cuatro algoritmos precondicionados (véase [36]), los cuales se detallan a continuación.

4.2.1 Versión DVS del Algoritmo BDDC

La versión DVS del algoritmo BDDC —una formulación de este algoritmo se da en la sección (5.2)— se obtiene cuando el algoritmo de complemento de Schur es precondicionado por medio de la matriz $\underline{\underline{a}}\underline{\underline{S}}^{-1}$. Entonces: "Dada $\underline{f}_\Delta \in \underline{\underline{a}}W_\Delta$, encontrar $\underline{u}_\Delta \in W_\Delta$ tal que

$$\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\underline{S}}\underline{u}_\Delta = \underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{\underline{j}}\underline{u}_\Delta = 0\text{"}. \tag{4.30}$$

Para este algoritmo, las siguientes propiedades llaman la atención:

1. Este es un algoritmo iterativo.
2. La matriz iterada es $\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\underline{S}}$.
3. La iteración es realizada dentro del subespacio $\underline{\underline{a}}W_\Delta \subset W_\Delta$.

4. Este algoritmo es aplicable siempre y cuando la matriz del complemento de Schur $\underline{\underline{S}}$ es tal que la implicación lógica

$$\underline{\underline{aS}}^{-1}\underline{w} = 0 \quad \text{y} \quad \underline{\underline{j}}\underline{w} = 0 \Rightarrow \underline{w} = 0 \tag{4.31}$$

es satisfecha, para cualquier $\underline{w} \in W_\Delta$.

5. En particular, es aplicable cuando $\underline{\underline{S}}$ es definida.

Las propiedades 1) a 3) están interrelacionadas. La condición $\underline{\underline{j}}\underline{u}_\Delta = 0$ es equivalente a $\underline{u}_\Delta \in \underline{\underline{a}}W_\Delta$; de este modo, la búsqueda se realiza en el espacio $\underline{\underline{a}}W_\Delta$. Cuando la matriz $\underline{\underline{aS}}^{-1}\underline{\underline{aS}}$ es aplicada repetidamente, siempre se termina en $\underline{\underline{a}}W_\Delta$, ya que para cada $\underline{w} \in W_\Delta$, se obtiene $\underline{\underline{j}}\left(\underline{\underline{aS}}^{-1}\underline{\underline{aS}}\underline{w}\right) = 0$.

Para la propiedad 4), cuando ésta se satisface, la Ec.(4.30) implica la Ec.(4.4). Pare ver esto, nótese que

$$\underline{\underline{j}}\left(\underline{\underline{aS}}\underline{u}_\Delta - \underline{f}_\Delta\right) = 0 \tag{4.32}$$

y también que la Ec.(4.30) implica

$$\underline{\underline{aS}}^{-1}\left(\underline{\underline{aS}}\underline{u}_\Delta - \underline{f}_\Delta\right) = 0 \tag{4.33}$$

cuando la implicación de la Ec.(4.31) se satisface, las Ecs.(4.32 y 4.33) juntas implican que

$$\underline{\underline{aS}}\underline{u}_\Delta - \underline{f}_\Delta = 0 \tag{4.34}$$

como se quería. Esto muestra que la Ec.(4.30) implica la Ec.(4.4), cuando la Ec.(4.31) se satisface.

Para la propiedad 5), nótese que la condición de la Ec.(4.31) es débil y requiere que la matriz del complemento de Schur $\underline{\underline{S}}$ sea definida, ya que la implicación de la Ec.(4.31) es siempre satisfecha cuando $\underline{\underline{S}}$ es definida. Asumiendo que $\underline{\underline{S}}$ es definida, entonces para cualquier vector $\underline{w} \in W_\Delta$ tal que $\underline{\underline{aS}}^{-1}\underline{w} = 0$ y $\underline{\underline{j}}\underline{w} = 0$, se obtiene

$$\underline{w}\cdot\underline{\underline{S}}^{-1}\underline{w} = \underline{w}\cdot\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{w} = \left(\underline{\underline{j}}\underline{w}\right)\cdot\underline{\underline{S}}^{-1}\underline{w} = 0 \tag{4.35}$$

esto implica que $\underline{w} = 0$, ya que $\underline{\underline{S}}^{-1}$ es definida cuando $\underline{\underline{S}}$ lo es. Por lo tanto la propiedad 5) es clara.

4.2.2 Versión DVS del Algoritmo FETI-DP

La versión DVS del algoritmo FETI-DP —una formulación de este algoritmo se da en la sección (5.1)— se obtiene cuando la formulación con Multiplicadores de Lagrange del problema no precondicionado Neumann-Neumann de la Ec.(4.11)

—véase sección (4.1.2)— es precondicionada por medio de la matriz $\underline{\underline{jS}}$. Entonces: "Dada $\underline{f}_\Delta \in \underline{\underline{a}} W_\Delta$, encontrar $\underline{\lambda}_\Delta \in W_\Delta$ tal que

$$\underline{\underline{jS}}\,\underline{\underline{jS}}^{-1}\underline{\lambda}_\Delta = \underline{\underline{jS}}\,\underline{\underline{jS}}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{\underline{a}}\underline{\lambda}_\Delta = 0" \tag{4.36}$$

donde

$$\underline{u}_\Delta = \underline{\underline{aS}}^{-1}\left(\underline{f}_\Delta - \underline{\underline{j}}\underline{\lambda}_\Delta\right). \tag{4.37}$$

Para este algoritmo, las siguientes propiedades llaman la atención:

1. Este es un algoritmo iterativo.
2. La matriz iterada es $\underline{\underline{jS}}\,\underline{\underline{jS}}^{-1}$.
3. La iteración es realizada dentro del subespacio $\underline{\underline{j}} W_\Delta \subset W_\Delta$.
4. Este algoritmo es aplicable siempre y cuando la matriz del complemento de Schur $\underline{\underline{S}}$ es tal que la implicación lógica

   $$\underline{\underline{jS}}\,\underline{w} = 0 \quad \text{y} \quad \underline{\underline{a}}\underline{w} = 0 \Rightarrow \underline{w} = 0 \tag{4.38}$$

   es satisfecha, para cualquier $\underline{w} \in W_\Delta$.
5. En particular, es aplicable cuando $\underline{\underline{S}}$ es positiva definida.

Las propiedades 1) a 3) están interrelacionadas. La condición $\underline{\underline{a}}\underline{\lambda}_\Delta = 0$ es equivalente a $\underline{\lambda}_\Delta \in \underline{\underline{j}} W_\Delta$; de este modo, la búsqueda se realiza en el espacio $\underline{\underline{j}} W_\Delta$. Cuando la matriz $\underline{\underline{jS}}\,\underline{\underline{jS}}^{-1}$ es aplicada repetidamente, siempre se termina en $\underline{\underline{j}} W_\Delta$, ya que para cada $\underline{\mu} \in W_\Delta$, se obtiene $\underline{\underline{a}}\left(\underline{\underline{jS}}\,\underline{\underline{jS}}^{-1}\underline{\mu}\right) = 0$.

Para la propiedad 4), cuando esta se satisface, la Ec.(4.36) implica la Ec.(4.11). Pare ver esto, se asume la Ec.(4.36) y nótese que

$$\underline{\underline{a}}\left(\underline{\underline{jS}}^{-1}\underline{\lambda}_\Delta - \underline{\underline{jS}}^{-1}\underline{f}_\Delta\right) = 0 \tag{4.39}$$

y también que la Ec.(4.36) implica

$$\underline{\underline{jS}}\left(\underline{\underline{jS}}^{-1}\underline{\lambda}_\Delta - \underline{\underline{jS}}^{-1}\underline{f}_\Delta\right) = 0 \tag{4.40}$$

cuando la implicación de la Ec.(4.38) se satisface, las Ecs.(4.39 y 4.40) juntas implican que

$$\underline{\underline{jS}}^{-1}\underline{\lambda}_\Delta - \underline{\underline{jS}}^{-1}\underline{f}_\Delta = 0 \tag{4.41}$$

como se quería. Esto muestra que la Ec.(4.36) implica la Ec.(4.11), cuando la Ec.(4.38) se satisface.

Para la propiedad 5), nótese que la condición de la Ec.(4.38) es débil y requiere que la matriz del complemento de Schur $\underline{\underline{S}}$ sea definida, ya que la implicación de la Ec.(4.38) es siempre satisfecha cuando $\underline{\underline{S}}$ es definida. Asumiendo

que $\underline{\underline{S}}$ es definida, entonces para cualquier vector $\underline{\mu} \in W_\Delta$ tal que $\underline{\underline{j}}\,\underline{\underline{S}}\,\underline{\mu} = 0$ y $\underline{\underline{a}}\,\underline{\mu} = 0$, se obtiene

$$\underline{\mu} \cdot \underline{\underline{S}}\,\underline{\mu} = \underline{\mu} \cdot \underline{\underline{a}}\,\underline{\underline{S}}\,\underline{\mu} = \left(\underline{\underline{a}}\,\underline{\mu}\right) \cdot \underline{\underline{S}}\,\underline{u} = 0 \tag{4.42}$$

esto implica que $\underline{\mu} = 0$, ya que $\underline{\underline{S}}$ es definida. Por lo tanto la propiedad 5) es clara.

4.2.3 Formulación Primal Precondicionada del Problema Neumann-Neumann

Este algoritmo es la versión precondicionada de la formulación libre de multiplicadores de Lagrange del problema no precondicionado Neumann-Neumann. Este puede derivarse multiplicando la Ec.(4.20) —véase sección (4.1.3)— por el precondicionador $\underline{\underline{S}}^{-1}\,\underline{\underline{j}}\,\underline{\underline{S}}$. "Entonces el algoritmo consiste en buscar $\underline{v}_\Delta \in W_\Delta$, tal que

$$\underline{\underline{S}}^{-1}\,\underline{\underline{j}}\,\underline{\underline{S}}\,\underline{\underline{j}}\,\underline{v}_\Delta = \underline{\underline{S}}^{-1}\,\underline{\underline{j}}\,\underline{\underline{S}}\,\underline{\underline{j}}\,\underline{\underline{S}}^{-1}\,\underline{f}_\Delta \quad \text{y} \quad \underline{\underline{a}}\,\underline{\underline{S}}\,\underline{v}_\Delta = 0" \tag{4.43}$$

donde

$$\underline{u}_\Delta = \underline{\underline{a}}\,\underline{\underline{S}}^{-1}\left(\underline{f}_\Delta - \underline{\underline{j}}\,\underline{\underline{S}}\,\underline{v}_\Delta\right). \tag{4.44}$$

Para este algoritmo, las siguientes propiedades llaman la atención:

1. Este es un algoritmo iterativo.
2. La matriz iterada es $\underline{\underline{S}}^{-1}\,\underline{\underline{j}}\,\underline{\underline{S}}\,\underline{\underline{j}}$.
3. La iteración es realizada dentro del subespacio $\underline{\underline{S}}^{-1}\,\underline{\underline{j}}\,W_\Delta \subset W_\Delta$.
4. Este algoritmo es aplicable siempre y cuando la matriz del complemento de Schur $\underline{\underline{S}}$ es tal que la implicación lógica

   $$\underline{\underline{j}}\,\underline{\underline{S}}\,\underline{w} = 0 \quad \text{y} \quad \underline{\underline{a}}\,\underline{w} = 0 \Rightarrow \underline{w} = 0 \tag{4.45}$$

   es satisfecha, para cualquier $\underline{w} \in W_\Delta$.
5. En particular, es aplicable cuando $\underline{\underline{S}}$ es positiva definida.

Las propiedades 1) a 3) están interrelacionadas. La condición $\underline{\underline{a}}\,\underline{\underline{S}}\,\underline{v}_\Delta = 0$ es equivalente a $\underline{v}_\Delta \in \underline{\underline{j}}\,\underline{\underline{S}}^{-1} W_\Delta$; de este modo, la búsqueda se realiza en el espacio $\underline{\underline{j}}\,\underline{\underline{S}}^{-1} W_\Delta$. Cuando la matriz $\underline{\underline{S}}^{-1}\,\underline{\underline{j}}\,\underline{\underline{S}}\,\underline{\underline{j}}$ es aplicada repetidamente, siempre se termina en $\underline{\underline{j}}\,\underline{\underline{S}}^{-1} W_\Delta$, ya que para cada $\underline{v}_\Delta \in W_\Delta$, se obtiene $\underline{\underline{S}}\,\underline{\underline{a}}\left(\underline{\underline{S}}^{-1}\,\underline{\underline{j}}\,\underline{\underline{S}}\,\underline{\underline{j}}\,\underline{v}_\Delta\right) = 0$.

Para la propiedad 4), cuando esta se satisface, la Ec.(4.43) implica la Ec.(4.20). Pare ver esto, se asume la Ec.(4.43) y se define

$$\underline{w} = \underline{\underline{j}}\,\underline{v}_\Delta - \underline{\underline{j}}\,\underline{\underline{S}}^{-1}\,\underline{f}_\Delta \text{ tal que } \underline{\underline{a}}\,\underline{w} = 0 \tag{4.46}$$

además, en vista de la Ec.(4.43) implica

$$\underline{\underline{S}}^{-1}\underline{j}\underline{\underline{S}}\underline{w} = 0 \text{ y por lo tanto } \underline{j}\underline{\underline{S}}\underline{w} = 0 \tag{4.47}$$

usando la Ec.(4.45) se ve que las Ecs.(4.46 y 4.47) juntas implican que

$$\underline{j}\underline{v}_\Delta - \underline{j}\underline{\underline{S}}^{-1}\underline{f}_\Delta = \underline{w} = 0 \tag{4.48}$$

ahora, la Ec.(4.20) es clara y la prueba se completa.

Para la propiedad 5), nótese que la condición de la Ec.(4.45) es débil y requiere que la matriz del complemento de Schur $\underline{\underline{S}}$ sea definida, ya que la implicación de la Ec.(4.45) es siempre satisfecha cuando $\underline{\underline{S}}$ es definida. Asumiendo que $\underline{\underline{S}}$ es definida, entonces para cualquier vector $\underline{w} \in W_\Delta$ tal que $\underline{j}\underline{\underline{S}}\underline{w} = 0$ y $\underline{\underline{a}}\underline{w} = 0$, se obtiene

$$\underline{w}\cdot\underline{\underline{S}}\underline{w} = \underline{w}\cdot\underline{\underline{a}}\underline{\underline{S}}\underline{w} = \left(\underline{\underline{a}}\underline{w}\right)\cdot\underline{\underline{S}}\underline{w} = 0 \tag{4.49}$$

esto implica que $\underline{w} = 0$, ya que $\underline{\underline{S}}$ es definida. Por lo tanto la propiedad 5) es clara.

4.2.4 Segunda Formulación Dual Precondicionada del Problema Neumann-Neumann

Este algoritmo es la versión precondicionada de la segunda formulación con multiplicadores de Lagrange del problema no precondicionado Neumann-Neumann. Este puede derivarse multiplicando la Ec.(4.27) —véase sección (4.1.4)— por el precondicionador $\underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\underline{S}}$. "Entonces el algoritmo consiste en buscar $\underline{\mu}_\Delta \in W_\Delta$, tal que

$$\underline{\underline{S}}\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\mu}_\Delta = \underline{\underline{S}}\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\underline{S}}\underline{j}\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{j}\underline{\underline{S}}^{-1}\underline{\mu}_\Delta = 0" \tag{4.50}$$

donde

$$\underline{u}_\Delta = \underline{\underline{a}}\underline{\underline{S}}^{-1}\left(\underline{f}_\Delta + \underline{\mu}_\Delta\right) \tag{4.51}$$

este algoritmo es similar a FETI-DP.

Para este algoritmo, las siguientes propiedades llaman la atención:

1. Este es un algoritmo iterativo.
2. La matriz iterada es $\underline{\underline{S}}\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{a}}$.
3. La iteración es realizada dentro del subespacio $\underline{\underline{S}}\underline{\underline{a}}W_\Delta \subset W_\Delta$.
4. Este algoritmo es aplicable siempre y cuando la matriz del complemento de Schur $\underline{\underline{S}}$ es tal que la implicación lógica

   $$\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{w} = 0 \quad \text{y} \quad \underline{j}\underline{w} = 0 \Rightarrow \underline{w} = 0 \tag{4.52}$$

   es satisfecha, para cualquier $\underline{w} \in W_\Delta$.
5. En particular, es aplicable cuando $\underline{\underline{S}}$ es positiva definida.

Las propiedades 1) a 3) están interrelacionadas. La condición $\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\mu}_{\Delta} = 0$ es equivalente a $\underline{\mu}_{\Delta} \in \underline{\underline{Sa}}W_{\Delta}$; de este modo, la búsqueda se realiza en el espacio $\underline{\underline{Sa}}W_{\Delta}$. Cuando la matriz $\underline{\underline{SaS}}^{-1}\underline{\underline{a}}$ es aplicada repetidamente, siempre se termina en $\underline{\underline{Sa}}W_{\Delta}$, ya que para cada $\underline{\mu}_{\Delta} \in W_{\Delta}$, se obtiene $\underline{\underline{aS}}\left(\underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\mu}_{\Delta}\right) = 0$.

Para la propiedad 4), cuando esta se satisface, la Ec.(4.50) implica la Ec.(4.27). Para ver esto, se asume la Ec.(4.50) y se define

$$\underline{\eta} = \underline{\underline{a}}\underline{\mu}_{\Delta} - \underline{\underline{aS}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_{\Delta} \text{ tal que } \underline{\underline{j}}\underline{\eta} = 0 \tag{4.53}$$

además, en vista de la Ec.(4.50) se obtiene

$$\underline{\underline{SaS}}^{-1}\underline{\eta} = 0 \tag{4.54}$$

usando las Ecs.(4.53 y 4.54), juntas implican que

$$\underline{\underline{a}}\underline{\eta} - \underline{\underline{aS}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_{\Delta} = \underline{\eta} = 0 \tag{4.55}$$

ahora, la Ec.(4.27) es clara, como se quería probar, además se ve que la Ec.(4.50) implica la Ec.(4.27), cuando la condición de la Ec.(4.52) se satisface.

Para la propiedad 5), nótese que la condición de la Ec.(4.52) es débil y requiere que la matriz del complemento de Schur $\underline{\underline{S}}$ sea definida, ya que la implicación de la Ec.(4.52) es siempre satisfecha cuando $\underline{\underline{S}}$ es definida. Asumiendo que $\underline{\underline{S}}$ es definida, entonces para cualquier vector $\underline{w} \in W_{\Delta}$ tal que $\underline{\underline{aS}}^{-1}\underline{w} = 0$ y $\underline{\underline{j}}\underline{w} = 0$, se obtiene

$$\underline{w} \cdot \underline{\underline{S}}^{-1}\underline{w} = \underline{w} \cdot \underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{w} = \left(\underline{\underline{j}}\underline{w}\right) \cdot \underline{\underline{S}}^{-1}\underline{w} = 0 \tag{4.56}$$

esto implica que $\underline{w} = 0$, ya que $\underline{\underline{S}}^{-1}$ es definida cuando $\underline{\underline{S}}$ lo es. Por lo tanto la propiedad 5) es clara.

4.3 El Operador de Steklov-Poincaré

El operador de Steklov-Poincaré generalmente se define como la ecuación para la traza de la solución exacta u sobre Γ del problema dado por la Ec.(2.4), donde el complemento de Schur —definido en la Ec.(3.70)— es una aproximación discreta del operador de Steklov-Poincaré (véase [49]). La discusión de este operador juega un papel importante en el desarrollo de la teoría del espacio de vectores derivados (véase [48]), para iniciar la discusión, se usa la siguiente definición.

Definición 1 *Sea $\underline{\underline{A}} : W_r \to W_r$ una matriz simétrica y positiva definida. El 'producto interior de energía' es definido por*

$$(\underline{u}, \underline{w}) \equiv \underline{u} \cdot \underline{\underline{A}}\underline{w} \quad \forall \underline{u}, \underline{w} \in W_r. \tag{4.57}$$

El espacio lineal, W_r, es un espacio de Hilbert (dimensionalmente finito) cuando este es dotado con el producto interior de energía. Usando la notación de la sección (3.4) y recordando que la matriz $\underline{\underline{A}}$ se escribe como

$$\underline{\underline{A}} \equiv \begin{pmatrix} \underline{\underline{A}}_{\Pi\Pi} & \underline{\underline{A}}_{\Pi\Delta} \\ \underline{\underline{A}}_{\Delta\Pi} & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix} \tag{4.58}$$

donde la notación usada es tal que

$$\begin{cases} \underline{\underline{A}}_{\Pi\Pi} : W_\Pi \to W_\Pi, & \underline{\underline{A}}_{\Pi\Delta} : W_\Delta \to W_\Pi \\ \underline{\underline{A}}_{\Delta\Pi} : W_\Pi \to W_\Delta, & \underline{\underline{A}}_{\Delta\Delta} : W_\Delta \to W_\Delta \end{cases} \tag{4.59}$$

además

$$\begin{cases} \underline{\underline{A}}_{\Pi\Pi}\underline{u} = \left(\underline{\underline{A}}\underline{u}_\Pi\right)_\Pi, & \underline{\underline{A}}_{\Delta\Pi}\underline{u} = \left(\underline{\underline{A}}\underline{u}_\Pi\right)_\Delta \\ \underline{\underline{A}}_{\Pi\Delta}\underline{u} = \left(\underline{\underline{A}}\underline{u}_\Delta\right)_\Pi, & \underline{\underline{A}}_{\Delta\Delta}\underline{u} = \left(\underline{\underline{A}}\underline{u}_\Delta\right)_\Delta \end{cases} \tag{4.60}$$

y nótese que se tiene la inmersión natural en W_r, i.e.

$$\begin{aligned} \underline{\underline{A}}_{\Pi\Pi} &\equiv \begin{pmatrix} \underline{\underline{A}}_{\Pi\Pi} & 0 \\ 0 & 0 \end{pmatrix} \\ \underline{\underline{A}}_{\Pi\Delta} &\equiv \begin{pmatrix} 0 & \underline{\underline{A}}_{\Pi\Delta} \\ 0 & 0 \end{pmatrix} \\ \underline{\underline{A}}_{\Delta\Pi} &\equiv \begin{pmatrix} 0 & 0 \\ \underline{\underline{A}}_{\Delta\Pi} & 0 \end{pmatrix} \\ \underline{\underline{A}}_{\Delta\Delta} &\equiv \begin{pmatrix} 0 & 0 \\ 0 & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix}. \end{aligned} \tag{4.61}$$

Ahora, se introducen las siguientes definiciones:

Definición 2 *Sea la matriz $\underline{\underline{L}}$ definida como*

$$\underline{\underline{L}} \equiv \begin{pmatrix} \underline{\underline{A}}_{\Pi\Pi} & \underline{\underline{A}}_{\Pi\Delta} \\ 0 & 0 \end{pmatrix} \tag{4.62}$$

y la matriz $\underline{\underline{R}}$ definida como

$$\underline{\underline{R}} \equiv \begin{pmatrix} 0 & 0 \\ \underline{\underline{A}}_{\Delta\Pi} & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix}. \tag{4.63}$$

Además, nótese la siguiente identidad

$$\underline{\underline{R}} = \underline{\underline{aR}} + \underline{j}\underline{\underline{R}} \tag{4.64}$$

implica también que $\underline{\underline{A}} = \underline{\underline{A}}^T$, así

$$\underline{\underline{L}} + \underline{\underline{aR}} + \underline{j}\underline{\underline{R}} = \underline{\underline{L}}^T + \underline{\underline{R}}^T\underline{a} + \underline{\underline{R}}^T\underline{j}. \tag{4.65}$$

Definición 3 *A la identidad*

$$\underline{\underline{L}} + \underline{\underline{\underline{aR}}} - \underline{\underline{R}}^T \underline{\underline{j}} = \underline{\underline{L}}^T + \underline{\underline{R}}^T \underline{\underline{a}} - \underline{\underline{\underline{jR}}} \tag{4.66}$$

la cual se deriva de la Ec.(4.65), la cual será referida como la fórmula Green-Herrera para matrices.

Nótese que los rangos de $\underline{\underline{L}}$ y $\underline{\underline{R}}$ son W_Π y W_Δ, respectivamente, mientras que los rangos de $\underline{\underline{\underline{aR}}}$ y $\underline{\underline{\underline{jR}}}$ están contenidos en $W_{12}(\Delta)$ y $W_{11}(\Delta)$ respectivamente. Inclusive más, estos últimos dos rangos son linealmente independientes. Además, para cualquier función $\underline{v} \in W_r$ se obtiene

$$\left(\underline{\underline{L}} + \underline{\underline{\underline{aR}}} - \underline{\underline{R}}^T \underline{\underline{j}}\right) \underline{v} = 0 \tag{4.67}$$

si y sólo si

$$\underline{\underline{L}}\underline{v} = 0, \underline{\underline{\underline{aR}}}\underline{v} = 0 \quad \text{y} \quad \underline{\underline{j}}\underline{v} = 0. \tag{4.68}$$

Esto establece la equivalencia entre las Ec.(4.67) y Ec.(4.68) y se puede usar el hecho de que los rangos de $\underline{\underline{L}}$ y $\underline{\underline{\underline{aR}}}$ son linealmente independientes, junto con la ecuación

$$\left(\underline{\underline{j}}\underline{v}\right) \cdot \underline{\underline{R}}^T \underline{\underline{j}}\underline{v} = \left(\underline{\underline{j}}\underline{v}\right) \cdot \underline{\underline{A}}_{\Delta\Delta} \underline{\underline{j}}\underline{v} = \left(\underline{\underline{j}}\underline{v}\right) \cdot \underline{\underline{\underline{A}}}\underline{\underline{j}}\underline{v} = 0 \tag{4.69}$$

la cual implica que $\underline{\underline{j}}\underline{v} = 0$. Esto es porque $\underline{\underline{\underline{A}}}$ es positiva definida sobre W_r.

En lo que sigue, se usa la siguiente notación, para cada $\underline{u} \in W_r$, se escribe

$$\dot{\underline{\widehat{u}}} \equiv \underline{\underline{a}}\underline{u} \quad y \quad [\![\underline{u}]\!] \equiv \underline{\underline{j}}\underline{u} \tag{4.70}$$

entonces $\dot{\underline{\widehat{u}}} \in W_r$, mientras $[\![\underline{u}]\!]$ pertenecen a $W_{11}(\Gamma) \subset W_r$. La fórmula de Green-Herrera de la Ec.(4.66) es equivalente a

$$\underline{w} \cdot \underline{\underline{L}}\underline{u} + \dot{\underline{\widehat{w}}} \cdot \underline{\underline{\underline{aR}}}\underline{u} - [\![\underline{u}]\!] \underline{\underline{\underline{jR}}}\underline{w} = \underline{u} \cdot \underline{\underline{L}}\underline{w} + \dot{\underline{\widehat{u}}} \cdot \underline{\underline{\underline{aR}}}\underline{w} - [\![\underline{w}]\!] \underline{\underline{\underline{jR}}}\underline{u} \tag{4.71}$$

$\forall \underline{u}, \underline{w} \in W_r$. Ahora, las fórmulas de Green-Herrera que originalmente fueron introducidas para operadores diferenciales parciales actuando sobre funciones discontinuas, pueden ser aplicadas a cualquier operador cuando este es lineal. La Ec.(4.66) por otro lado, es vista como una extensión de este tipo de fórmulas actuando sobre vectores discontinuos y se tiene interés de comparar la Ec.(4.71) con las fórmulas de Green-Herrera para operadores diferenciales parciales. Para hacer esto, se introduce la siguiente notación

$$[\![\underline{\underline{R}}]\!] = -\underline{\underline{\underline{aR}}} \quad \text{y} \quad \dot{\underline{\underline{\widehat{R}}}} \equiv -\underline{\underline{\underline{jR}}} \tag{4.72}$$

haciendo uso de esta notación, la Ec.(4.71) se reescribe como

$$\underline{w} \cdot \underline{\underline{L}}\underline{u} + [\![\underline{u}]\!] \cdot \dot{\underline{\underline{\widehat{R}}}}\,\underline{w} - \dot{\underline{\widehat{w}}} \cdot [\![\underline{\underline{R}}]\!]\,\underline{u} = \underline{u} \cdot \underline{\underline{L}}\underline{w} + [\![\underline{w}]\!] \cdot \dot{\underline{\underline{\widehat{R}}}}\,\underline{u} - \dot{\underline{\widehat{u}}} \cdot [\![\underline{\underline{R}}]\!]\,\underline{w} \tag{4.73}$$

$\forall \underline{u}, \underline{w} \in W_r$.

Para el operador diferencial de Laplace actuando sobre funciones discontinuas definidas por tramos que satisfacen condiciones de frontera homogéneas, la fórmula Green-Herrera queda como

$$\int_\Omega w\mathcal{L}u dx + \int_\Gamma \left\{ [\![u]\!] \dot{\widehat{\frac{\partial w}{\partial n}}} - \dot{\widehat{w}} \left[\!\left[\frac{\partial u}{\partial n} \right]\!\right] \right\} dx =$$

$$\int_\Omega u\mathcal{L}w dx + \int_\Gamma \left\{ [\![w]\!] \dot{\widehat{\frac{\partial u}{\partial n}}} - \dot{\widehat{u}} \left[\!\left[\frac{\partial w}{\partial n} \right]\!\right] \right\} dx \tag{4.74}$$

la siguiente correspondencia entre las funcionales bilineales involucradas en ambas ecuaciones, comparando con las Ecs. (4.73 y 4.74) se obtiene

$$\begin{array}{c} \int_\Omega w\mathcal{L}u dx \leftrightarrow \underline{w} \cdot \underline{\underline{L}}\underline{u} \\ \int_\Gamma [\![u]\!] \dot{\widehat{\frac{\partial w}{\partial n}}} dx \leftrightarrow [\![\underline{u}]\!] \cdot \dot{\underline{\underline{\widehat{R}}}}\, \underline{w} \\ \int_\Gamma \dot{\widehat{w}} \left[\!\left[\frac{\partial u}{\partial n} \right]\!\right] dx \leftrightarrow \dot{\underline{\widehat{w}}} \cdot [\![\underline{\underline{R}}]\!]\, \underline{u}. \end{array} \tag{4.75}$$

Para operadores diferenciales, en particular para el operador de Laplace, el operador de Steklov-Poincaré asociado con el salto de la derivada normal y de la funcional bilineal es

$$\int_\Gamma \dot{\widehat{w}} \left[\!\left[\frac{\partial u}{\partial n} \right]\!\right] dx \tag{4.76}$$

a nivel matricial, el operador de Steklov-Poincaré es asociado con la forma bilineal

$$\dot{\underline{w}} \cdot [\![\underline{\underline{R}}]\!]\, \underline{u} = \underline{w} \cdot \underline{\underline{aR}}\underline{u}, \forall \underline{u}, \underline{w} \in W_r \tag{4.77}$$

o más simplemente, con la matriz $\underline{\underline{aR}}$.

Definición 4 *Se define al operador de Steklov-Poincaré como la matriz*

$$\underline{\underline{aR}}. \tag{4.78}$$

En el esquema de vectores derivados (DVS), la versión discreta del operador de Steklov-Poincaré es $\underline{\underline{aS}}$, por lo tanto, la versión discreta de la Ec.(2.14) es

$$\underline{\underline{aS}}\underline{v} = \underline{\underline{aS}}\underline{u_p} \qquad \text{junto con } \underline{\underline{j}}\underline{v} = 0 \tag{4.79}$$

la cual puede ponerse en correspondencia con la Ec.(4.4) reemplazando

$$-\left[\!\left[\frac{\partial u_p}{\partial n} \right]\!\right] \leftrightarrow \underline{f}_\Delta \qquad \text{y} \qquad \underline{v} \leftrightarrow \underline{u}_\Delta. \tag{4.80}$$

Otra formulación al operador de Steklov-Poincaré, se deriva de la fórmula de Green-Herrera para el operador elíptico general, simétrico y de segundo orden

$$\int_{\Omega} w\mathcal{L}u dx + \int_{\Gamma} \left\{ [\![u]\!] \, \underline{a}_n \cdot \widehat{\nabla} w - \dot{\widehat{w}} \, [\![\underline{a}_n \cdot \nabla u]\!] \right\} dx =$$

$$\int_{\Omega} u\mathcal{L}w dx + \int_{\Gamma} \left\{ [\![w]\!] \, \underline{a}_n \cdot \widehat{\nabla} u - \dot{\widehat{u}} \, [\![\underline{a}_n \cdot \nabla w]\!] \right\} dx. \tag{4.81}$$

La correspondencia de la ecuación (4.75) todavía permanece para este caso, excepto que

$$\begin{cases} [\![\underline{a}_n \cdot \nabla u]\!] \leftrightarrow - \, [\![\underline{\underline{R}}]\!] \, \underline{u} \\ \underline{a}_n \cdot \widehat{\nabla} w \leftrightarrow - \dot{\widehat{\underline{\underline{R}}}} \underline{u} \end{cases} . \tag{4.82}$$

Correspondencias similares a las dadas por las Ecs. (4.75 y 4.82) pueden ser establecidas en general —su aplicación incluye a los sistemas gobernantes de ecuaciones de elasticidad lineal y muchos problemas más—. Nótese que las Ecs. (4.75 y 4.82) implican una nueva fórmula para el operador *Steklov-Poincaré,* i.e., el salto de la derivada normal en el nivel discreto, el cual es diferente a las interpretaciones estándar que han sido presentadas por muchos autores. La fórmula (véase [34]) para el operador *Steklov-Poincaré* es

$$- \, [\![\underline{\underline{R}}]\!] \, \underline{u} \equiv -\underline{\underline{j}} \underline{\underline{R}} \tag{4.83}$$

en particular, esta no contiene el lado derecho de la ecuación para ser resuelta; ganando con ello, en consistencia teórica. Nótese que la fórmula es aplicable para cualquier vector (función) independientemente de si está es solución del problema bajo consideración o no.

Aplicando la fórmula de Green-Herrera al problema transformado dado al inicio de este capítulo, se tiene el siguiente resultado:

Teorema 5 *Sea* $\overline{\underline{f}} = \begin{pmatrix} \overline{\underline{f}} \\ \overline{\underline{\underline{f}}}^{\Pi} \\ \underline{f}_{\Delta} \end{pmatrix} \in W_r = W_{12}\left(\overline{\Omega}\right)$, *entonces una* $\underline{v} \in W_r$ *satisface*

$$\left(\underline{\underline{L}} + \underline{\underline{aR}} - \underline{\underline{R}}^T \underline{\underline{j}} \right) \underline{v} = \overline{\underline{f}} \tag{4.84}$$

si y sólo si, $\underline{v}$ *es solución del el problema transformado.*

Demostración. Sea $\underline{u} \in W_r$ una solución del problema transformado y asumiendo que $\underline{v} \in W_r$ satisface la Ec.(4.84) entonces

$$\left(\underline{\underline{L}} + \underline{\underline{aR}} - \underline{\underline{R}}^T \underline{\underline{j}} \right) \underline{u} = \left(\underline{\underline{L}} + \underline{\underline{aR}} \right) \underline{u} = \underline{\underline{aAu}} = \overline{\underline{f}} \tag{4.85}$$

para probar el inverso, se define $\underline{w} = \underline{v} - \underline{u}$, así se obtiene

$$\left(\underline{\underline{L}} + \underline{\underline{aR}} - \underline{\underline{R}}^T \underline{\underline{j}} \right) \underline{w} = 0 \tag{4.86}$$

y usando las Ec.(4.67) y Ec.(4.68), se tiene que $\underline{v} = \underline{u} + \underline{w}$ satisface

$$\underline{\underline{A}}\,\underline{v} = \left(\underline{\underline{L}} + \underline{\underline{aR}}\right)\underline{v} = \left(\underline{\underline{L}} + \underline{\underline{aR}}\right)\underline{u} = \underline{\underline{aA}}\,\underline{u} = \overline{f} \tag{4.87}$$

y

$$\underline{\underline{j}}\,\underline{v} = \underline{\underline{j}}\,\underline{u} = 0. \tag{4.88}$$

■

Por otro lado, para obtener la versión discreta del operador μ definido en la Ec.(2.21), primero se escribe la versión discreta de la Ec.(2.19), esta es

$$\underline{\underline{j}}\,\underline{\underline{S}}\,\underline{v} = \underline{q}_{\Gamma} \qquad \text{junto con} \qquad \underline{\underline{aS}}\,\underline{v} = 0 \tag{4.89}$$

por lo tanto, se tiene que

$$\underline{\underline{aq}}_{\Gamma} = 0 \text{ y } \underline{\underline{j}}\,\underline{v} = \underline{\underline{j}}\,\underline{\underline{S}}^{-1}\underline{\underline{S}}\,\underline{v} = \underline{\underline{j}}\,\underline{\underline{S}}^{-1}\underline{\underline{j}}\,\underline{\underline{S}}\,\underline{v} = \underline{\underline{j}}\,\underline{\underline{S}}^{-1}\underline{q}_{\Gamma} \tag{4.90}$$

esto establece la correspondencia de

$$\mu \leftrightarrow \underline{\underline{j}}\,\underline{\underline{S}}^{-1} \tag{4.91}$$

la versión discreta de la Ec.(2.22) es

$$\underline{\underline{j}}\,\underline{\underline{S}}^{-1}\underline{q}_{\Gamma} = -\underline{\underline{j}}\,\underline{u}_{p} \qquad \text{junto con} \qquad \underline{\underline{aq}}_{\Gamma} = 0. \tag{4.92}$$

Otra opción de abordar la versión discreta de la Ec.(2.20) sin hacer uso de la contraparte del operador de Steklov-Poincaré μ, en el cual, el correspondiente problema es

$$\underline{\underline{j}}\,\underline{v} = -\underline{\underline{j}}\,\underline{u}_{p} \qquad \text{y} \qquad \underline{\underline{aS}}\,\underline{v} = 0 \tag{4.93}$$

sin embargo, esta última ecuación no define un algoritmo iterativo, ya que

$$\underline{\underline{aS}}\,\underline{\underline{j}} \neq 0. \tag{4.94}$$

Una ecuación equivalente a la Ec.(4.93), la cual puede ser aplicada en un algoritmo iterativo es

$$\underline{\underline{S}}^{-1}\underline{\underline{j}}\,\underline{v} = -\underline{\underline{S}}^{-1}\underline{\underline{j}}\,\underline{u}_{p} \qquad \text{y} \qquad \underline{\underline{aS}}\,\underline{v} = 0. \tag{4.95}$$

5 Los Métodos FETI-DP y BDDC en el Marco de DVS

En el presente trabajo se ha introducido el método de descomposición de dominio referido como el espacio de vectores derivados (DVS). Este esquema es capaz de englobar a los métodos FETI-DP y BDDC ampliamente usados. Nótese que algunos de los algoritmos desarrollados, corresponden a los métodos FETI-DP y BDDC, pero simplificando y generalizando a estos métodos.

Para mostrar la correspondencia entre DVS y los métodos FETI-DP y DBBC, en lo que resta se este capítulo se usa la notación de estos enfoques, para ello se inicia considerando el problema dado por la ecuación

$$\begin{aligned} \mathcal{L}u &= f \quad \text{en } \Omega \\ u &= g \quad \text{sobre } \partial\Omega \end{aligned} \tag{5.1}$$

en el dominio Ω, el cual es subdividido en E subdominios Ω_i, $i = 1, 2, ..., E$ sin traslape, también conocida como malla gruesa $\mathcal{T}_H$, es decir

$$\Omega_i \cap \Omega_j = \varnothing \quad \forall i \neq j \quad \text{y} \quad \overline{\Omega} = \bigcup_{i=1}^{E} \overline{\Omega}_i \tag{5.2}$$

y al conjunto

$$\Gamma = \bigcup_{i=1}^{E} \Gamma_i, \quad \text{si } \Gamma_i = \partial\Omega_i \backslash \partial\Omega \tag{5.3}$$

se llama la frontera interior del dominio Ω. La notación $\partial\Omega$ y $\partial\Omega_i$, $i = 1, ..., E$ es tomada de la frontera del dominio Ω y la frontera del subdominio Ω_i respectivamente, claramente

$$\partial\Omega \subset \bigcup_{i=1}^{E} \partial\Omega_i \quad \text{y} \quad \Omega = \left(\bigcup_{i=1}^{E} \Omega_i \right) \bigcup \Gamma. \tag{5.4}$$

Para llevar a cabo el complemento de Schur del sistema lineal asociado a la discretización de la ecuación diferencial parcial, se introduce una reordenación de la solución u de los nodos, como

$$\begin{aligned} u_I &\rightarrow \text{nodos interiores} \\ u_\Gamma &\rightarrow \text{nodos sobre la interfase } \Gamma \end{aligned}$$

tal que el sistema lineal asociado $Au = f$ tome la forma

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} u_I \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_I \\ f_\Gamma \end{pmatrix} \tag{5.5}$$

donde las variables asociadas a los nodos interiores han sido reordenadas por subdominios, obteniendo

$$A_{II} = \begin{pmatrix} A_{II}^{(1)} & & 0 \\ & \ddots & \\ 0 & \cdots & A_{II}^{(E)} \end{pmatrix}. \tag{5.6}$$

Eliminando las variables asociadas a los nodos interiores u_I —véase sección (A.7)—, se obtiene el complemento de Schur

$$Su_\Gamma = \tilde{f}_\Gamma \tag{5.7}$$

donde

$$S = A_{\Gamma\Gamma} - A_{\Gamma I}\left(A_{II}\right)^{-1} A_{I\Gamma} \tag{5.8}$$

y $\tilde{f}_\Gamma = f_\Gamma - A_{\Gamma I}\left(A_{II}\right)^{-1} A_{I\Gamma}$. Además, se define S_i como el complemento local de Schur al subdominio Ω_i derivado de la matriz $A^{(i)}$, como

$$S_i = A^{(i)}_{\Gamma\Gamma} - A^{(i)}_{\Gamma I}\left(A^{(i)}_{II}\right)^{-1} A^{(i)}_{I\Gamma}. \tag{5.9}$$

5.1 El Método FETI-DP en el Marco de DVS

Tomando como base la metodología de Finite Element Tearing and Interconnect Dual-Primal (FETI-DP) (véase [19] pág. 156) y haciendo las adecuaciones pertinentes, esta se pondrá en términos del esquema DVS, en esta sección se trabajará con el caso de Multiplicadores de Lagrange redundantes, i.e. Encontrar $u \in W$ tal que

$$\left.\begin{array}{c} J(u) = \frac{1}{2}\langle Su, u\rangle - \langle f, u\rangle \to \min \\ B_r u = 0 \end{array}\right\} \tag{5.10}$$

donde la matriz B_r denota al operador de salto

$$B_r = [B_r^{(1)}, B_r^{(2)}, ..., B_r^{(E)}] \tag{5.11}$$

tal que los valores de u asociados a más de un subdominio coincidan, es decir

$$B_r u = 0 \tag{5.12}$$

donde $B_r^{(i)}$ consiste de las columnas de B_r atribuidas a la i-ésima componente del espacio producto W.

El vector de Multiplicadores de Lagrange es denotado por λ_r. El renglón de $B_r^{(i)}$ relativo al Multiplicador de Lagrange que hace cumplir la continuidad entre los valores en los nodos de $w_i \in W_i$ y $w_j \in W_j$, en $x \in \partial\Omega_{i,h} \cap \partial\Omega_{j,h}$, es escalado por $\delta_j^\dagger(x)$ y este factor de escala define el correspondiente elemento de $D_r^{(i)}$. Finalmente, se define al operador escalado de salto por

$$B_{D_r} = (D_r^{(1)}B_r^{(1)}, D_r^{(2)}B_r^{(2)}, ..., D_r^{(E)}B_r^{(E)}). \tag{5.13}$$

Con la introducción de un vector de Multiplicadores de Lagrange λ para hacer cumplir las restricciones $B_r u = 0$, se obtiene una formulación punto silla de la Ec.(5.10): Encontrar $(u, \lambda) \in W \times U$ tal que

$$\left\{\begin{array}{c} Su + B_r^T \lambda = f \\ B_r u = 0 \end{array}\right. \tag{5.14}$$

sustituyendo la expresión para u en la segunda ecuación de Ec.(5.14), entonces

$$B_r S^\dagger B_r^T \lambda = B_r S^\dagger f - B_r R\alpha \tag{5.15}$$

y finalmente se obtiene el sistema

$$\begin{cases} F_r\lambda - G_r\alpha = d_r \\ G_r^T\lambda = e \end{cases} . \tag{5.16}$$

La matriz del sistema lineal reducido puede ser escrito como

$$F_r = B_r S^\dagger B_r^T \tag{5.17}$$

donde el precondicionador de FETI esta dado por

$$\widehat{M_r}^{-1} = B_{D_r} S B_{D_r}^T = \sum_{i=1}^{E} D_r^{(i)} B_r^{(i)} S^{(i)} B_r^{(i)T} D_r^{(i)} \tag{5.18}$$

que define el sistema precondicionado

$$P_r \widehat{M_r}^{-1} P_r^T F_r \lambda_r = P_r \widehat{M_r}^{-1} P_r^T d_r \tag{5.19}$$

con

$$P_r = I - Q_r G_r \left(G_r^T Q_r G_r\right)^{-1} G_r^T \tag{5.20}$$

donde

$$G_r = B_r R \qquad \text{y} \qquad d_r = B_r S^\dagger f \tag{5.21}$$

con la condición inicial λ_0 escogida tal que

$$G_r \lambda_0 = e \tag{5.22}$$

donde $e = R^T f$.

Desarrollando la expresión dada por la Ec.(5.19) y reemplazando $\widehat{M_r}^{-1}$ de la Ec.(5.18), se obtiene

$$P_r \left(B_{D_r} S B_{D_r}^T\right) P_r^T \left(B_r S^\dagger B_r^T\right) \lambda_r = P_r \left(B_{D_r} S B_{D_r}^T\right) P_r^T \left(B_r S^\dagger f\right) \tag{5.23}$$

si se asume que $Q_r = I$, entonces $P_r = I$, así

$$\left(B_{D_r} S B_{D_r}^T\right) \left(B_r S^\dagger B_r^T\right) \lambda_r = \left(B_{D_r} S B_{D_r}^T\right) \left(B_r S^\dagger f\right) \tag{5.24}$$

reemplazando $B_{D_r} = D_r B_r$, resulta

$$D_r B_r S \left(D_r B_r\right)^T B_r S^\dagger B_r^T \lambda_r = D_r B_r S \left(D_r B_r\right)^T B_r S^\dagger f \tag{5.25}$$

y simplificando, se obtiene

$$D_r B_r S B_r^T D_r B_r S^\dagger B_r^T \lambda_r = D_r B_r S B_r^T D_r B_r S^\dagger f. \tag{5.26}$$

Ahora, ya que Br sólo debe satisfacer la restricción $B_r u = 0$, entonces es posible sustituir con j, definida como $j = I - a$ —véase la Ec.(4.6)—, ésta también satisface $ju = 0$. Entonces reemplazando B_r por j, se obtiene

$$D_r j S j^T D_r j S^\dagger j^T \lambda_r = D_r j S j^T D_r j S^\dagger f \tag{5.27}$$

además, como $j = j^T$ (véase [31]), entonces

$$D_r j S j D_r j S^\dagger j \lambda_r = D_r j S j D_r j S^\dagger f \tag{5.28}$$

y en este caso $S^\dagger = S^{-1}$, entonces

$$D_r j S j D_r j S^{-1} j \lambda_r = D_r j S j D_r j S^{-1} f. \tag{5.29}$$

Suponiendo que la matriz diagonal D_r es la matriz diagonal igual a la identidad I, entonces

$$j S j j S^{-1} j \lambda_r = j S j j S^{-1} f \tag{5.30}$$

por otro lado, $jj = j$ por se idempotente (véase [31]), finalmente se obtiene

$$j S j S^{-1} j \lambda_r = j S j S^{-1} f \tag{5.31}$$

esta es una formulación equivalente a la formulación del método de FETI precondicionado en términos de DVS —véase la Ec.(4.36)—.

Ahora, se desarrolla la expresión dada por la Ec.(5.19) sin precondicionar

$$F_r \lambda_r = d_r \tag{5.32}$$

sustituyendo F_r y d_r, entonces

$$B_r S^\dagger B_r^T \lambda_r = B_r S^\dagger f \tag{5.33}$$

y reemplazando B_r por j y $S^\dagger$ por S^{-1}, se obtiene

$$j S^{-1} j \lambda_r = j S^{-1} f \tag{5.34}$$

esta es una formulación equivalente al método de FETI sin precondicionar en términos del esquema DVS —véase la Ec.(4.36)—.

5.2 El Método BDDC en el Marco de DVS

Tomando como base la metodología (véase [29]) de Balancing Domain Decomposition (BDD) y haciendo las adecuaciones pertinentes, esta se pondrá en términos del esquema DVS. Para ello, sea u definida sobre Γ_i, entonces se define el complemento local de Schur al subdominio Ω_i como

$$S_i = A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)} \right)^{-1} A_{I\Gamma}^{(i)} \tag{5.35}$$

y al operador restricción $\bar{R}_i : \Gamma \to \Gamma_i$, entonces se obtiene

$$S = \sum_{i=1}^{E} \bar{R}_i^T S_i \bar{R}_i. \tag{5.36}$$

Observación 6 *Nótese que*

$$S_i \neq \bar{R}_i S \bar{R}_i^T \tag{5.37}$$

ya que está última involucra contribuciones de los subdominios vecinos. Sin embargo, dado $u \in V_\Gamma^{(i)}$ *entonces*

$$u^T S_i u \leq u^T \left(\bar{R}_i S \bar{R}_i^T \right) u \tag{5.38}$$

ya que S *y* S_i *son positivas definidas.*

Una primera definición para el método BDD fue inicialmente propuesto para la ecuación de Poisson por De Roeck y Le Tallec (véase [29]), su idea fue usar un Schwarz aditivo como precondicionador de la forma

$$M^{-1} = \sum_{i=1}^{E} R_i^T S_i^{-1} R_i \tag{5.39}$$

donde $R_i : \Gamma \to \Gamma_i$ es un operador de restricción pesado definido como

$$R_i = D_i^{-1} \bar{R}_i \tag{5.40}$$

donde S_i es el complemento de Schur local de la matriz local $A^{(i)}$, además

$$\bar{R}_i : \Gamma \to \Gamma_i \tag{5.41}$$

es el operador discreto de restricción a Γ_i, y D_i^{-1} es una matriz diagonal definida como una partición de la unidad sobre Γ, i.e.

$$\sum_{i=1}^{N} \bar{R}_i^T D_i^{-1} \bar{R}_i = I \tag{5.42}$$

donde I es la identidad sobre Γ. La partición de la unidad puede ser definida a través de una función de conteo, la cual para cada subdominio Ω_i es definida como el operador $\delta_i : \Gamma_i \to \mathbb{R}$ tal que

$$\delta_i (x) = \{\text{numero de subdominios que comparten el nodo } x \in \Gamma_i\}. \tag{5.43}$$

Entonces, se define $D_i = diag\{\delta_i\}$.

El método BDD precondicionado queda en términos de

$$M^{-1} S u = M^{-1} f \tag{5.44}$$

sustituyendo S y M^{-1} —Ecs.(5.8 y 5.39)—, se tiene que

$$\begin{aligned} & \left(\sum_{i=1}^{E} \left(D_i^{-1} \bar{R}_i \right)^T S_i^{-1} D_i^{-1} \bar{R}_i \right) \left(\sum_{i=1}^{E} \bar{R}_i^T S_i \bar{R}_i \right) u \qquad (5.45) \\ = & \left(\sum_{i=1}^{E} \left(D_i^{-1} \bar{R}_i \right)^T S_i^{-1} D_i^{-1} \bar{R}_i \right) f \end{aligned}$$

y finalmente sustituyendo S_i, i.e Ec.(5.35), se obtiene

$$\begin{aligned} & \left(\sum_{i=1}^{E} \bar{R}_i^T D_i^{-1} \left(A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)}\right)^{-1} A_{I\Gamma}^{(i)}\right)^{-1} D_i^{-1} \bar{R}_i\right) \\ & \left(\sum_{i=1}^{E} \bar{R}_i^T \left(A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)}\right)^{-1} A_{I\Gamma}^{(i)}\right) \bar{R}_i\right) u \\ = & \left(\sum_{i=1}^{E} \bar{R}_i^T D_i^{-1} \left(A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)}\right)^{-1} A_{I\Gamma}^{(i)}\right)^{-1} D_i^{-1} \bar{R}_i\right) f. \end{aligned} \tag{5.46}$$

Multiplicando u por $\left(A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)}\right)^{-1} A_{I\Gamma}^{(i)}\right) \bar{R}_i$ se obtienen E vectores $\tilde{u}_i$, tales que al aplicar $D_i^{-1} \bar{R}_i \left(\sum_{i=1}^{E} \bar{R}_i^T \tilde{u}_i\right)$ es equivalente a aSu. A estos E vectores resultantes $\hat{u}_i$, se aplica $\sum_i^E \bar{R}_i^T D_i^{-1} \left(A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)}\right)^{-1} A_{I\Gamma}^{(i)}\right)^{-1} \hat{u}_i$, siendo equivalente a $aS^{-1}aSu$. Que es el lado derecho de la Ec.(4.30).

Haciendo algo similar al lado izquierdo, se obtienen E vectores $\tilde{f}_i = D_i^{-1} \bar{R} f$ tales que al multiplicar por $\sum_i^E \bar{R}_i^T D_i^{-1} \left(A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)}\right)^{-1} A_{I\Gamma}^{(i)}\right)^{-1} \tilde{f}_i$, es equivalente a $aS^{-1}f$.

Así, la Ec.(5.46) es equivalente a

$$aS^{-1}aSu = aS^{-1}f \tag{5.47}$$

que es la formulación del método BDDC en términos del esquema DVS —véase Ec.(4.30)—.

5.3 Comparaciones

El enfoque desarrollado del espacio de vectores derivado (DVS) y por lo tanto el esquema DVS de FETI-DP y BDDC, inician con la matriz que es obtenida después de que el problema ha sido discretizado y para su aplicación no se requiere información acerca del sistema de ecuaciones diferenciales parciales del cual la matriz es originaria. Generalizando, todos los algoritmos que han sido presentados en este trabajo son igualmente aplicables a matrices simétricas, indefinidas y no simétricas. Las condiciones específicas requeridas para su aplicación son detalladas en el capítulo 3 (véase [33]), a través de todo el desarrollo se asume que el complemento de la matriz de Schur $\underline{\underline{S}}$ es no singular.

Como se mencionó antes, para el algoritmo FETI se muestra que la versión DVS de FETI-DP puede ser obtenida cuando se aplican las adecuadas condiciones a las expresiones generales de FETI-DP. Aunque, esas opciones representen un caso particular del algoritmo general de FETI-DP, en algún sentido las elecciones hechas son óptimas ya que las matrices $\underline{\underline{a}}$ y $\underline{\underline{j}}$ son proyecciones ortogonales complementarias (véase [34] y [33]); además de otros ejemplos numéricos

que se muestran en este trabajo en el capítulo de Análisis y Discusión de Resultados.

Cuando se lleva a cabo la incorporación del método BDDC en el esquema DVS se encuentran diferencias más sustanciales. Por ejemplo, cuando las inversas del complemento de Schur local existen en el esquema DVS la inversa de $\underline{\underline{S}}^t$ está dada por

$$\left(\underline{\underline{S}}^t\right)^{-1} = \sum_{\alpha=1}^{E} \left(\underline{\underline{S}}^\alpha\right)^{-1} \tag{5.48}$$

una relación similar no es satisfecha por el algoritmo BDDC. A saber, la ecuación íntimamente relacionada es por ejemplo Ec.(5.36)

$$\underline{\underline{S}} = \sum_{\alpha=1}^{E} \underline{\underline{\bar{R}}}_\alpha^T \underline{\underline{S}}_\alpha \underline{\underline{\bar{R}}}_\alpha \tag{5.49}$$

sin embargo

$$\left(\underline{\underline{S}}\right)^{-1} \neq \sum_{\alpha=1}^{E} \left(\underline{\underline{\bar{R}}}_\alpha^T \underline{\underline{S}}_\alpha \underline{\underline{\bar{R}}}_\alpha\right)^{-1} \tag{5.50}$$

aquí, la igualdad no se satisface debido a que los rangos de $\underline{\underline{\bar{R}}}_\alpha^T \underline{\underline{S}}_\alpha \underline{\underline{\bar{R}}}_\alpha$ y $\underline{\underline{\bar{R}}}_\beta^T \underline{\underline{S}}_\beta \underline{\underline{\bar{R}}}_\beta$ no son ajenos, cuando $\alpha \neq \beta$. Esta última limitación es debido al hecho que en BDDC, la formulación no se establece directamente en el espacio producto y que en su implementación se retorna a los grados de libertad asociados con los nodos originales, mientras en el esquema DVS uno se olvida completamente de los nodos originales y se trabaja exclusivamente con los nodos derivados, es decir, de los nodos que resultan después de que los nodos originales han sido partidos. Por lo tanto, las fórmulas desarrolladas para el esquema DVS y sus códigos computacionales se simplifican de manera notable. La unificación y simplificación lograda de esta manera permite producir Software genérico, robusto y eficiente.

6 Formulación Numérica de los Métodos DVS

Partiendo de las formulaciones Dirichlet-Dirichlet y Neumann-Neumann en el marco del espacio de vectores derivados (DVS) —véase sección (4)—, aquí se muestran los detalles de la formulación numérica la cual esta íntimamente relacionada con la implementación computacional, tanto en la versión secuencial como paralela del código (véase [39] y [38]).

La implementación computacional dicta ciertas consideraciones sobre selección de los algoritmos numéricos, así como, la forma de implementación de estos. Con la única finalidad de aprovechar sus propiedades computacionales en la definición de una robusta jerarquía de clases que resuelva de forma eficiente el problema.

Para ello, se inicia considerando el operador de segundo orden dado por la ecuación

$$\begin{aligned} \mathcal{L}u &= f \quad \text{en } \Omega \\ u &= g \quad \text{sobre } \partial\Omega \end{aligned} \tag{6.1}$$

en el dominio Ω, el cual es subdividido en E subdominios Ω_α, $\alpha = 1, 2, ..., E$ mediante una descomposición de dominio sin traslape —véase sección (3.2)—. Para resolver dicho problema, se puede usar cualquiera de los ocho algoritmos que se han desarrollado, de cada uno de ellos se han dado formulaciones explícitas en términos de matrices.

Así, para generar la implementación de cualquiera de los algoritmos desarrollados se necesita generar las matrices locales

$$\underline{\underline{A}}^\alpha_{\Pi\Pi}, \underline{\underline{A}}^\alpha_{\Pi\Delta}, \underline{\underline{A}}^\alpha_{\Delta\Pi}, \underline{\underline{A}}^\alpha_{\Delta\Delta} \tag{6.2}$$

que están definidas de $W_r \to W_r$, o más explícitamente las matrices

$$\underline{\underline{A}}^\alpha_{II}, \underline{\underline{A}}^\alpha_{I\pi}, \underline{\underline{A}}^\alpha_{I\Delta}, \underline{\underline{A}}^\alpha_{\pi I}, \underline{\underline{A}}^\alpha_{\pi\pi}, \underline{\underline{A}}^\alpha_{\pi\Delta}, \underline{\underline{A}}^\alpha_{\Delta I}, \underline{\underline{A}}^\alpha_{\Delta\pi} \text{ y } \underline{\underline{A}}^\alpha_{\Delta\Delta} \tag{6.3}$$

ello se puede hacer de dos formas, a saber:

- A partir de la matriz que es obtenida después de que el problema se ha discretizado —a partir de una discretización por el método de Elemento Finito, Volumen Finito o Diferencias Finitas— y para su aplicación no se requiere ninguna información acerca de la ecuación diferencial parcial de la cual se originó.
- A partir la discretización de la ecuación diferencial parcial generada localmente en cada subdominio por algún método de descomposición de dominio sin traslapes, como el usado para FETI-DP o BDDC.

En la sección (3.5) se derivo la forma de obtener las matrices locales a partir de la matriz $\underline{\underline{A}}^t : W \to W$ que es obtenida después de que el problema se ha discretizado y la cual es puesta en el espacio de vectores derivados. En la

siguiente sección se da la forma de derivar cada una de las matrices locales a partir la discretización de la ecuación diferencial parcial generada localmente en cada subdominio; para que en las siguientes secciones, mediante el uso de dichas matrices se pueda implementar cualquiera de los métodos desarrollados.

6.1 Discretización de los Métodos Partiendo de la Formulación Local

Los métodos de descomposición de dominio —como son FETI-DP y BDDC— y el esquema DVS puede partir de la discretización local mediante algún método de discretización como Diferencias Finitas, Volumen Finito o Elemento Finito –este último es uno de los más usados en los DDM— para construir cada una de las matrices locales

$$\underline{\underline{A}}^{\alpha}_{II}, \underline{\underline{A}}^{\alpha}_{I\pi}, \underline{\underline{A}}^{\alpha}_{I\Delta}, \underline{\underline{A}}^{\alpha}_{\pi I}, \underline{\underline{A}}^{\alpha}_{\pi\pi}, \underline{\underline{A}}^{\alpha}_{\pi\Delta}, \underline{\underline{A}}^{\alpha}_{\Delta I}, \underline{\underline{A}}^{\alpha}_{\Delta\pi} \text{ y } \underline{\underline{A}}^{\alpha}_{\Delta\Delta} \tag{6.4}$$

para cada Ω_α con $\alpha = 1, ..., E$. Una vez construida las matrices locales, se procede a definir al complemento de Schur local por subdominio —más detalles véase sección (A.7)—

$$\underline{\underline{S}}^{\alpha}_{\pi} = \underline{\underline{A}}^{\alpha}_{\pi\pi} - \underline{\underline{A}}^{\alpha}_{\pi I} \left(\underline{\underline{A}}^{\alpha}_{II} \right)^{-1} \underline{\underline{A}}^{\alpha}_{I\pi} \tag{6.5}$$

con él se define

$$\underline{\underline{A}}^{\alpha}_{\Pi\Pi} = \begin{pmatrix} \underline{\underline{A}}^{\alpha}_{II} & \underline{\underline{A}}^{\alpha}_{I\pi} \\ \underline{\underline{A}}^{\alpha}_{\pi I} & \underline{\underline{A}}^{\alpha}_{\pi\pi} \end{pmatrix} \tag{6.6}$$

que a su vez define

$$\underline{\underline{S}}^{\alpha} = \underline{\underline{A}}^{\alpha}_{\Delta\Delta} - \underline{\underline{A}}^{\alpha}_{\Delta\Pi} \left(\underline{\underline{A}}^{\alpha}_{\Pi\Pi} \right)^{-1} \underline{\underline{A}}^{\alpha}_{\Pi\Delta} \tag{6.7}$$

recordando que

$$\begin{cases} \underline{\underline{A}}_{\Pi\Pi} = \sum_{\alpha=1}^{E} \underline{\underline{A}}^{\alpha}_{\Pi\Pi}, & \underline{\underline{A}}_{\Pi\Delta} = \sum_{\alpha=1}^{E} \underline{\underline{A}}^{\alpha}_{\Pi\Delta} \\ \underline{\underline{A}}_{\Delta\Pi} = \sum_{\alpha=1}^{E} \underline{\underline{A}}^{\alpha}_{\Delta\Pi}, & \underline{\underline{A}}_{\Delta\Delta} = \sum_{\alpha=1}^{E} \underline{\underline{A}}^{\alpha}_{\Delta\Delta} \end{cases} \tag{6.8}$$

$$\underline{\underline{S}} = \sum_{\alpha=1}^{E} \underline{\underline{S}}^{\alpha} \qquad \text{y} \qquad \left(\underline{\underline{S}} \right)^{-1} = \sum_{\alpha=1}^{E} \left(\underline{\underline{S}}^{\alpha} \right)^{-1} \tag{6.9}$$

entonces, finalmente se tienen definidas a $\underline{\underline{S}}$ y $\underline{\underline{S}}^{-1}$. Con estas definiciones, ahora ya es posible implementar cualesquiera de los métodos del esquema del espacio de vectores derivados.

Nótese que, por la forma de construcción de las matrices, se tienen las siguientes propiedades importantes

$$\left(\underline{\underline{A}}_{\Pi\Pi} \right)^{-1} = \sum_{\alpha=1}^{E} \left(\underline{\underline{A}}^{\alpha}_{\Pi\Pi} \right)^{-1} \qquad \text{y} \qquad \left(\underline{\underline{A}} \right)^{-1} = \sum_{\alpha=1}^{E} \left(\underline{\underline{A}}^{\alpha} \right)^{-1} \tag{6.10}$$

esto implica que el cálculo de la inversa de la matriz $\underline{\underline{A}}_{\Pi\Pi}$ es exclusivamente a partir de las inversas locales a cada subdominio.

6.2 Formulación Operacional de los Métodos DVS

Para la resolución de problemas de interés en Ciencias e Ingenierías donde es necesario resolver, por ejemplo el problema dado por la Ec.(3.74) —más detalles véase la sección (3.6)— que consiste en buscar una función $\underline{u}^{'} \in W_r$ que satisfaga

$$\underline{\underline{aA}}\,\underline{u}^{'} = \underline{\overline{f}} \qquad \text{y} \qquad \underline{\underline{j}}\,\underline{u}^{'} = 0 \tag{6.11}$$

aquí, el vector $\underline{\overline{f}} \in W_{12}$ es un dato del problema y donde el vector $\underline{u}^{'}$ y $\underline{\overline{f}}$ esta formado por

$$\begin{pmatrix} \underline{u}_{\Pi} \\ \underline{u}_{\Delta} \end{pmatrix} = \underline{u}^{'} \qquad \text{y} \qquad \begin{pmatrix} \underline{f}_{\Pi} \\ \underline{f}_{\Delta} \end{pmatrix} = \underline{\overline{f}} \tag{6.12}$$

que satisfaga —véase sección (4.3) del operador Steklov-Poincaré Ecs.(4.62 y 4.63)—

$$\left(\underline{\underline{L}} + \underline{\underline{aR}} - \underline{\underline{R}}^{T}\,\underline{\underline{j}}\right)\underline{u}^{'} = \underline{\overline{f}}. \tag{6.13}$$

Entonces es necesario transformar el problema Ec.(6.13) en uno que

$$\underline{f}_{\Pi} = 0 \tag{6.14}$$

para ello, se introduce un vector auxiliar

$$\underline{u}_p = \underline{\underline{A}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{f}_{\Pi} \tag{6.15}$$

en el cual $\left(\underline{u}_p\right)_{\Delta} = 0$, por lo tanto la Ec.(6.13) toma la forma

$$\left(\underline{\underline{aR}} - \underline{\underline{R}}^{T}\,\underline{\underline{j}}\right)\underline{u}_{\Delta} = \underline{f}_{\Delta} - \underline{u}_p \tag{6.16}$$

así, la solución $\underline{u}^{'}$ al problema será $\underline{u}^{'} = \underline{u}_{\Delta} - \underline{u}_p$.

Dado que se necesita expresar el problema en términos del espacio W_{12} entonces

$$\underline{f}_{\Delta_2} = \underline{\underline{a}}\,\underline{f}_{\Delta}, \qquad \text{y} \qquad \underline{f}_{\Delta_1} = \underline{\underline{j}}\,\underline{f}_{\Delta} = 0 \tag{6.17}$$

$$\left(\underline{u}_p\right)_{\Delta} = \underline{\underline{aA}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{f}_{\Pi} \tag{6.18}$$

de lo anterior, la expresión dada por la Ec.(6.16), se puede reescribir como

$$\left(\underline{\underline{aS}} - \underline{\underline{Sj}}\right)\underline{u}_{\Delta} = \underline{\underline{a}}\,\underline{f}_{\Delta} - \underline{\underline{aA}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{f}_{\Pi} \tag{6.19}$$

donde

$$\underline{\underline{S}} = \underline{\underline{A}}_{\Delta\Delta} - \underline{\underline{A}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{\underline{A}}_{\Pi\Delta} \tag{6.20}$$

o más compactamente se escribe como

$$\left(\underline{\underline{aS}} - \underline{\underline{Sj}}\right)\underline{u}_{\Delta} = \underline{f}_{\Delta} - \left(\underline{u}_p\right)_{\Delta}. \tag{6.21}$$

Por todo lo anterior, los algoritmos desarrollados quedan escritos de manera explícita como:

1. **Formulacion del Algoritmo del Complemento de Schur** (PRIMAL#1) —véase ecuación (4.4) en la sección (4.1.1)—:

   "Encontrar a $\underline{u}_\Delta \in W_\Delta$ tal que

   $$\underline{\underline{aS}}\underline{u}_\Delta = \underline{f}_\Delta - \left(\underline{u}_p\right)_\Delta \tag{6.22}$$

   sujeto a $\underline{\underline{j}}\underline{u}_\Delta = 0$". O en su forma desarrollada

   $$\underline{\underline{aS}}\underline{u}_\Delta = \underline{f}_\Delta - \underline{\underline{aA}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{f}_\Pi. \tag{6.23}$$

2. **Formulación Dual del Problema Neumann-Neumann** (DUAL#1) —véase la ecuación (4.11) de la sección (4.1.2)—:

   "Dada $\underline{f}_\Delta \in W_\Delta$, buscar a $\underline{\lambda} \in W_\Delta$ tal que

   $$\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\lambda}_\Delta = \underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{f}_\Delta - \left(\underline{u}_p\right)_\Delta\right) \tag{6.24}$$

   sujeto a $\underline{\underline{a}}\underline{\lambda}_\Delta = 0$". O en su forma desarrollada

   $$\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\lambda}_\Delta = \underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{f}_\Delta - \underline{\underline{aA}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{f}_\Pi\right). \tag{6.25}$$

3. **Formulación Primal del Problema Neumann-Neumann** (PRIMAL#2) —véase ecuación (4.21) en la sección (4.1.3)—:

   "Dada $\underline{f}_\Delta \in W_\Delta$, encontrar $\underline{v}_\Delta \in W_\Delta$ tal que

   $$\underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{v}_\Delta = \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{f}_\Delta - \left(\underline{u}_p\right)_\Delta\right) \tag{6.26}$$

   sujeto a $\underline{\underline{aS}}\underline{v}_\Delta = 0$". O en su forma desarrollada

   $$\underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{v}_\Delta = \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{f}_\Delta - \underline{\underline{aA}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{f}_\Pi\right). \tag{6.27}$$

4. **Formulación Dual del Problema Neumann-Neumann** (DUAL#2) —véase la ecuación (4.29) de la sección (4.1.4)—:

   "Dada $\underline{f}_\Delta \in W_\Delta$, sea $\underline{\mu}_\Delta \in W_\Delta$ tal que

   $$\underline{\underline{Sa}}\underline{\mu}_\Delta = \underline{\underline{SaS}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{f}_\Delta - \left(\underline{u}_p\right)_\Delta\right) \tag{6.28}$$

   sujeto a $\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\mu}_\Delta = 0$". O en su forma desarrollada

   $$\underline{\underline{Sa}}\underline{\mu}_\Delta = \underline{\underline{SaS}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{f}_\Delta - \underline{\underline{aA}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{f}_\Pi\right). \tag{6.29}$$

5. **Formulación Precondicionada del Algoritmo BDDC** (PRIMAL#1) —véase la ecuación (4.30) en la sección (4.2.1)—:

   "Buscar a $\underline{u}_\Delta \in W_\Delta$ tal que

$$\underline{\underline{aS}}^{-1}\underline{\underline{aS}}\underline{u}_\Delta = \underline{\underline{aS}}^{-1}\left(\underline{f}_\Delta - \left(\underline{u}_p\right)_\Delta\right) \tag{6.30}$$

   sujeto a $\underline{\underline{j}}\underline{u}_\Delta = 0$". O en su forma desarrollada

$$\underline{\underline{aS}}^{-1}\underline{\underline{aS}}\underline{u}_\Delta = \underline{\underline{aS}}^{-1}\left(\underline{f}_\Delta - \underline{\underline{aA}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{f}_\Pi\right). \tag{6.31}$$

6. **Formulación Precondicionada del Algoritmo FETI-DP** (DUAL#1) —véase la ecuación (4.36) en la sección (4.2.2)—:

   "Dada $\underline{f}_\Delta \in W_\Delta$, sea $\underline{\lambda}_\Delta \in W_\Delta$ tal que

$$\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\lambda}_\Delta = \underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{f}_\Delta - \left(\underline{u}_p\right)_\Delta\right) \tag{6.32}$$

   sujeto a $\underline{\underline{a}}\underline{\lambda}_\Delta = 0$". O en su forma desarrollada

$$\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\lambda}_\Delta = \underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{f}_\Delta - \underline{\underline{aA}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{f}_\Pi\right) \tag{6.33}$$

   donde una vez calculada $\underline{\lambda}_\Delta$ entonces $\underline{u}_\Delta$ es obtenida mediante

$$\underline{u}_\Delta = \underline{\underline{aS}}^{-1}\left(\underline{f}_\Delta - \underline{\underline{j}}\underline{\lambda}_\Delta\right). \tag{6.34}$$

7. **Formulación Primal Precondicionada del Problema Neumann-Neumann DVS-PRIMAL** (PRIMAL#2) —véase la ecuación (4.43) en la sección (4.2.3)—:

   "Dada $\underline{f}_\Delta \in W_\Delta$, buscar $\underline{v} \in W_\Delta$ tal que

$$\underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{v}_\Delta = \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{f}_\Delta - \left(\underline{u}_p\right)_\Delta\right) \tag{6.35}$$

   sujeto a $\underline{\underline{aS}}\underline{v}_\Delta = 0$". O en su forma desarrollada

$$\underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{v}_\Delta = \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{f}_\Delta - \underline{\underline{aA}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{f}_\Pi\right) \tag{6.36}$$

   donde una vez calculada $\underline{v}_\Delta$ entonces $\underline{u}_\Delta$ es obtenida mediante

$$\underline{u}_\Delta = \underline{\underline{aS}}^{-1}\left(\underline{f}_\Delta - \underline{\underline{j}}\underline{\underline{S}}\underline{v}_\Delta\right). \tag{6.37}$$

8. **Formulación Dual Precondicionada del Problema Neumann-Neumann DVS-DUAL** (DUAL#2) —véase la ecuación (4.50) en la sección (4.2.4)—:

   "Dada $\underline{f}_{\Delta} \in W_{\Delta}$, sea $\underline{\mu}_{\Delta} \in W_{\Delta}$ tal que

$$\underline{\underline{S}}\underline{\underline{a}}\underline{S}^{-1}\underline{\underline{a}}\underline{\mu}_{\Delta} = \underline{\underline{S}}\underline{\underline{a}}\underline{S}^{-1}\underline{\underline{a}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{f}_{\Delta} - \left(\underline{u}_{p}\right)_{\Delta}\right) \tag{6.38}$$

   sujeto a $\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\mu}_{\Delta} = 0$". O en su forma desarrollada

$$\underline{\underline{S}}\underline{\underline{a}}\underline{S}^{-1}\underline{\underline{a}}\underline{\mu}_{\Delta} = \underline{\underline{S}}\underline{\underline{a}}\underline{S}^{-1}\underline{\underline{a}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\left(\underline{f}_{\Delta} - \underline{\underline{a}}\underline{A}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{f}_{\Pi}\right) \tag{6.39}$$

   donde una vez calculada $\underline{\mu}_{\Delta}$ entonces $\underline{u}_{\Delta}$ es obtenida mediante

$$\underline{u}_{\Delta} = \underline{\underline{a}}\underline{S}^{-1}\left(\underline{f}_{\Delta} + \underline{\mu}_{\Delta}\right). \tag{6.40}$$

6.3 Implementación Numérica de DVS

La implementación numérica de por ejemplo, el algoritmo DVS-BDDC (PRIMAL#1) puesto en su forma operacional, Ec.(6.31) es

$$\underline{\underline{a}}\underline{S}^{-1}\underline{\underline{a}}\underline{S}\underline{u}_{\Delta} = \underline{\underline{a}}\underline{S}^{-1}\left(\underline{f}_{\Delta} - \underline{\underline{a}}\underline{A}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{f}_{\Pi}\right) \tag{6.41}$$

en la cual define el sistema lineal virtual a resolver

$$\underline{\underline{M}}\underline{u}_{\Delta} = \underline{b} \tag{6.42}$$

donde

$$\underline{\underline{M}} = \underline{\underline{a}}\underline{S}^{-1}\underline{\underline{a}}\underline{S} \quad \text{y} \quad \underline{b} = \underline{\underline{a}}\underline{S}^{-1}\left(\underline{f}_{\Delta} - \underline{\underline{a}}\underline{A}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{f}_{\Pi}\right) \tag{6.43}$$

entonces el sistema lineal virtual puede ser implementado mediante el método de Gradiente Conjugado o alguna variante de GMRES, dependiendo del tipo de matriz que sea $\underline{\underline{S}}$. Si $\underline{\underline{S}}$ es simétrica y definida positiva, entonces $\underline{\underline{S}}^{-1}$ también será simétrica y definida positiva y por tanto se usaría el método de Gradiente Conjugado, en caso contrario se usaría el método de GMRES o algún otro.

Nótese que, en los métodos iterativos, la adecuada selección de $\underline{u}^0$ puede ser tan costosa —computacionalmente hablando— como encontrar la solución $\underline{u}$, pues tomar una $\underline{u}^0$ no adecuada, generalmente ocasiona realizar más iteraciones para converger en el método que tomar $\underline{u}^0 = 0$.

6.3.1 Implementación para Matrices Simétricas

Suponiendo que $\underline{\underline{S}}$ es simétrica y definida positiva, la formulación $\underline{\underline{M}}\underline{u}_{\Delta} = \underline{b}$ puede ser implementada como el método de Gradiente Conjugado —véase sección (B.2.1)— usando el algoritmo dado en la Ec.(B.13) en el cual se usa el producto interior según corresponda:

- $\underline{\underline{aS}}^{-1}\underline{\underline{aS}}$ será simétrico con respecto al producto interior definido por

$$\langle \underline{u}, \underline{w} \rangle = \underline{u} \cdot \underline{w} \tag{6.44}$$

- $\underline{\underline{Sj}}\underline{\underline{S}}^{-1}\underline{\underline{j}}$ será simétrico con respecto al producto interior definido por

$$\langle \underline{u}, \underline{w} \rangle = \underline{\underline{S}}\underline{u} \cdot \underline{w} \tag{6.45}$$

La implementación del algoritmo se inicia tomando $\underline{u}^0$, una elección común es tomar a $\underline{u}^0 = 0$, si este es el caso, entonces

$$\underline{r}^0 = \underline{\underline{aS}}^{-1}\left(\underline{f}_\Delta - \underline{\underline{aA}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{f}_\Pi\right), \qquad \underline{p}^0 = \underline{r}^0 \tag{6.46}$$

y la parte iterativa[13] queda como:

$$\begin{array}{l}
\text{Para } n = 1, 2, ...\{ \\
\quad \alpha^n = \frac{\langle \underline{p}^n, \underline{\underline{S}}\underline{p}^n \rangle}{\langle \underline{p}^n, \underline{\underline{SaS}}^{-1}\underline{\underline{aS}}\underline{p}^n \rangle} \\
\quad \underline{u}^{n+1} = \underline{u}^n + \alpha^n \underline{p}^n \\
\quad \underline{r}^{n+1} = \underline{r}^n - \alpha^n \underline{\underline{aS}}^{-1}\underline{\underline{aS}}\underline{p}^n \\
\quad < \text{Prueba de convergencia} > \\
\quad \beta^n = \frac{\langle \underline{r}^{n+1}, \underline{\underline{S}}\underline{r}^{n+1} \rangle}{\langle \underline{r}^n, \underline{\underline{S}}\underline{r}^n \rangle} \\
\quad \underline{p}^{n+1} = \underline{r}^{n+1} + \beta^n \underline{p}^n \\
\}
\end{array} \tag{6.47}$$

6.3.2 Implementación para Matrices no Simétricas e Indefinidas

Suponiendo que $\underline{\underline{S}}$ es no simétrica o indefinida, la formulación $\underline{\underline{M}}\underline{u}_\Delta = \underline{b}$ puede ser implementada como el método de GMRES —véase sección (B.2.2)— usando el algoritmo dado en la Ec.(6.31), en cual se inicia tomando $\underline{u}^0$, una elección común es tomar a $\underline{u}^0 = 0$, si este es el caso, entonces

$$\underline{r}^0 = \underline{\underline{aS}}^{-1}\left(\underline{f}_\Delta - \underline{\underline{aA}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{f}_\Pi\right), \beta^0 = \left\|r^0\right\|, \underline{v}^1 = \underline{r}^0/\beta^0 \tag{6.48}$$

y la parte iterativa queda como:

$$\begin{array}{l}
\text{Para } n = 1, 2, ..., \text{Mientras } \beta^n < \tau\beta^0 \ \{ \\
\quad \underline{w}_0^{n+1} = \underline{\underline{aS}}^{-1}\underline{\underline{aS}}\underline{v}^n \\
\quad \text{Para } l = 1 \text{ hasta } n \ \{ \\
\qquad h_{l,n} = \langle \underline{w}_l^{n+1}, \underline{v}^l \rangle \\
\qquad \underline{w}_{l+1}^{n+1} = \underline{w}_l^{n+1} - h_{l,n}\underline{v}^l \\
\quad \} \\
\quad h_{n+1,n} = \left\|\underline{w}_{n+1}^{n+1}\right\| \\
\quad \underline{v}^{n+1} = \underline{w}_{n+1}^{n+1}/h_{n+1,n} \\
\quad \text{Calcular } \underline{y}^n \text{ tal que } \beta n = \left\|\beta^0 \underline{e}_1 - \underline{\underline{\hat{H}}}_n \underline{y}^y\right\| \text{ es mínima} \\
\}
\end{array} \tag{6.49}$$

[13] En este caso se usa el producto interior definido por $\langle \underline{u}, \underline{w} \rangle = \underline{\underline{S}}\underline{u} \cdot \underline{w}$.

donde $\underline{\underline{\hat{H}}}_n = [h_{ij}]_{1\leq i\leq n+1, 1\leq j\leq n}$ y la solución aproximada será $\underline{u}^n = \underline{u}^0 + \underline{\underline{V}}_n \underline{y}^n$, el vector residual será

$$\underline{r}^n = \underline{r}^0 - \underline{\underline{aS}}^{-1} \underline{\underline{aSV}}_k y^n = \underline{\underline{V}}_{n+1} \left(\beta^0 \underline{e}_1 - \underline{\underline{\hat{H}}}_n \underline{y}^n \right). \tag{6.50}$$

6.4 Evaluación de los Operadores Virtuales $\underline{\underline{S}}$ y $\underline{\underline{S}}^{-1}$

Para implementar cualesquiera de los ocho algoritmos desarrollados, sin importar si las matrices involucradas son simétricas o no simétricas; y como se mostró en las implementaciones de los algoritmos en la sección anterior, estos requieren por un lado, la evaluación de $\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1} \underline{f}_\Pi$ y por otro la evaluación de los operadores virtuales $\underline{\underline{S}}^{-1}\underline{v}$ y $\underline{\underline{S}}\underline{v}$, en los tres casos, ninguna de estas matrices es construida pues resultarían matrices densas con el consiguiente costo computacional de su manejo. Para mostrar como hacer su evaluación óptima, primero nótese que el sistema lineal virtual $\underline{\underline{A}}$ a partir del cual se deriva el esquema DVS, está definido como

$$\underline{\underline{A}} = \begin{pmatrix} \underline{\underline{A}}_{\Pi\Pi} & \underline{\underline{A}}_{\Pi\Delta} \\ \underline{\underline{A}}_{\Delta\Pi} & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix} = \begin{pmatrix} \underline{\underline{A}}_{II} & \underline{\underline{A}}_{I\pi} & \underline{\underline{A}}_{I\Delta} \\ \underline{\underline{A}}_{\pi I} & \underline{\underline{A}}_{\pi\pi} & \underline{\underline{A}}_{\pi\Delta} \\ \underline{\underline{A}}_{\Delta I} & \underline{\underline{A}}_{\Delta\pi} & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix} \tag{6.51}$$

donde

$$\begin{aligned} \underline{\underline{A}}_{\Pi\Pi} &= \begin{pmatrix} \underline{\underline{A}}_{II} & \underline{\underline{A}}_{I\pi} \\ \underline{\underline{A}}_{\pi I} & \underline{\underline{A}}_{\pi\pi} \end{pmatrix} \qquad \underline{\underline{A}}_{\Pi\Delta} = \begin{pmatrix} \underline{\underline{A}}_{I\Delta} \\ \underline{\underline{A}}_{\pi\Delta} \end{pmatrix} \\ \underline{\underline{A}}_{\Delta\Pi} &= \begin{pmatrix} \underline{\underline{A}}_{\Delta I} & \underline{\underline{A}}_{\Delta\pi} \end{pmatrix} \end{aligned} \tag{6.52}$$

entonces el operador $\underline{\underline{S}}$ de los nodos duales queda definido por

$$\underline{\underline{S}} = \underline{\underline{A}}_{\Delta\Delta} - \underline{\underline{A}}_{\Delta\Pi} \left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1} \underline{\underline{A}}_{\Pi\Delta} \tag{6.53}$$

y este es formado por $\underline{\underline{S}} = \sum_{\alpha=1}^{E} \underline{\underline{S}}^\alpha$, donde $\underline{\underline{S}}^\alpha$ esta formada por el complemento de Schur local

$$\underline{\underline{S}}^\alpha = \underline{\underline{A}}^\alpha_{\Delta\Delta} - \underline{\underline{A}}^\alpha_{\Delta\Pi} \left(\underline{\underline{A}}^\alpha_{\Pi\Pi}\right)^{-1} \underline{\underline{A}}^\alpha_{\Pi\Delta}. \tag{6.54}$$

En el apéndice A se detallan la forma de realizar todas las operaciones involucradas en los métodos DVS, aquí, bajo el supuesto de que se tienen construidas las matrices locales $\underline{\underline{A}}^i_{II}, \underline{\underline{A}}^i_{I\pi}, \underline{\underline{A}}^i_{I\Delta}, \underline{\underline{A}}^i_{\pi\pi}, \underline{\underline{A}}^i_{\pi\Delta}$ y $\underline{\underline{A}}^i_{\Delta\Delta}$ en cada uno de los subdominios de la partición. Entonces, para resolver $\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1} \underline{u}$, donde $\underline{u} \in W_\Pi$, se puede reescribir como

$$\begin{pmatrix} \underline{w}_I \\ \underline{w}_\pi \end{pmatrix} = \begin{pmatrix} \underline{\underline{A}}_{II} & \underline{\underline{A}}_{I\pi} \\ \underline{\underline{A}}_{\pi I} & \underline{\underline{A}}_{\pi\pi} \end{pmatrix}^{-1} \begin{pmatrix} \underline{u}_I \\ \underline{u}_\pi \end{pmatrix} \tag{6.55}$$

i.e. se necesita resolver $\underline{\underline{A}}_{\Pi\Pi}\underline{w} = \underline{u}$,la cual se expresa como

$$\begin{pmatrix} \underline{\underline{A}}_{II} & \underline{\underline{A}}_{I\pi} \\ \underline{\underline{A}}_{\pi I} & \underline{\underline{A}}_{\pi\pi} \end{pmatrix} \begin{pmatrix} \underline{w}_I \\ \underline{w}_\pi \end{pmatrix} = \begin{pmatrix} \underline{u}_I \\ \underline{u}_\pi \end{pmatrix} \tag{6.56}$$

entonces, $\underline{w}_\pi \in W_\Pi$ es solución de

$$\left(\underline{\underline{A}}_{\pi\pi} - \underline{\underline{A}}_{\pi I}\left(\underline{\underline{A}}_{II}\right)^{-1}\underline{\underline{A}}_{I\pi}\right)\underline{w}_\pi \equiv \underline{\underline{S}}_\pi \underline{w}_\pi = \underline{u}_\pi - \underline{\underline{A}}_{\pi I}\left(\underline{\underline{A}}_{II}\right)^{-1}\underline{u}_I \tag{6.57}$$

mientras

$$\underline{w}_I = \left(\underline{\underline{A}}_{II}\right)^{-1}\left(\underline{u}_I - \underline{\underline{A}}_{I\pi}\underline{w}_\pi\right) \tag{6.58}$$

donde

$$\left(\underline{\underline{A}}_{II}\right)^{-1} = \sum_{i=1}^{N}\left(\underline{\underline{A}}^i_{II}\right)^{-1}. \tag{6.59}$$

Por último, para evaluar $\underline{\underline{S}}\underline{v}$ se aplica el procedimiento similar al detallado en la sección (A.2) y para evaluar $\underline{\underline{S}}^{-1}\underline{v}$ se aplica el procedimiento detallado en la sección (A.3).

De las evaluaciones indicadas en esta sección, una gran parte de ellas es posible realizar en paralelo, y como se mostrará más tarde, la granularidad[14] paralela es gruesa, la cual es ideal para implementarse en equipos de cómputo paralelos como los Clusters, esto se demuestra mediante ejemplos en el capítulo de Análisis de Rendimiento (véase [36], [39] y [38]).

[14] La granularidad de un conjunto de tareas paralelas es la cantidad de trabajo que se puede hacer de forma independiente de otros cálculos.

7 Implementación Computacional de DVS

La implementación computacional de los métodos de descomposición de dominio sin traslape en general (véase [9], [4], [5] y [6]) y de los métodos de descomposición de dominio en el espacio de vectores derivados (DVS) en particular (véase [39] y [38]), en los cuales cada subdominio genera sus matrices locales y el método que resuelve el sistema global virtual —CGM o GMRES— es tal que necesita sólo una porción de la información que generan los subdominios, queda fácilmente estructurado mediante el esquema Maestro-Esclavo, tanto para la implementación del código secuencial como paralela.

El esquema Maestro-Esclavo parece ser un forma óptima[15] de dividir la carga computacional requerida para solucionar un problema de descomposición de dominio sin traslapes, en el cual, uno o más subdominios son asignados a un nodo esclavo, tanto en su implementación secuencial —donde cada nodo esclavo es un objeto— como en su implementación paralela —donde cada nodo esclavo esta asignado a un procesador—, en el cual el nodo maestro de forma síncrona controla las tareas que requiere el esquema DVS, las cuales son llevadas a cabo por los nodos esclavos, donde la comunicación sólo se da entre el nodo maestro y cada nodo esclavo —no existiendo comunicación entre los nodos esclavos—, optimizando así las comunicaciones.

7.1 Esquema Maestro-Esclavo como una Forma de Implementación

El esquema Maestro-Esclavo permite que en los nodos esclavos se definan uno o más subdominios —en los cuales se generen y manipulen las matrices locales de cada subdominio— y que el maestro controle las actividades necesarias para implementar cualquiera de los métodos desarrollados. En particular la implementación del método de resolución del sistema lineal virtual $\underline{\underline{M}}\underline{u}_{\Delta} = \underline{b}$ esquematizados por los algoritmos descritos en la Ec.(6.47 ó 6.49), donde el nodo maestro controlará a cada uno de sus nodos esclavos mediante comunicaciones entre este y los esclavos, pero no entre esclavos, como se muestra en la figura.

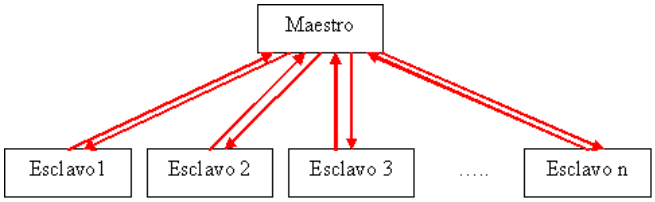

Figura 6: Esquema del Maestro-Esclavo

[15] El esquema Maestro-Esclavo esta intrínseco a la definición de los métodos de descomposición de dominio tipo subestructuración, ya que las tareas implementadas por los subdominios son pensados como procesos esclavos los cuales son controlados por el maestro que implementa la solución de los nodos de frontera interior (véase [39] y [38]).

Esta forma de descomponer el problema dentro del esquema Maestro-Esclavo, permite hacer la implementación del código tanto para la parte secuencial como su paralelización de manera fácil y eficientemente, donde tomando en cuenta la implementación en estrella del Cluster o equipo multiCore, el modelo de paralelismo de MPI y las necesidades propias de comunicación del programa, el nodo maestro tendrá comunicación sólo con cada nodo esclavo, esto reducirá las comunicaciones y optimizará el paso de mensajes (véase [17], [15] y [16]).

Además el esquema de paralelización Maestro-Esclavo permite sincronizar fácilmente por parte del nodo maestro las tareas que realizan en paralelo los nodos esclavos, éste modelo puede ser explotado de manera eficiente si existe poca comunicación entre el maestro y los esclavos; y los tiempos consumidos en realizar las tareas asignadas son mayores que los períodos involucrados en las comunicaciones para la asignación de dichas tareas. De esta manera se garantiza que la mayoría de los procesadores estarán siendo usados de forma eficiente y existirán pocos tiempos muertos, aumentando así la eficiencia global de la implementación de los métodos de descomposición de dominio en el espacio de vectores derivados bajo el esquema Maestro-Esclavo.

7.2 Análisis, Diseño y Programación Orientada a Objetos

Desde el inicio del proyecto para la implementación computacional de los métodos de descomposición de dominio en el espacio de vectores derivados se planteó la necesidad de que el código desarrollado fuera orientado a objetos, que su implementación computacional debería de correr en equipos secuenciales y paralelos para dominios en dos y tres dimensiones. Por ello se optó por usar el lenguaje de programación C++ y la paralelización se haría usando la biblioteca de paso de mensajes MPI.

Dentro de las consideraciones básicas en el análisis orientado a objetos es que el código debería de correr tanto en equipos secuenciales como en paralelos, con un mínimo de cambios y que la interdependencia de la parte paralela no debería afectar la parte secuencial. Para que cualquier cambio en el código de los métodos desarrollados no requiera grandes cambios en el código paralelo. Esto se logra mediante la programación orientada a objetos haciendo uso de clases abstractas[16] o contenedores.

Esto permitió desarrollar un código que fuera robusto y flexible, además de que la escritura, depuración y optimización se hace desde cualquier Notebook y su ejecución puede ser hecha en cualquier computadora personal o Clusters sin ningún cambio en el código.

Por ejemplo, en el uso de los métodos numéricos tanto directos como iterativos para resolver sistemas lineales en los cuales la matriz es real —existe como tal— o es virtual —esta dispersa por los distintos subdominios— se creo

[16] En general las clases abstractas que definen comportamientos virtuales pueden no ser eficientes si son llamadas una gran cantidad de veces durante la ejecución del programa. Para el caso del esquema DVS, en el cual se usa CGM o GMRES para resolver el sistema lineal virtual, este sólo se llama una sola vez; y el proceso de solución del sistema lineal asociado consume la mayoría del tiempo de ejecución, por eso se considera eficiente.

una jerarquía de clases que implementa mediante herencia a la clase abstracta, la cual usan los algoritmos que requerían solucionar un sistema lineal, esta clase abstracta se llama Solvable —véase apéndice B—. La jerarquía[17] de clases mostrada en la figura (7) permite contener a cualquiera de los métodos numéricos de solución de sistemas lineales actuales y cualquier implementación futura y es independiente de si se usa para generar código secuencial o paralelo.

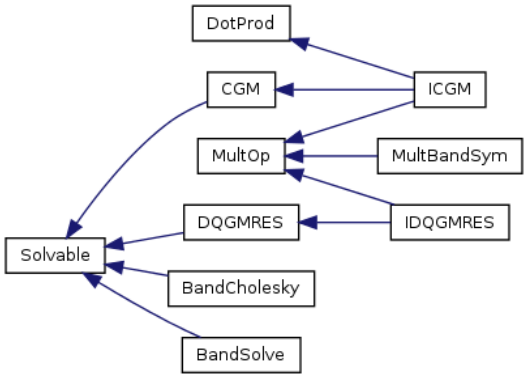

Figura 7: Jerarquía de clases para la implementación de los métodos de resolución de sistemas lineales.

Nótese que, en general, el paradigma de programación orientada a objetos sacrifica algo de eficiencia computacional por requerir mayor manejo de recursos computacionales al momento de la ejecución. Pero en contraste, permite mayor flexibilidad a la hora adaptar los códigos a nuevas especificaciones. Adicionalmente, disminuye notoriamente el tiempo invertido en el mantenimiento y búsqueda de errores dentro del código, además de hacer el código extensible y reutilizable. Esto tiene especial interés cuando se piensa en la cantidad de meses invertidos en la programación comparada con los segundos consumidos en la ejecución del mismo.

7.2.1 Implementación Secuencial en C++

Usando la filosofía del manejo de clases abstractas desde el análisis y durante el diseño de la implementación computacional de los ocho métodos de descomposición de dominio en el espacio de vectores derivados, se pensó en usar una jerarquía de clases que especializarían a una clase abstracta llamada DPMethod, la cual permite implementar uno o más de los métodos de descomposición de dominio desarrollados; y dada una ecuación o sistemas de ecuaciones diferenciales parciales, se usaría el método iterativo —Gradiente Conjugado o el método Residual Mínimo Generalizado o cualquier otro— dependiendo de que la matriz

[17] Las jerarquías de clases de herencia mostradas en las figuras fueron generadas usando Doxygen documentation (véase [63]) a partir del código fuente en C++.

global virtual fueran simétrica o no simétrica; su jerarquía de clases se muestra en la figura (8).

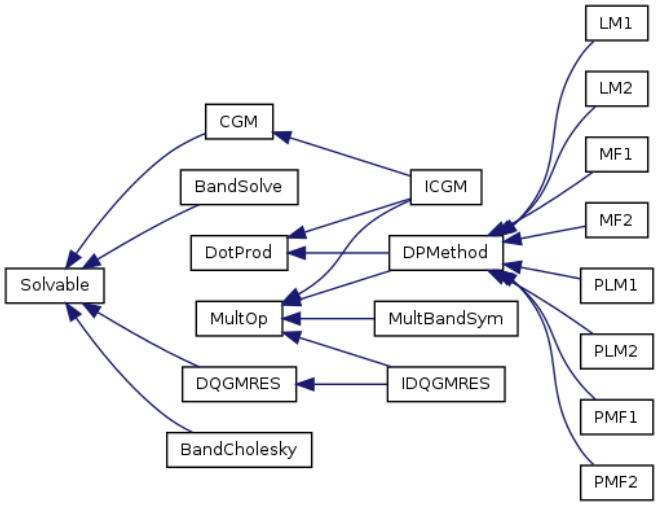

Figura 8: Jerarquía de clases para la implementación secuencial

De esta forma, es posible implementar uno o más de los algoritmos desarrollados de descomposición de dominio en el espacio de vectores derivados sin realizar cambios en la base del código, permitiendo especializar el código para alguna necesidad particular sin cargar con código no requerido en la resolución de un problema específico, pero en caso de evaluar el desempeño de cada uno de los métodos ante un problema determinado, se pueda realizar sin afectación del código.

Además de la flexibilidad anteriormente comentada, también se reutiliza la jerarquía de clases para la resolución de sistemas lineales, permitiendo que cualquier cambio o refinamiento a estas clases redunde en el desempeño global del sistema, permitiendo que en un futuros se agreguen y refinen métodos que manejen con eficiencia la solución de los sistemas lineales asociados al método de descomposición de dominio DVS.

7.2.2 Implementación Paralela en C++ Usando MPI

Para poder intercomunicar al nodo maestro con cada uno de los nodos esclavos se usa la interfaz de paso de mensajes —Message Passing Interface (MPI)—, una biblioteca de comunicación para procesamiento en paralelo. MPI ha sido desarrollado como un estándar para el paso de mensajes y operaciones relacionadas. Este enfoque es adoptado por usuarios e implementadores de bibliotecas, en la cual se proveen a los programas de procesamiento en paralelo de portabilidad y herramientas necesarias para desarrollar aplicaciones que puedan usar el

cómputo paralelo de alto desempeño.

El modelo de paso de mensajes posibilita a un conjunto de procesos —que tienen solo memoria local— la comunicación con otros procesos usando Bus o red, mediante el envío y recepción de mensajes. El paso de mensajes posibilita transferir datos de la memoria local de un proceso a la memoria local de cualquier otro proceso que lo requiera.

En el modelo de paso de mensajes mediante MPI para equipos con uno o más Cores, los procesos se ejecutan en paralelo, teniendo direcciones de memoria separada para cada proceso, la comunicación ocurre cuando una porción de la dirección de memoria de un proceso es copiada mediante el envío de un mensaje dentro de otro proceso en la memoria local mediante la recepción del mismo.

Las operaciones de envío y recepción de mensajes es cooperativa y ocurre sólo cuando el primer proceso ejecuta una operación de envío y el segundo proceso ejecuta una operación de recepción, los argumentos base de estas funciones son:

- Para el que envía, la dirección de los datos a transmitir y el proceso destino al cual los datos se enviarán.

  Send(dir, lg, td, dest, etiq, com)
  $\{dir, lg, td\}$ describe cuántas ocurrencias lg de elementos del tipo de dato td se transmitirán empezando en la dirección de memoria dir; $\{des, etiq, com\}$ describe el identificador etq de destino des asociado con la comunicación com.

- Para el que recibe, debe de tener la dirección de memoria donde se pondrán los datos recibidos, junto con la dirección del proceso del que los envío.

  Recv(dir, mlg, td, fuent, etiq, com, st)
  $\{dir, lg, td\}$ describe cuántas ocurrencias lg de elementos del tipo de dato td se transmitirán empezando en la dirección de memoria dir; $\{fuent, etiq, com, est\}$ describe el identificador etq de la fuente $fuent$ asociado con la comunicación com y el estado st.

El conjunto básico de directivas (en este caso sólo se usan estas) en C++ de MPI son:

MPI::Init	Inicializa al MPI
MPI::COMM_WORLD.Get_size	Busca el número de procesos existentes
MPI::COMM_WORLD.Get_rank	Busca el identificador del proceso
MPI::COMM_WORLD.Send	Envía un mensaje
MPI::COMM_WORLD.Recv	Recibe un mensaje
MPI::Finalize	Termina al MPI

La estructura básica del programa bajo el esquema Maestro-Esclavo codificada en C++ y usando MPI es:

```
main(int argc, char *argv[])
{
      MPI::Init(argc,argv);
      ME_id = MPI::COMM_WORLD.Get_rank();
      MP_np = MPI::COMM_WORLD.Get_size();
      if (ME_id == 0) {
          // Operaciones del Maestro
      } else {
          // Operaciones del esclavo con identificador ME_id
      }
      MPI::Finalize();
}
```

En este único programa se deberá de codificar todas las tareas necesarias para el nodo maestro y cada uno de los nodos esclavos, así como las formas de intercomunicación entre ellos usando como distintivo de los distintos procesos a la variable *ME_id* (véase [15] y [16]).

La jerarquía de clases del esquema Maestro-Esclavo en su implementación paralela permite repartir la carga de varias maneras en uno o más Cores. Reutilizando toda la jerarquía de clases de la implementación secuencial de los algoritmos DVS y sólo es necesario agregar clase que especializa algunos comportamientos que requieren hacer uso de las comunicaciones, mediante la biblioteca de paso de mensajes MPI. La jerarquía de clases es mostrada en la figura siguiente:

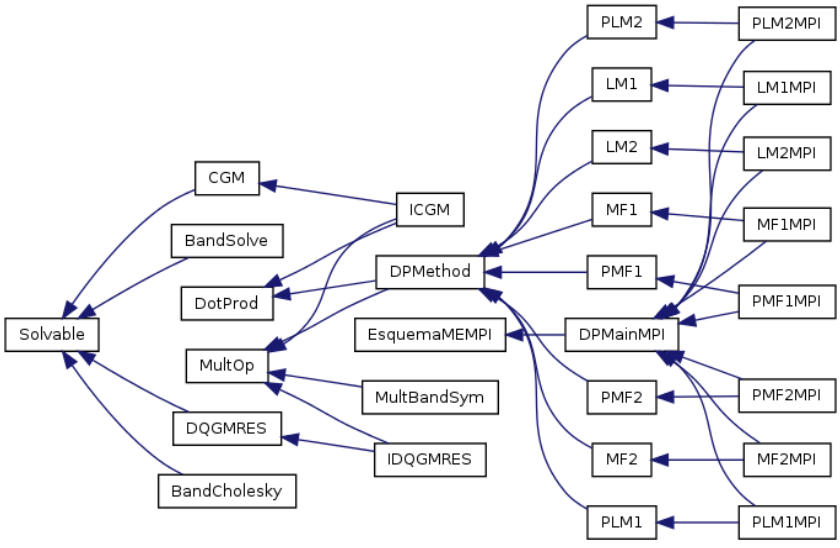

Figura 9: Jerarquía de clases para la implementación paralela rehusando toda la jerarquía de la implementación secuencial y de resolución de sistemas lineales

La reutilización de toda la jerarquía de clases generada para la implementación secuencial permite que el código paralelo soporte una gran cantidad de cambios sin afectación a la implementación paralela, teniendo así, un código robusto, flexible, modular y de fácil mantenimiento (véase [17]).

7.3 Alcances y Limitaciones del Esquema Maestro-Esclavo

El esquema Maestro-Esclavo es eficiente cuando se tiene una carga casi homogénea en cada nodo esclavo y se manejan una cantidad moderada de ellos. Un factor limitante en el esquema Maestro-Esclavo, es que el nodo maestro deberá de atender todas las peticiones hechas por todos y cada uno de los nodos esclavos, esto toma especial relevancia cuando todos o casi todos los nodos esclavos compiten por ser atendidos por el nodo maestro.

Una opción para optimizar el esquema Maestro-Esclavo es contar con un nodo maestro lo suficientemente poderoso para atender simultáneamente la mayor cantidad de las tareas síncronas del método de descomposición de dominio en el menor tiempo posible. Pero los factores limitantes del esquema Maestro-Esclavo son de tres tipos, a saber:

1. Los inherentes al método de descomposición de dominio.
2. Los inherentes al propio esquema Maestro-Esclavo.
3. Los inherentes al equipo de cómputo en paralelo en el que se ejecute el programa.

En el primer caso, en cuanto a los inherentes al método de descomposición de dominio destacan:

- El método de descomposición de dominio es síncrono, es decir, si un nodo esclavo acaba la tarea asignada y avisa al nodo maestro, este no podrá asignarle otra tarea hasta que todos los nodos esclavos concluyan la suya, y se realicen las operaciones necesarias para asignar las nuevas tareas a los nodos esclavos.
- El nodo maestro sólo realiza tareas de control y sincronización pero no conoce o realiza cálculos relacionados con los sistemas lineales locales a cada uno de los subdominios que están asignados a los nodos esclavos.
- Por lo anterior, el esquema Maestro-Esclavo no es eficiente si sólo se usan dos procesos o Cores —uno para el nodo maestro y otro para el nodo esclavo—, por otro lado, cuando se realiza el análisis de rendimiento en P Cores, hay que tomar en cuenta que los únicos nodos que manipulan los sistemas lineales asociados al método de descomposición de dominio son los esclavos $(P-1)$ y el nodo maestro sólo realiza el control y sincronización de las tareas del los métodos DVS.

En el segundo caso, en cuanto a los inherentes al propio esquema Maestro-Esclavo destacan:

- El nodo maestro deberá distribuir las tareas a los nodos esclavos acorde al número de subdominios existentes en la descomposición y la malla fina de cada subdominio, de tal forma que cada nodo esclavo tenga una carga computacional equivalente a los demás nodos esclavos.

- En el caso de una carga homogénea en cada subdominio, si se usan P Cores en el equipo paralelo y la descomposición del dominio tiene E subdominios, tal que $(P-1) \nmid E$, esa descomposición de dominio no es adecuada para trabajar en dicha cantidad de Cores. En este caso, el número de procesadores P que se usen para tener buen balance de cargas es conocido a priori cuando el dominio Ω se descompone en $n \times m$ —$n \times m \times o$— subdominios homogéneos, entonces se generarán $E = n * m$ —$E = n * m * o$— subdominios Ω_α, teniendo un buen balanceo de cargas si $(P-1) \mid E$.

- Pese al buen balanceo de la carga en los nodos esclavos, es común que, un gran número de nodos esclavos envíen simultáneamente datos al nodo maestro saturando su canal de comunicación; y este en algún momento tendrá que tratar atender las múltiples comunicaciones, degradando su rendimiento al aumentar el número de nodos esclavos involucrados en la descomposición.

En el caso de generar desbalance de la carga en los nodos esclavos o una saturación de comunicaciones en el nodo maestro, se propicia a que algunos procesadores terminen antes que otros, generando tiempos muertos de ejecución en dichos Cores; propiciando una notoria degradación en la eficiencia global en el procesamiento, es por esto que, en algunos casos al aumentar el número de procesadores no se aprecia una disminución sustancial del tiempo de ejecución y en casos extremos puede ocasionar un aumento en el tiempo.

En el tercer caso, en cuanto a los inherentes al equipo de cómputo en paralelo en el que se ejecute el programa destacan:

- El programa se diseño para correr en cualquier cantidad de procesos o Cores y no hay límite establecido en cuanto al número de subdominios que soporta el programa, pero el equipo en el que se ejecute tiene un número predeterminado de Cores y cada uno de ellos tiene asignado una cantidad limitada de RAM, es por ello que, las dimensiones del problema que es posible correr en un equipo paralelo dado esta determinado por estas limitantes.

- En los equipos paralelos, el cuello de botella en cuanto a la eficiencia global de la ejecución, lo presentan las comunicaciones, entre más comunicaciones necesite el programa, es imperante el contar con una infraestructura que permita la mejor velocidad de comunicaciones entre el nodo maestro y los nodos esclavos; además de que esta cuente con la menor latencia posible

en las comunicaciones. Por otro lado, el acceso al disco duro es mínimo y no representa un costo significativo en las comunicaciones totales de la ejecución.

Para ejemplificar lo discutido anteriormente, se considera como modelo matemático el problema de valor en la frontera (BVP) asociado con la ecuación de Poisson, con condiciones de frontera Dirichlet, definido en Ω como:

$$\begin{aligned} -\nabla^2 u &= f_\Omega \text{ en } \Omega \\ u &= g_{\partial\Omega} \text{ en } \partial\Omega \end{aligned} \tag{7.1}$$

este ejemplo, gobierna los modelos de muchos sistemas de la ingeniería y de la ciencia, entre ellos el flujo de agua subterránea a través de un acuífero isotrópico, homogéneo bajo condiciones de equilibrio y es muy usado en múltiples ramas de la física, por ejemplo, gobierna la ecuación de la conducción de calor en un sólido bajo condiciones de equilibrio.

En particular se considera el problema con Ω definido en:

$$\Omega = [-1, -1] \times [1, 1] \tag{7.2}$$

donde

$$f_\Omega = 2n^2\pi^2 \sin(n\pi x) * \sin(n\pi y) \quad \text{y} \quad g_{\partial\Omega} = 0 \tag{7.3}$$

cuya solución es

$$u(x, y) = \sin(n\pi x) * \sin(n\pi y) \tag{7.4}$$

Por ejemplo para $n = 4$, la solución es $u(x, y) = \sin(4\pi x) * \sin(4\pi y)$, cuya gráfica se muestra a continuación:

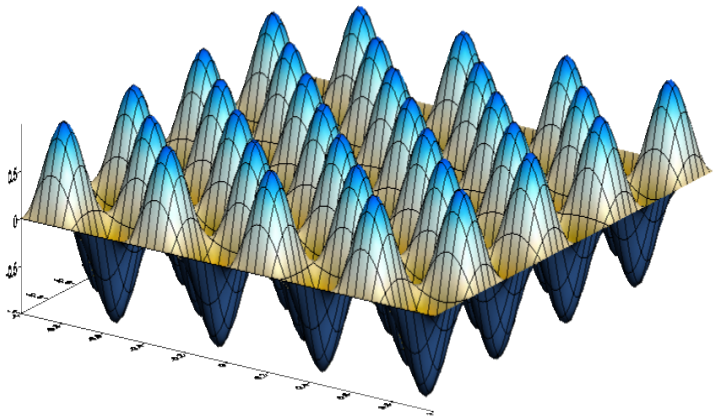

Figura 10: Solución a la ecuación de Poisson para n=4.

Para las pruebas de rendimiento[18] en las cuales se evalúa el desempeño de los programas realizados se usa $n = 100$, pero es posible hacerlo con $n \in \mathbb{N}$ grande.

Supóngase que se desea resolver el dominio Ω usando 1024×1024 nodos —$1,048,576$ grados de libertad— mediante el algoritmo precondicionado PRIMAL#2 , de manera inmediata surgen las siguientes preguntas: ¿cuáles son las posibles descomposiciones posibles? y ¿en cuántos procesadores se pueden resolver cada descomposición?. Para este ejemplo en particular, sin hacer la tabla exhaustiva, se tiene:

Partición	Subdominios	Procesadores
2x2 y 512x512	4	2,3,5
4x4 y 256x256	16	2,3,5,9,17
8x8 y 128x128	64	2,3,5,9,17,33,65
16x16 y 64x64	256	2,3,5,9,17,33,65,129,257
32x32 y 32x32	1024	2,3,5,9,17,33,65,129,..,1025
64x64 y 16x16	4096	2,3,5,9,17,33,65,129,...,4097
128x128 y 8x8	16384	2,3,5,9,17,33,65,129,...,16385
256x256 y 4x4	65536	2,3,5,9,17,33,65,129,...,65537
512x512 y 2x2	262144	2,3,5,9,17,33,65,129,...,262145

De esta tabla es posible seleccionar las descomposiciones que se adecuen a las necesidades particulares del equipo paralelo con que se cuente, para evaluar el tiempo de ejecución de este ejemplo se usó la PC Antipolis Intel Xeon a 2.33 GHtz de 64 bits con 8 Cores y 32 GB de RAM, obteniendo los siguientes resultados para una tolerancia de 10^{-6} usando norma infinita en todos los casos (tiempo en segundos):

	1 Core	2 Cores	3 Cores	4 Cores	5 Cores	6 Cores	7 Cores	8 Cores
Partición	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo
2x2 y 512x512	16465	10659	7207	7105	4641			
4x4 y 256x256	2251	5063	2252	2103	1643	1233	1068	947
8x8 y 128x128	855	885	482	395	314	311	283	272
16x16 y 64x64	321	348	190	149	121	125	118	117
32x32 y 32x32	26	39	26	24	23	21	21	21
64x64 y 16x16	205	595	485	477	481	461	469	469
128x128 y 8x8	1026	5453	5352	5431	5633	5843	5843	5903
256x256 y 4x4	8544	26167	25892	25902	25939	25950	25969	26003
512x512 y 2x2	34845	64230	63293	63308	63389	63475	63502	63693

[18] En todos los ejemplos del presente trabajo no se realiza una análisis de comunicación de forma independiente al tiempo de cálculo, ya que los Clusters a los que se obtuvo acceso carecen de herramientas que permitan realizar dicho análisis, pero si se realizaron algunas pruebas con XMPI (véase [64]) y Vampir (véase [65]) para algunos ejemplos representativos en los cuales se muestra que se tiene granularidad gruesa (véase [39]).

De estos resultados, se desprende que:

1. Dependiendo del tamaño de la malla gruesa —número de subdominios a trabajar— y de la malla fina, es siempre posible encontrar una descomposición de dominio —32×32 y 32×32— en que el tiempo de cálculo sea mínimo:

   - Al usar un solo Core (programa secuencial 26 seg.).
   - Al usar múltiples Cores interconectados mediante MPI (6 Cores en 21 seg.).

2. Es notorio el efecto que genera el mal balanceo de carga[19], el cual se refleja en que no disminuye el tiempo de ejecución al aumentar el número de procesadores y en algunos casos el tiempo aumenta conforme se agregan más Cores.

En contraste con los 110 segundos en que se resolvió el mismo problema usando los métodos de Elemento Finito y Diferencias Finitas, usando en ambos casos Factorización Cholesky para resolver el sistema lineal asociado.

7.4 Afectación del Rendimiento al Refinar la Descomposición

Una parte fundamental al trabajar con problemas reales usando una descomposición fina es conocer a priori que factores afectan el rendimiento de la aplicación ante las posibles elecciones en la descomposición de dominio, la afectación se da por:

1. En el caso de contar con un gran número de subdominios que estén asignados a distintos nodos esclavos, la afectación se da por la saturación del nodo maestro con una gran cantidad de comunicaciones simultáneas por parte de los nodos esclavos que el nodo maestro deberá de atender y la velocidad de comunicación del canal usado para ello. Esto es especialmente importante en la implementación paralela en la cual la interconexión del equipo paralelo se hace mediante un canal de comunicación lento u ocupado por otros procesos.

2. En el caso de realizar una descomposición muy fina en cada subdominio, la afectación del rendimiento se da al aumentar el número de nodos involucrados en el complemento de Schur local $\underline{\underline{S}}^i$, ya que esto significa, por un lado generar matrices locales más grandes

$$\underline{\underline{A}}^{\alpha}_{II}, \underline{\underline{A}}^{\alpha}_{I\pi}, \underline{\underline{A}}^{\alpha}_{I\Delta}, \underline{\underline{A}}^{\alpha}_{\pi I}, \underline{\underline{A}}^{\alpha}_{\pi\pi}, \underline{\underline{A}}^{\alpha}_{\pi\Delta}, \underline{\underline{A}}^{\alpha}_{\Delta I}, \underline{\underline{A}}^{\alpha}_{\Delta\pi} \text{ y } \underline{\underline{A}}^{\alpha}_{\Delta\Delta} \tag{7.5}$$

[19] Por ejemplo, al usar una descomposición gruesa de $64 \times 64 = 4096$ subdominios repartidos en $3, 5, 6, 7$ nodos esclavos.

además de resolver el sistema $y = \left(\underline{\underline{A}}^i_{II}\right)^{-1} x$ de alguna forma. Si el número de nodos interiores en el subdominio es grande entonces solucionar el complemento de Schur local será costoso computacionalmente.

Para el primer caso, el uso de algunos cientos o miles de subdominios no afectan de manera considerable el desempeño del Esquema Maestro-Esclavo si la red es relativamente rápida (de un Gigabit por segundo o más), y como los avances en las comunicaciones son vertiginosos, en un corto tiempo se tendrá acceso a redes de mayor velocidad reduciendo el efecto de manipular un gran número de subdominios simultáneamente.

Para el segundo caso, al resolver el complemento de Schur local, se puede emplear diversos métodos de solución, la selección del método más adecuado al problema en particular depende por un lado de las capacidades computacionales del equipo donde se implemente la ejecución y las características propias de los sistemas lineales asociados al problema. Así, para solucionar el sistema $y = \left(\underline{\underline{A}}^i_{II}\right)^{-1} x$ correspondiente al complemento de Schur local $\underline{\underline{S}}^i$ se puede usar por ejemplo: Factorización LU, Factorización Cholesky, Gradiente Conjugado o alguna variante de GMRES, pero deberá de usarse aquel método que proporcione la mayor velocidad en el cálculo o que consuma la menor cantidad de memoria —ambas condicionantes son mutuamente excluyentes—, por ello la decisión de que método usar deberá de tomarse al momento de tener que resolver un problema particular en un equipo dado y básicamente el condicionante es el tamaño del la matriz $\underline{\underline{A}}^i_{II}$ versus el método numérico usado para resolver el sistema lineal asociado —todas esas opciones están implementadas en el código y pueden seleccionarse conforme sean requeridos en la ejecución, mediante directivas de compilación—

Por lo visto en el ejemplo anterior, si el problema involucra una gran cantidad de nodos interiores y el equipo —secuencial o paralelo— en el que se implantará la ejecución del programa tiene una cantidad de memoria reducida, es recomendable en los procesos locales a los subdominios usar métodos iterativos —Gradiente Conjugado o alguna variante de GMRES—, estos consume una cantidad de memoria pequeña comparada con los métodos directos —Factorización LU o Cholesky— pero requieren una gran cantidad de iteraciones para obtener la misma precisión que los directos.

Hay que tomar en cuenta que al aumentar el número de subdominios en una descomposición particular, se garantiza que las matrices a generar y calcular sean cada vez más pequeñas y fáciles de manejar. Pero hay un límite al aumento del número de subdominio y disminución del tamaño de las matrices a generar por subdominio; y esto se refleja en una pérdida de eficiencia en el tiempo de ejecución, esto es generado por la gran cantidad de subdominios que es necesario crear y manejar por el nodo maestro, incrementando sustancialmente las comunicaciones y por otro lado, cada subdominio manejará cada vez matrices más pequeñas con el consecuente aumento de los tiempos muertos, al invertir mucho más tiempo en comunicaciones que en cálculos.

Para mitigar los factores limitantes inherente al propio esquema Maestro-Esclavo, es posible implementar algunas operaciones del nodo maestro en paralelo, usando uno o más Cores distintos a los asignados a los nodos esclavos. Para la parte inherente al método de descomposición de dominio, la parte medular la da el balanceo de cargas. Es decir, cada nodo esclavo debe tener una carga de trabajo equivalente al resto de los nodos.

Tomando en cuenta lo discutido, para un problema particular y la descomposición del dominio Ω en la implementación paralela, hay que tomar en cuenta lo siguiente:

- Buscar que la descomposición de malla gruesa y su asociada malla fina, en la que cada nodo esclavo —asociado a un procesador— tenga una carga casi homogénea con respecto a los demás nodos esclavos, .i.e. buscar que en su conjunto, todos los subdominios Ω_α de la malla gruesa y su descomposición fina de cada uno de ellos, que estén asignados a cada nodo esclavo sean computacionalmente equivalentes.

- Elegir el método numérico local a cada subdominio para garantizar el uso de la menor cantidad de memoria posible y/o la mayor velocidad de ejecución versus la precisión global esperada del método de descomposición de dominio.

- Elegir de las distintas descomposiciones balanceadas del dominio Ω y las diferentes opciones de los métodos numéricos usados localmente en cada subdominio, aquella que presente el mejor rendimiento computacional acorde al equipo paralelo en el cual se implemente la solución del problema.

Nótese que el esquema Maestro-Esclavo paralelo lanza P procesos —uno para el nodo maestro y $P-1$ para los nodos esclavos—, estos en principio corren en un solo procesador pero pueden ser lanzados en múltiples procesadores usando una directiva de ejecución, de esta manera es posible que en una sola máquina se programe, depure y sea puesto a punto el código usando mallas relativamente pequeñas —del orden de miles o millones de nodos— y cuando este listo para producción se puede mandar a cualquier equipo paralelo sin cambio alguno en el código.

7.5 Opciones para Soportar una Descomposición Fina del Dominio

Supóngase ahora que se necesita resolver el problema de una descomposición fina del dominio Ω, sin pérdida de generalidad, se puede suponer por ejemplo, que se usa una malla de 8192×8192 nodos, este tipo de problemas es común y surgen cotidianamente en la resolución de sistemas reales y las opciones para implantarlo en un equipo paralelo son viables, existen y son actualmente usadas. Aquí las opciones de partición del dominio son muchas y variadas, y la variante seleccionada dependerá fuertemente de las características del equipo de

cómputo paralelo del que se disponga. Si se supone que una descomposición de 100×100 nodos en un subdominio consume 1 GB de RAM y que el consumo de memoria crece linealmente con el número de nodos, entonces algunas posibles descomposiciones son:

Procesadores	Descomposición	Nodos Subdominio	RAM Mínimo
5	2×2 y 4096×4096	4096×4096	≈40.0 GB
257	16×16 y 512×512	512×512	≈5.0 GB
1025	32×32 y 256×256	256×256	≈2.5 GB
4097	64×64 y 128×128	128×128	≈1.2 GB

Nótese que para las primeras particiones, el consumo de RAM es excesivo y en las últimas particiones la cantidad de procesadores en paralelo necesarios es grande —pero ya de uso común en nuestros días—. Como en general, contar con equipos paralelos de ese tamaño es en extremo difícil, ¿es posible resolver este tipo de problemas con una cantidad de procesadores fijo menor al sugerido y donde cada uno de ellos tiene solo memoria suficiente para soportar uno o más subdominios?, la respuesta es si.

Primero, nótese que al considerar una descomposición fina del tipo 64×64 y 128×128 se requiere aproximadamente 1.2 GB de RAM por Core, si además se supone que sólo se tienen unos cuantos procesadores con poca memoria —por ejemplo 2 GB—, entonces no es posible tener en memoria de manera conjunta a las matrices generadas por el método.

Una de las grandes ventajas de los métodos de descomposición de domino es que los subdominios son en principio independientes entre si y que sólo están acoplados a través de la solución en la interfase de los subdominios que es desconocida.

Como sólo se requiere tener en memoria la información de la frontera interior, es posible bajar a disco duro todas las matrices y datos complementarios generados en cada subdominio —que consumen el 99% de la memoria del objeto *RectSub*—, que no se requieran en ese instante para la operación del esquema Maestro-Esclavo.

Recuperando del disco duro solamente los datos del subdominio a usarse en ese momento —ya que el proceso realizado por el nodo maestro es secuencial— y manteniéndolos en memoria por el tiempo mínimo necesario. Así, es posible resolver un problema de una descomposición fina, usando una cantidad de procesadores fija y con una cantidad de memoria reducida por procesador.

En un caso extremo, la implementación para resolver un dominio Ω descompuesto en un número de nodos grande es posible implementarla usando sólo dos procesos en un procesador, uno para el proceso maestro y otro para el proceso esclavo, en donde el proceso esclavo construiría las matrices necesarias por cada subdominio y las guardaría en disco duro, recuperándolas conforme el proceso del nodo maestro lo requiera. Nótese que la descomposición del domino Ω estará sujeta a que cada subdominio Ω_i sea soportado en memoria conjuntamente con los procesos Maestro y Esclavo.

De esta forma es posible resolver un problema de gran envergadura usando recursos computacionales limitados, sacrificando velocidad de procesamiento en

aras de poder resolver el problema. Está es una de las grandes ventajas de los métodos de descomposición de dominio con respecto a los otros métodos de discretización tipo Diferencias Finitas y Elemento Finito.

El ejemplo anterior da una buena idea de las limitantes que existen en la resolución de problemas con dominios que tienen una descomposición fina y pone de manifiesto las características mínimas necesarias del equipo paralelo para soportar dicha implantación.

7.6 Otras Opciones de Paralelización

En la actualidad, casi todos los equipos de cómputo usados en estaciones de trabajo y Clusters cuentan con dos o más Cores, en ellos siempre es posible usar MPI para intercambiar mensajes entre procesos corriendo en el mismo equipo de cómputo, pero no es un proceso tan eficiente como se puede querer. En estas arquitecturas llamadas de memoria compartida es mejor usar OpenMP o cualquiera de sus variantes para trabajar en paralelo. Por otro lado es ya común contar con las cada vez más omnipresentes tarjetas NVIDIA, con los cada vez más numerosos Cores CUDA —que una sola tarjeta NVIDIA TESLA puede tener del orden de cientos de ellos— y que en un futuro serán cada vez más numerosos.

Para lograr obtener la mayor eficiencia posible de estos tres niveles de paralelización, se están implementando procesos híbridos (véase [61] y [62]), en donde la intercomunicación de equipos con memoria compartida se realiza mediante MPI y la intercomunicación entre Cores que comparten la misma memoria se realiza con OpenMP, además las operaciones matriciales se le encargan a los numerosos Cores CUDA de las tarjetas NVIDIA.

Los métodos de descomposición de dominio sin traslape y en particular el esquema DVS con sus ocho algoritmos, pueden hacer uso de esta forma integradora de paralelismo. Para ello, la interconexión de equipos de memoria compartida se realizaría mediante MPI y en cada equipo de memoria compartida se manipularían uno o más subdominios mediante OpenMP —ya que cada subdominio es independiente de los demás— y la manipulación de matrices y operaciones entre matrices y vectores que requiere cada subdominio se realizarían en las tarjetas NVIDIA mediante los numerosos Cores CUDA sin salir a la RAM de la computadora.

Para integrar esta forma de paralelismo en los códigos, es necesario hacer cambios mínimos[20] al mismo, ya que sólo es necesario reimplementar los comportamientos locales que requieren otro tipo de paralelismo, ya que la jerarquía de clases del código desarrollado permite especializar los comportamientos que implementan las comunicaciones, esto queda de manifiesto al reutilizar toda la jerarquía de clases de la implementación secuencial en la implementación paralela como se aprecia en la figura (9).

[20] Ya se tiene una versión operacional del código en los cuales se han realizado algunas pruebas de rendimiento, pero los resultados son limitados por la complejidad de la programación de las tarjetas NVIDIA y la falta de herramientas y bibliotecas de código abierto que optimicen y depuren las implementaciones.

Además, el esquema Maestro-Esclavo sólo requiere enviar un vector a cada subdominio en cada paso de la iteración del sistema lineal virtual —mediante el paso de mensajes usando MPI— el cual se coloca en la RAM de la memoria compartida, después este es copiado[21] a la RAM de la tarjeta NVIDIA según el subdominio que se este trabajando —se controla usando OpenMP—, aquí los múltiples Cores CUDA sin salir de su RAM local efectuarían las operaciones de multiplicación de matriz vector necesarias para regresar un único vector a la RAM de la memoria compartida y de ahí se enviaría por MPI al nodo maestro, concluyendo la iteración.

Permitiendo así, tener una creciente eficiencia de paralelización que optimizan en gran medida los recursos computacionales, ya que todas las matrices y vectores se generarían en la RAM de la tarjeta NVIDIA, véase figura (11).

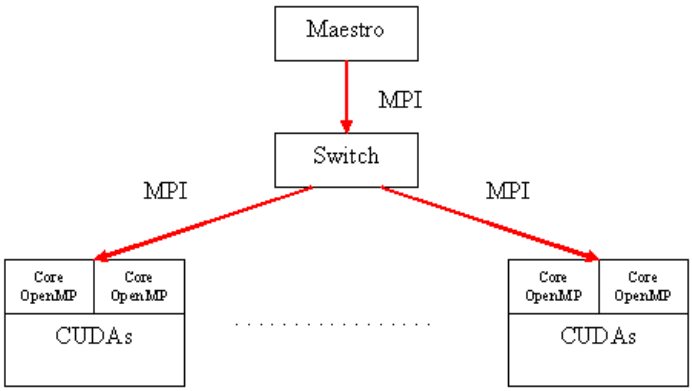

Figura 11: La intercomunicación de equipos con memoria compartida se realiza mediante MPI y la intercomunicación entre Cores que comparten la misma memoria se realiza con OpenMP, además las operaciones matriciales se le encargan a los numerosos Cores CUDA de las tarjetas NVIDIA.

De esta manera es posible adaptar el código para todos y cada uno de los métodos desarrollados, de forma tal que sea reutilizable y que pueda usarse en problemas en los que el número de grados de libertad sea grande, permitiendo hacer uso de equipos de cómputo cada vez más asequibles y de menor costo, pero con una creciente eficiencia computacional que podrán competir en un futuro con los grandes equipos de cómputo de alto desempeño.

[21] En tránsito de datos entre la RAM de la computadora y la RAM de los CUDAs, no es tan rápido como se requiere. Esto genera una baja de rendimiento considerable, que en ciertos problemas como los no lineales y con coeficientes variables es notorio.

8 Análisis y Discusión de Resultados

La uniformidad de las fórmulas presentadas en la sección (6.2) para los métodos de descomposición de dominio en el espacio de vectores derivados (DVS), nos han permitido implementar de forma eficiente dichos algoritmos en múltiples equipos secuenciales y paralelos, aprovechando el hecho que los algoritmos derivados son capaces de obtener *la solución global por la resolución de problemas locales exclusivamente*.

El grupo de ocho algoritmos desarrollados —de los cuales cuatro son precondicionados— a los que nos referiremos como los *algoritmos DVS* (véase [36], [38]), operan exclusivamente sobre los nodos primales y duales en la frontera interior y estos se implementan eficientemente mediante el uso de métodos ite-rativos — CGM para el caso simétrico o GMRES para el caso no simétrico— para resolver el sistema algebraico virtual asociado.

En esta sección, se mostrará —mediante los resultados de algunos experimentos numéricos conspicuos en Ciencias de la Tierra e Ingeniería— la eficiencia del esquema DVS (véase [39]), para ello; primero, se muestra el rendimiento en problemas simétrico y no simétricos —en dos y tres dimensiones—; segundo, se muestra el rendimiento en problemas indefinidos; tercero, se muestra el rendimiento en problemas de Advección-Difusión; después, se muestra el análisis de rendimiento en equipos paralelos hasta con 1024 Cores y por último se muestra lo que se consideran como criterios integrales para evaluar métodos de descomposición de dominio sin traslape y en especial al esquema DVS.

Para los experimentos numéricos reportados en esta sección, sólo se muestran los resultados de los cuatro métodos precondicionados del esquema DVS; en donde el dominio Ω fue discretizado usando una malla estructurada uniforme. Y en todos los casos, se tomaron como nodos primales a los vértices de los subdominios de la partición gruesa en dos dimensiones y a las aristas de los subdominios de la partición gruesa para tres dimensiones. Además, la tolerancia usada para concluir los métodos iterativos de resolución de sistemas lineal virtual asociado es en norma infinita.

8.1 Análisis de Rendimiento para Problemas Simétricos y no Simétricos

Para realizar el análisis de rendimiento, se usa la ecuación elíptica

$$-a\nabla^2 \underline{u} + \underline{b} \cdot \nabla \underline{u} + c\underline{u} = f \tag{8.1}$$

con condiciones de frontera Dirichlet cero, donde $a, c > 0$ son constantes, mientras que $\underline{b}$ es un vector constante de dimensión n. El dominio $\Omega \subset \mathbb{R}^n$ fue tomado con $n = 2, 3$ donde Ω es el cuadrado o cubo unitario según corresponda.

Las matrices generadas fueron obtenidas por discretización local en ambos casos —en dos y tres dimensiones— del problema con valores en la frontera descrito anteriormente, donde $a = 1$. La elección $\underline{b} = (1, 1)$ y $\underline{b} = (1, 1, 1)$ con $c = 0$ generan matrices no simétricas, escogiendo $c = 1$ y $\underline{b} = 0$ se obtienen

matrices simétricas las cuales son usadas con propósitos de comparación. En todos los ejemplos se usa una tolerancia de $1e-6$.

Problemas en Dos Dimensiones En la primera tabla se muestra la descomposición usada, los grados de libertad asociados al sistema y el número de vértices primales usados:

Ejemplo	Partición	Subdominios	Grados Libertad	Primales
1	2×2 y 2×2	4	9	1
2	4×4 y 4×4	16	225	9
3	6×6 y 6×6	36	1,225	25
4	8×8 y 8×8	64	3,969	49
5	10×10 y 10×10	100	9,801	81
6	12×12 y 12×12	144	20,449	121
7	14×14 y 14×14	196	38,025	169
8	16×16 y 16×16	256	65,025	225
9	18×18 y 18×18	324	104,329	289
10	20×20 y 20×20	400	159,201	361
11	22×22 y 22×22	484	233,289	441
12	24×24 y 24×24	576	330,625	529
13	26×26 y 26×26	676	455,625	625
14	28×29 y 28×28	784	613,089	729
15	30×30 y 30×30	900	808,201	841

En la segunda tabla se muestra el número de iteraciones requeridas para satisfacer la tolerancia solicitada al método CGM para el caso simétrico:

Ejemplo	PRIMAL#1	PRIMAL#1	DUAL#1	DUAL#2
1	2	1	2	1
2	7	7	6	5
3	9	9	7	6
4	10	10	9	7
5	11	11	10	8
6	12	11	13	9
7	12	12	13	12
8	13	12	14	12
9	13	13	15	13
10	13	13	15	14
11	13	14	15	16
12	14	14	15	15
13	14	14	15	15
14	14	14	15	15
15	15	14	15	15

En la tercer tabla se muestra el número de iteraciones requeridas para satisfacer la tolerancia solicitada al método GMRES para el caso no simétrico:

Ejemplo	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
1	2	1	2	1
2	8	6	6	6
3	10	8	8	8
4	12	10	9	9
5	13	12	9	10
6	14	12	10	10
7	15	13	11	11
8	15	14	11	11
9	16	14	11	12
10	16	15	12	12
11	17	16	12	12
12	17	16	12	13
13	17	16	13	13
14	18	17	13	13
15	18	17	13	13

Cuando la eficiencia de los algoritmos para matrices simétricas y no simétricas son comparadas, se observa que el número de iteraciones son del mismo orden y comparables con los resultados para este mismo tipo de problemas (véase [34]).

Problemas en Tres Dimensiones En la primera tabla se muestra la descomposición usada, los grados de libertad asociados al sistema y el número de vértices primales usados:

Ejemplo	Partición	Subdominios	Grados Libertad	Primales
1	$2 \times 2 \times 2$ y $2 \times 2 \times 2$	8	27	7
2	$3 \times 3 \times 3$ y $3 \times 3 \times 3$	27	512	80
3	$4 \times 4 \times 4$ y $4 \times 4 \times 4$	64	3375	351
4	$5 \times 5 \times 5$ y $5 \times 5 \times 5$	125	13824	1024
5	$6 \times 6 \times 6$ y $6 \times 6 \times 6$	216	42875	2375
6	$7 \times 7 \times 7$ y $7 \times 7 \times 7$	343	110592	4752
7	$8 \times 8 \times 8$ y $8 \times 8 \times 8$	512	250047	8575
8	$9 \times 9 \times 9$ y $9 \times 9 \times 9$	729	512000	14336
9	$10 \times 10 \times 10$ y $10 \times 10 \times 10$	1000	970299	22599

En la segunda tabla se muestra el número de iteraciones requeridas para satisfacer la tolerancia solicitada al método CGM para el caso simétrico:

Ejemplo	PRIMAL#1	PRIMAL#1	DUAL#1	DUAL#2
1	2	2	2	2
2	4	4	3	3
3	5	5	4	3
4	6	5	4	3
5	6	6	4	4
6	7	6	4	4
7	8	7	5	6
8	8	8	7	7
9	8	8	8	8

En la tercer tabla se muestra el número de iteraciones requeridas para satisfacer la tolerancia solicitada al método GMRES para el caso no simétrico:

Ejemplo	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
1	3	2	2	2
2	6	4	4	4
3	7	6	5	5
4	8	7	5	5
5	10	7	6	6
6	11	8	6	6
7	11	9	7	7
8	12	10	8	8
9	13	11	9	9

Cuando la eficiencia de los algoritmos para matrices simétricas y no simétricas son comparadas, se observa que el número de iteraciones son del mismo orden y comparables con los resultados para este mismo tipo de problemas (véase [34]).

De estos resultados, se puede concluir que los métodos de descomposición de dominio en el espacio de vectores derivados desarrollados tanto para tratar problemas simétricos y no simétricos en experimentos numéricos en dos y tres dimensiones presentan una eficiencia del mismo orden (véase [34] y [19]). Además, el desarrollo de los códigos se simplifica, al poder tratar con problemas simétricos y no simétricos en un mismo código.

8.2 Análisis de Rendimiento para Problemas Indefinidos

Para los problemas indefinidos, como es en el caso de la ecuación de Helmholtz, interesa encontrar una malla —lo más gruesa posible— en la cual el problema sea soluble sin obtener un error considerable para valores grandes de k, que normalmente generan inestabilidad numérica en los métodos de discretización, las cuales siempre se eliminan al refinar adecuadamente la malla, la ecuación utilizada es:

$$-\Delta u - k^2 u = f \tag{8.2}$$

en este caso, la discretización se realizo mediante el método de Diferencias Finitas centradas, los ejemplos se resolvieron mediante el método de GMRES con una tolerancia de 10^{-6}, con fines de ejemplificación, aquí mostramos los resultados para $k = 10$.

Para el primer ejemplo, la ecuación utilizada es en 2D; $(x, y) \in [-1, 1] \times [-1, 1]$, donde $u(x, y) = 0$ sobre $\partial\Omega$, los resultados obtenidos para las distintas descomposiciones de dominio, se muestran en la siguiente tabla:

Partición	Grados de Libertad	Primales	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
6×6 y 6×6	1225	25	8	8	8	7
10×10 y 10×10	9801	81	16	13	16	13
14×14 y 14×14	38025	169	18	15	18	15
18×18 y 18×18	104329	289	21	16	20	16
22×22 y 22×22	233289	441	20	17	21	16
26×26 y 26×26	455625	625	21	17	20	17
30×30 y 30×30	808201	841	26	18	21	17

Para el segundo ejemplo, la ecuación utilizada es en 3D; $(x, y, z) \in [-1, 1] \times [-1, 1] \times [-1, 1]$, donde $u(x, y, z) = 0$ sobre $\partial\Omega$, los resultados obtenidos para las distintas descomposiciones de dominio, se muestran en la siguiente tabla:

Partición	Grados de Libertad	Primales	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
$2\times2\times2$ y $2\times2\times2$	27	7	1	1	1	1
$3\times3\times3$ y $3\times3\times3$	512	80	4	4	4	3
$4\times4\times4$ y $4\times4\times4$	3375	351	5	4	4	3
$5\times5\times5$ y $5\times5\times5$	13824	1024	6	6	5	5
$6\times6\times6$ y $6\times6\times6$	42875	2375	7	7	6	5
$7\times7\times7$ y $7\times7\times7$	110592	4752	7	7	6	5
$8\times8\times8$ y $8\times8\times8$	250047	8575	8	8	6	5
$9\times9\times9$ y $9\times9\times9$	512000	14336	8	8	6	6
$10\times10\times10$ y $10\times10\times10$	970299	22599	9	6	6	6

De los resultados mostrados en esta sección, se puede concluir que el esquema de descomposición de dominio en el espacio de vectores derivados para problemas indefinidos presenta buenos resultados para distintos valores de k sin mostrar significativas inestabilidades numéricas en mallas burdas.

Además, haciendo los ajustes pertinentes al esquema de discretización de diferencias finitas —sin hacer cambio alguno al esquema DVS—, es posible resolver la ecuación de Helmholtz tal que no se introduzca error de truncamiento, consecuentemente se puede calcular la solución numérica exacta para la ecuación de Helmholtz para cualquier número de onda sin usar una malla fina (véase [60]).

8.3 Análisis de Rendimiento para Problemas de Advección-Difusión

En el caso de los problemas de Advección-Difusión interesa encontrar una malla —lo más gruesa posible— en la cual el problema sea soluble sin obtener un error considerable al usar valores de viscosidad pequeños que normalmente generan inestabilidad numérica en los métodos de discretización, las cuales siempre se eliminan al refinar adecuadamente la malla.

Para el primer ejemplo, la ecuación utilizada es:

$$-\nu\Delta u + \underline{b} \cdot \nabla u = 0 \tag{8.3}$$

en $(x, y) \in [0, 1] \times [0, 1]$, donde

$$u(x, y) = \begin{cases} 0, & (x, y) \in \xi_1 \\ 1, & (x, y) \in \xi_2 \end{cases} \tag{8.4}$$

y $\underline{b} = (1, 3)$, como se muestra en la figura:

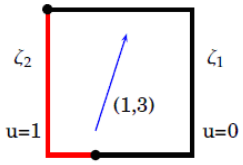

Figura 12: Dominio del problema

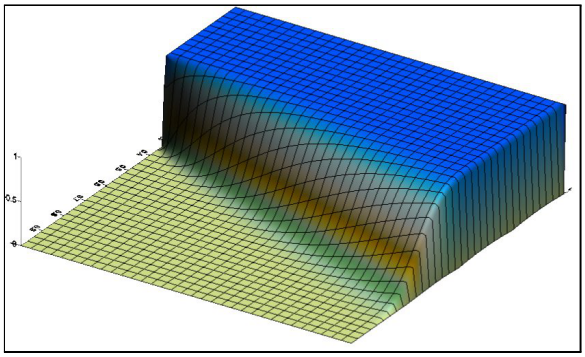

Figura 13: Solución del problema para $\nu = 0.01$

En este caso, la discretización se realizo mediante el método de Diferencias Finitas centradas y en la estabilización se usa el método de Difusión Artificial (véase [44]). Los ejemplos se resolvieron mediante el método de GMRES con

una tolerancia de 10^{-6}, en una malla global de 512×512 (261,121 grados de libertad), para distintos valores de la viscosidad ν (véase [29]). Los resultados obtenidos para las distintas descomposiciones de dominio usando el método BDDC[22] versus los algoritmos DVS, se muestran en la siguiente tabla:

Partición	ν	BDDC	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
8×8 y 64×64	0.01	12	12	11	11	11
8×8 y 64×64	0.001	9	8	8	8	7
8×8 y 64×64	0.0001	9	7	7	7	7
8×8 y 64×64	0.00001	9	7	7	7	7
16×16 y 32×32	0.01	20	19	17	17	18
16×16 y 32×32	0.001	17	14	13	14	13
16×16 y 32×32	0.0001	15	13	13	13	13
16×16 y 32×32	0.00001	16	13	13	13	13
32×32 y 16×16	0.01	33	33	29	29	31
32×32 y 16×16	0.001	30	26	25	25	25
32×32 y 16×16	0.0001	28	25	25	25	25
32×32 y 16×16	0.00001	29	25	25	25	26
64×64 y 8×8	0.01	52	53	53	52	59
64×64 y 8×8	0.001	53	46	46	46	47
64×64 y 8×8	0.0001	53	45	45	47	47
64×64 y 8×8	0.00001	54	45	45	47	48

Además se muestra el residual relativo de decaimiento para la malla gruesa 16×16 y varias mallas finas en las cuales se ve que la mejor convergencia se obtiene cuando la malla fina se incrementa y la convergencia es lenta cuando el subdominio tiene una pequeña cantidad de grados de libertad.

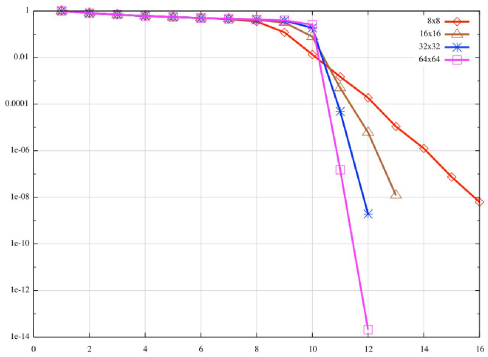

Figura 14: Residual relativo para la malla local de 16×16,en este caso $\underline{b} = (1, 3)$ y $\nu = 0.00001$ que corresponde a un valor de $Pe = 3.16e + 5$.

[22] Ejemplo realizado conjuntamente con Alberto Rosas Medina (véase [44]).

Para el segundo ejemplo, la ecuación a trabajar es

$$-\nu\Delta u + \underline{b}\cdot\nabla u + cu = 0 \tag{8.5}$$

en $(x,y) \in [-1,1]\times[0-1,1]$, donde

$$\begin{aligned} u(x,y) &= 1 \begin{cases} y=-1, & 0<x\leq 1 \\ y=1, & 0<x\leq 1 \\ x=1, & -1\leq y\leq 1 \end{cases} \\ u(x,y) &= 0, \qquad \text{en cualquier otro caso} \end{aligned} \tag{8.6}$$

el coeficiente advectivo esta dado por $\underline{b} = (y,x)$, el valor de $c = 10^{-4}$.

En este caso, la discretización se realizo mediante el método de Diferencias Finitas centradas y en la estabilización se usa el método de Difusión Artificial (véase [44]). Los ejemplos se resolvieron mediante el método de GMRES con una tolerancia de 10^{-6}, en una malla global de 32×32 (961 grados de libertad), para distintos valores de la viscosidad ν (véase [50]), cuya solución es mostrada en la gráfica:

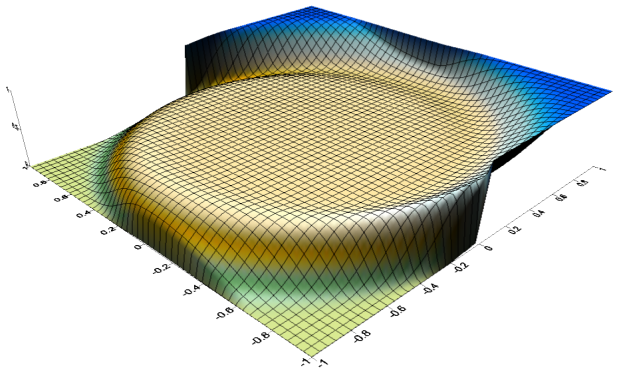

Figura 15: Solución del problema para $\nu = 0.01$

Los resultados obtenidos para las distintas descomposiciones de dominio usando el método FETI[23] versus los algoritmos DVS, se muestran en la siguiente tabla:

[23] Ejemplo realizado conjuntamente con Alberto Rosas Medina (véase [44]).

Partición	ν	FETI-DP	PRIMAL#1	DUAL#1	PRIMAL#2	DUAL#2
4×4 y 8×8	1	11	9	8	8	8
4×4 y 8×8	0.01	12	11	8	10	9
4×4 y 8×8	0.001	23	20	16	20	16
4×4 y 8×8	0.0001	45	24	19	24	18
4×4 y 8×8	0.00001	69	24	19	24	18
8×8 y 4×4	1	10	9	8	8	8
8×8 y 4×4	0.01	11	16	9	10	13
8×8 y 4×4	0.001	27	24	21	24	22
8×8 y 4×4	0.0001	68	32	25	30	26
8×8 y 4×4	0.00001	111	33	24	29	27
16×16 y 2×2	1	9	8	6	6	6
16×16 y 2×2	0.01	16	26	8	9	21
16×16 y 2×2	0.001	63	47	23	28	41
16×16 y 2×2	0.0001	176	48	29	34	42
16×16 y 2×2	0.00001	200	48	30	34	42

Para el tercer ejemplo, la ecuación a trabajar es

$$-\nu\Delta u + \underline{b} \cdot \nabla u + cu = 0 \tag{8.7}$$

en $(x, y) \in [-1, 1] \times [-1, 1]$, donde

$$\begin{aligned} u(x,y) &= 1 \begin{cases} x = -1, & -1 < y \leq 1 \\ y = 1, & -1 \leq x \leq 1 \end{cases} \\ u(x,y) &= 0, \qquad y = -1, -1 \leq x \leq 1 \\ u(x,y) &= \frac{1+y}{2}, \qquad x = 1, -1 \leq y \leq 1 \end{aligned} \tag{8.8}$$

el coeficiente advectivo esta dado por $\underline{b} = \left(\frac{1+y}{2}, 0\right)$, el valor de $c = 10^{-4}$.

En este caso, la discretización se realizo mediante el método de Diferencias Finitas centradas y en la estabilización se usa el método de Difusión Artificial (véase [44]), Los ejemplos se resolvieron mediante el método de GMRES con una to-lerancia de 10^{-6} en la norma infinita en una malla global de 32×32 (961 grados de libertad), para distintos valores de la viscosidad ν (véase [50]) , cuya solución es mostrada en la gráfica:

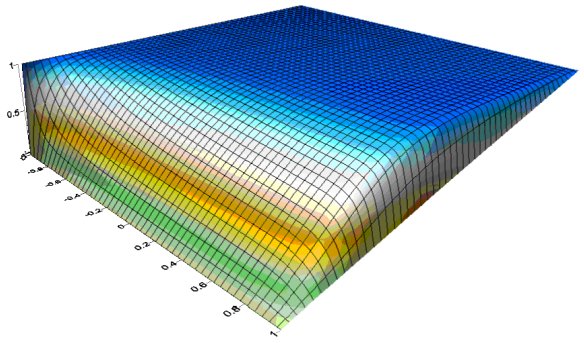

Figura 16: Solución del problema para $\nu = 0.01$

Los resultados obtenidos para las distintas descomposiciones de dominio usando el método FETI-DP[24] versus los algoritmos DVS, se muestran en la siguiente tabla:

Partición	ν	FETI-DP	PRIMAL#1	DUAL#1	PRIMAL#2	DUAL#2
4×4 y 8×8	1	13	10	10	8	8
4×4 y 8×8	0.01	13	11	10	8	7
4×4 y 8×8	0.001	9	8	9	6	6
4×4 y 8×8	0.0001	10	10	10	4	4
4×4 y 8×8	0.00001	11	9	10	3	4
4×4 y 8×8	0.000001	11	9	9	2	3
8×8 y 4×4	1	44	9	9	8	8
8×8 y 4×4	0.01	34	15	14	10	10
8×8 y 4×4	0.001	16	15	15	10	10
8×8 y 4×4	0.0001	16	27	28	9	9
8×8 y 4×4	0.00001	16	32	32	8	8
8×8 y 4×4	0.000001	16	25	25	6	5
16×16 y 2×2	1	159	8	8	6	5
16×16 y 2×2	0.01	98	22	21	9	8
16×16 y 2×2	0.001	38	37	37	18	18
16×16 y 2×2	0.0001	33	48	48	23	22
16×16 y 2×2	0.00001	46	42	41	20	20
16×16 y 2×2	0.000001	51	37	36	15	15

De los resultados mostrados en esta sección, se puede concluir que el esquema de descomposición de dominio en el espacio de vectores derivados para problemas de Advección-Difusión presenta una eficiencia del mismo orden y en algunos casos mejoran la mostrada por los métodos FETI y BDD (véase [29] y [50]).

[24] Ejemplo realizado conjuntamente con Alberto Rosas Medina (véase [44]).

8.4 Análisis de Rendimiento para Sistemas de Ecuaciones

En esta sección se muestra como usar el esquema DVS para resolver problemas con condiciones de frontera Dirichlet donde los desplazamientos son cero sobre la frontera del cuerpo elástico que ocupa el dominio Ω del espacio físico, donde sobre cada subdominio Ω_i que forma la partición gruesa del dominio Ω es resuelto el problema local usando el Método de Elemento Finito (FEM), usando funciones lineales como base.

En el caso de sistemas de ecuaciones, se resolvió[25] un sistema de ecuaciones diferenciales parciales en tres dimensiones que corresponde a la ecuación de elasticidad lineal

$$(\lambda + \mu)\,\nabla\nabla \cdot \underline{u} + \mu\Delta\underline{u} = \underline{f}_{\Omega},\ \text{en } \Omega \tag{8.9}$$

la cual es sujeta a las condiciones de frontera Dirichlet

$$\underline{u} = 0,\ \text{en } \partial\Omega \tag{8.10}$$

el dominio Ω para los experimentos numéricos es un cubo unitario homogéneo isotrópico lineal elástico. En todos nuestros experimentos los nodos primales fueron localizados en las aristas de los subdominios de la partición gruesa, lo cual es suficiente para que la matriz $\underline{\underline{A}}^t$ no sea singular.

Considerando λ y μ iguales a uno, La solución analítica de este problema se escribe como

$$\underline{u} = (\sin\pi x \sin\pi y \sin\pi z, \sin\pi x \sin\pi y \sin\pi z)\,. \tag{8.11}$$

En este caso el operador es simétrico y positivo definido, por ello se usa el método iterativo de Gradiente Conjugado para resolver el sistema lineal de ecuaciones que se genera en el esquema DVS, con una tolerancia de 10^{-7} (véase [45]). Los resultados obtenidos para las distintas descomposiciones de dominio usando el cluster Olintlali se muestran en la siguiente tabla:

Partición	Subdominios	DOF	PRIMAL#1	DUAL#1	PRIMAL#2	DUAL#2
5×5×5 y 5×5×5	125	41472	8	7	9	9
6×6×6 y 6×6×6	216	128625	8	8	10	10
7×7×7 y 7×7×7	343	331776	8	8	11	11
8×8×8 y 8×8×8	512	750141	8	8	12	12

Nótese que, el código desarrollado y usado para problemas escalares que originalmente se desarrollo para resolver una sola ecuación usando en la discretización al método de Diferencias Finitas, fue extendido para resolver problemas con el método de Elemento Finito para resolver sistemas de ecuaciones.

[25] Ejemplo realizado conjuntamente con Iván Contreras Trejo (véase [45]).

8.5 Análisis de Rendimiento en Equipos Paralelos

Para conocer el análisis de rendimiento en equipos paralelos de los métodos desarrollados de descomposición de dominio en el espacio de vectores derivados, se realizaron varias pruebas con la finalidad de conocer la eficiencia y escalabilidad de los códigos y por ende de los métodos en distintos equipos paralelos a los que se tuvo acceso, estos incluyen equipos con 8, 22, 104 y 1024 Cores.

Primeramente, es menester fundamental el encontrar la mejor descomposición de dominio para el problema a trabajar al usar la implementación secuencial y paralela acorde al equipo del que se disponga en aras de obtener la más alta eficiencia posible, después cuando el caso lo permite, se muestran las distintas métricas utilizables y sus limitaciones al aplicarlas en problemas de descomposiciones finas; por último se muestra la escalabilidad del esquema DVS usando hasta 1024 Cores.

8.5.1 Selección Óptima de una Descomposición del Dominio

Para comenzar con la selección óptima de la descomposición del dominio Ω, se toma el problema dado por la Ec.(7.1) como caso particular de la Ec.(8.1) en dos dimensiones con una descomposición fina de 1024×1024 nodos —$1,048,576$ grados de libertad— del dominio Ω, donde por ejemplo se toma sin pérdida de generalidad el algoritmo PRIMAL#1, calculado los tiempos de ejecución en los cuales se usa de uno a ocho Cores de la PC Antipolis y probando las diferentes descomposiciones[26] del dominio —que van desde 2×2 y 512×512 hasta 512×512 y 2×2— se muestran en la siguiente tabla:

	1 Core	2 Cores	3 Cores	4 Cores	5 Cores	6 Cores	7 Cores	8 Cores
Partición	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo
2x2 y 512x512	16465	10659	7207	7105	4641			
4x4 y 256x256	2251	5063	2252	2103	1643	1233	1068	947
8x8 y 128x128	855	885	482	395	314	311	283	272
16x16 y 64x64	321	348	190	149	121	125	118	117
32x32 y 32x32	26	39	26	24	23	21	21	21
64x64 y 16x16	205	595	485	477	481	461	469	469
128x128 y 8x8	1026	5453	5352	5431	5633	5843	5843	5903
256x256 y 4x4	8544	26167	25892	25902	25939	25950	25969	26003
512x512 y 2x2	34845	64230	63293	63308	63389	63475	63502	63693

Por ejemplo, suponiendo que se quiere resolver una descomposición de 2×2 y 512×512 y usar la menor cantidad de Cores posible, entonces se tienen algunas

[26] Para las corridas secuenciales usando métodos de descomposición de dominio, el mejor tiempo de ejecución se obtuvo en una descomposición de 32×32 y 32×32 en 26 segundos —el tiempo de ejecución para el programa secuencial de elemento finito que resuelve el sistema lineal algebraico asociado mediante factorización Cholesky fue de 111 segundos—. Para las corridas en paralelo se obtuvo el mejor tiempo de ejecución en una descomposición de 32×32 y 32×32 con un tiempo de 21 segundos usando 6 Cores —uno para el maestro y 5 para los esclavos—.

opciones para mantener un buen balanceo de cargas —en este caso se tienen 4 subdominios— usando 3 ó 5 Cores:

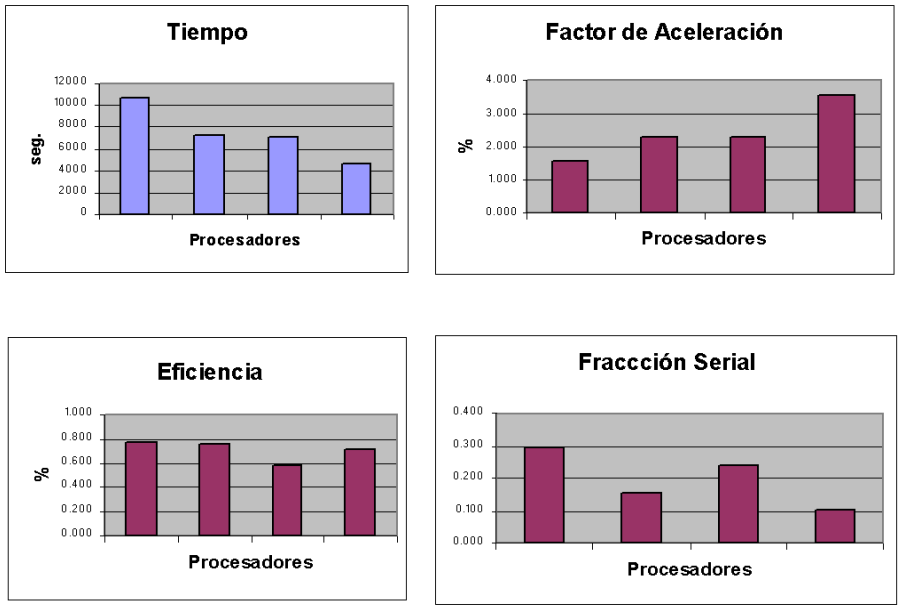

Figura 17: Métricas para la descomposición 2×2 y 512×512

- Si se desea usar 1 Core para el nodo maestro y 2 Cores para los nodos esclavos —dos subdominios por Core—, entonces el factor de aceleración[27] es

$$S(3) = T(1)/T(3) = 16465/7207 = 2.28,$$

la eficiencia es

$$E(3) = T(1)/\left(3 * T(3)\right) = 16465/(3 * 7207) = 0.761,$$

y la fracción serial es

$$F(3) = \frac{\frac{1}{S(3)} - \frac{1}{3}}{1 - \frac{1}{3}} = 0.158.$$

[27] El factor de aceleración S es tal que $1 \leq S(n) \leq n$, la eficiencia E es tal que $1/n \leq E(n) \leq 1$ y la fracción serial F es tal que $0 \leq F(n) \leq 1$.

Se considera que en el caso ideal, el factor de aceleración debería aumentar linealmente al aumentar el número de procesadores $S(p) \simeq p$; por su parte la eficiencia debería de ser cercana a la unidad cuando el hardware se está usando de forma eficiente y en caso contrario se desaprovecha este; por último la fracción serial debería tender a cero y cualquier aumento indica una sobrecarga en los procesos de comunicación.

- Si se desea usar 1 Core para el nodo maestro y 4 Cores para los nodos esclavos —un subdominio por Core—, entonces el factor de aceleración es

$$S(5) = T(1)/T(5) = 16465/4641 = 3.548,$$

la eficiencia es

$$E(5) = T(1)/\left(5 * T(5)\right) = 16465/(5 * 4641) = 0.709$$

y la fracción serial es

$$F(5) = \frac{\frac{1}{S(5)} - \frac{1}{5}}{1 - \frac{1}{5}} = 0.102.$$

En otro ejemplo, suponiendo que se quiere resolver una descomposición de 32×32 y 32×32 y usar la menor cantidad de Cores posible, entonces se tienen algunas opciones para mantener un buen balanceo de cargas —en este caso se tienen 1024 subdominios— usando 3 ó 5 Cores:

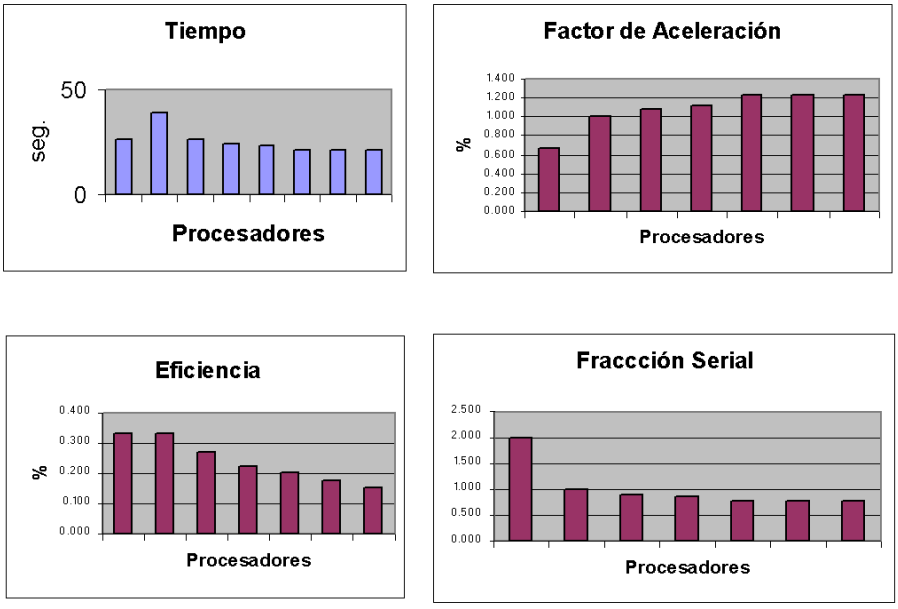

Figura 18: Métricas para la descomposición 32×32 y 32×32

- Si se desea usar 1 Core para el nodo maestro y 2 Cores para los nodos esclavos —512 subdominios por Core—, entonces el factor de aceleración es

$$S(3) = T(1)/T(3) = 26/26 = 1,$$

la eficiencia es

$$E(3) = T(1)/\left(3 * T(3)\right) = 26/(3 * 26) = 0.333$$

y la fracción serial es

$$F(3) = \frac{\frac{1}{S(3)} - \frac{1}{3}}{1 - \frac{1}{3}} = 1.$$

- Si se desea usar 1 Core para el nodo maestro y 4 Cores para los nodos esclavos —256 subdominios por Core—, entonces el factor de aceleración es

  $$S(5) = T(1)/T(5) = 26/23 = 1.130,$$

  la eficiencia es

  $$E(5) = T(1)/\left(5 * T(5)\right) = 26/(5 * 23) = 0.377,$$

  y la fracción serial es

  $$F(5) = \frac{\frac{1}{S(5)} - \frac{1}{5}}{1 - \frac{1}{5}} = 0.856.$$

Nótese que la descomposición usada en el primer ejemplo dista mucho de ser la óptima[28], ya que la descomposición de 32×32 y 32×32 usada en el segundo ejemplo genera el mejor tiempo de ejecución tanto en secuencial como en paralelo, pero las métricas no reflejan esta mejora, aunque el tiempo de ejecución es mínimo. Por ello es necesario siempre hacer corridas de prueba buscando la descomposición que presente el menor tiempo de ejecución posible para el equipo paralelo con el que se cuente. Estas pruebas dependen fuertemente de la capacidad computacional de cada Core y de la red usada para interconectar los Cores que forman parte del equipo paralelo.

Observación 7 *Nótese que esta forma de medir la eficiencia, el factor de aceleración y la fracción serial tiene un detalle fino, ya que el tiempo de ejecución tomado en un procesador —para este caso es de* 16465 *segundos en la descomposición* 2×2 *y* 512×512*— dista mucho de ser el mejor tiempo posible para el problema global de* 1024 *nodos —*26 *segundos—. Esto puede ser una limitante para obtener valores adecuados en las métricas; y medir la eficiencia al usar equipo paralelo cuando se trabaja con la resolución de dominios en los cuales se realiza una descomposición fina; particularmente cuando las descomposiciones son adecuadas para cientos de Cores, pues las pruebas en un Core o en pocos Cores no son posibles de realizar por el consumo excesivo de recursos computacionales y por que no son conmensurables con las corridas secuenciales.*

[28] En la descomposición de 2×2 y 512×512 el tiempo de ejecución secuencial es 16,465 seg., en paralelo usando 3 Cores es de 7,207 seg. y en 5 Cores es de 4,641 seg. Mientras que para la descomposición de 32×32 y 32×32, el tiempo de ejecución secuencial es de 26 seg., en paralelo usando 3 Cores es de 26 seg. y en 5 Cores es de 23 seg.

Además, de los datos de las corridas mostradas en la tabla, es notorio el efecto del mal balanceo de carga, nótese que:

- Para la malla de 32×32 y 32×32 el mejor tiempo de ejecución se obtiene en 6 Cores —21 segundos— y al aumentar el número de Cores en la corrida, no hay disminución del tiempo de cálculo.
- Para la malla de 512×512 y 2×2 el aumento en el número de procesadores sólo incide en un aumento en el tiempo de ejecución.
- En particular, para la malla de 2×2 y 512×512 al usar 3 Cores —sólo dos son usados realmente para el cálculo ya que el tercer Core se usa para la asignación de tareas y el control de los nodos esclavos— el factor de aceleración 2.28 es el esperado para el esquema Maestro-Esclavo.

8.5.2 Análisis de Rendimiento Usando Métricas

En esta sección se muestra mediante particiones relativamente pequeñas del dominio, el uso de las métricas —aceleración, eficiencias y fracción serial— en las cuales es posible obtener una alta eficiencia computacional cuando se proporciona una descomposición del dominio adecuada para el equipo paralelo con el que se cuente.

Haciendo uso del Cluster Pohualli de 104 Cores, a continuación se presentan varias tablas en las cuales se muestran las métricas para diferentes descomposiciones del dominio Ω.

1) Para una descomposición de 4×4 y 150×150 —$360,000$ grados de libertad— se obtiene

Cores	Tiempo	Aceleración	Eficiencia	Frac. Ser.
1	267			
3	146	1.82	0.60	0.32
5	85	3.14	0.62	0.14
9	56	4.76	0.52	0.11
17	33	8.09	0.47	0.06

2) Para una descomposición de 4×4 y 200×200 —$640,000$ grados de libertad— se obtiene

Cores	Tiempo	Aceleración	Eficiencia	Frac. Ser.
1	1082			
3	391	2.76	0.92	0.04
5	216	5.0	1.00	0.00
9	146	7.41	0.82	0.02
17	82	13.19	0.77	0.01

3) Para una descomposición de 4×4 y 250×250 —$1,000,000$ grados de libertad— se obtiene

Cores	Tiempo	Aceleración	Eficiencia	Frac. Ser.
1	2628			
3	946	2.77	0.92	0.039
5	539	4.87	0.97	0.006
9	329	7.98	0.88	0.015
17	184	14.20	0.83	0.012

4) Para una descomposición de 4×4 y 300×300 —$1,440,000$ grados de libertad— se obtiene

Cores	Tiempo	Aceleración	Eficiencia	Frac. Ser.
1	5295			
3	2538	2.08	0.69	0.218
5	1391	3.80	0.76	0.078
9	804	6.58	0.73	0.045
17	441	12.00	0.70	0.025

De estas tablas se desprende que seleccionando la descomposición adecuada se pueden tener excelentes resultados en la eficiencia como es el caso de la descomposición 4×4 y 200×200. Además se muestra una gama de otras eficiencias según el número de procesadores usado y la descomposión seleccionada. Nótese que en todos los casos la fracción serial disminuye sustancialmente con el aumento del número de procesadores.

Otros ejemplos interesantes en los cuales se muestra el efecto de mandar descomposiciones no adecuadas y que se reflejan en una baja eficiencia computacional sin importar el aumento del número de procesadores, pero en todos los casos el tiempo de ejecución siempre disminuye:

i) Para una descomposición de 8×8 y 250×250 —$4,000,000$ grados de libertad— en el Cluster Kanbalam se obtiene

Cores	Tiempo	Aceleración	Eficiencia	Frac. Ser.
1	11366			
3	5541	2.05	0.68	0.23
5	3011	3.77	0.75	0.08
9	1855	6.12	0.68	0.05
17	1031	11.02	0.64	0.03
33	595	19.10	0.57	0.02
65	375	30.30	0.46	0.01

ii) Para una descomposición de 10×9 y 250×250 —$5,625,000$ grados de libertad— en el Cluster Pohualli se obtiene

Cores	Tiempo	Aceleración	Eficiencia	Frac. Ser.
1	19387			
6	4777	4.05	0.67	0.09
11	2702	7.17	0.65	0.05
46	801	24.20	0.52	0.02
91	509	38.08	0.41	0.01

De todos estos ejemplos se desprende que buscando la adecuada descomposición es posible encontrar eficiencias altas para el esquema DVS, pero siempre se tiene que tener en cuenta el buscar el menor tiempo de ejecución —véase sección anterior— antes que una alta eficiencia con un mayor tiempo de ejecución.

Por otro lado, pese a que Kanbalam es más eficiente[29] que los otros Clusters a los que se tuvo acceso —en particular Pohualli—, es posible encontrar una descomposición del dominio que mejore el tiempo de ejecución, aún en equipos con recursos inferiores, para ello es necesario aprovechar las características propias del Hardware del Cluster haciendo una adecuada selección de la descomposición del dominio. Para mostrar esto, se toma una descomposición de 32×32 y 150×150 —$23,040,000$ grados de libertad— en ambos Clusters con los siguientes resultados:

Cores	Pohualli	Kanbalam
16	9158 seg	ND
32	5178 seg	5937 seg
64	3647 seg	4326 seg
100	2661 seg	
128		2818 seg

Como se muestra en la tabla, en todos los casos el Cluster Pohualli usando como máximo 100 Cores obtiene un tiempo de cálculo inferior al que requiere Kanbalam usando a lo más los 128 Cores.

Haciendo uso de las métricas de aceleración y eficiencia relativa[30] se tiene que para el Cluster Kanbalam $S_{128}^{32} = 5937/2818 = 2.10$ donde lo esperado sería $S_{128}^{32} = 32/128 = 4.00$, para el caso de la eficiencia $E_{128}^{32} = (32/128) * (5937/2818) = 0.52$.

En el caso del Cluster Pohualli se tiene que $S_{100}^{16} = 9158/2661 = 3.44$ donde lo esperado sería $S_{100}^{16} = 16/100 = 6.35$, para el caso de la eficiencia $E_{100}^{16} = (16/100) * (9158/2661) = 0.55$.

Haciendo uso del mismo número de Cores base para Pohualli que para Kanbalam, se tiene que $S_{100}^{32} = 5178/2661 = 1.94$ donde lo esperado sería $S_{100}^{16} = 32/100 = 3.12$, para el caso de la eficiencia $E_{100}^{16} = (32/100)*(5178/2661) = 0.62$;

De todo lo anterior, se desprende que el Cluster Pohualli obtiene valores de una aceleración y eficiencias relativas ligeramente mejores que el Cluster Kanbalam, pero esto no se refleja en la disminución de casi 6% del tiempo de ejecución y del uso de 28 Cores menos.

[29] El Cluster Kanbalam esta formado de procesadores AMD Opteron a 2.6 GHtz de 64 bits, cada 4 Cores con 8 GB de RAM interconectados con un switch de 10 Gbps de baja latencia.
El Cluster Pohualli esta formado de procesadores Intel Xeon a 2.33 GHtz de 64 bits, cada 8 Cores cuentan con 32 GB de RAM interconectados con un switch de 1 Gbps.

[30] Aceleración relativa es $S_p^{p'} = \frac{T_{p'}}{T_p}$ para $p \geq p'$, en la cual se espera que $S_p^{p'} \simeq \frac{p}{p'}$ y eficiencia relativa es $E_p^{p'} = \frac{p'}{p} S_p^{p'} = \frac{p'}{p} \frac{T_{p'}}{T_p}$.

Además, el costo computacional[31] $C_p = P * T_p$, que para el caso del cluster Kanbalam es $C_{128} = 360,704$ y en Pohualli es $C_{100} = 266,100$ que representa una disminución de 27%; además de un factor muy importante, el Cluster Pohualli tuvo un costo monetario mucho menor con respecto del Cluster Kanbalam.

8.5.3 Escalabilidad del Esquema DVS

Por último, se realizaron pruebas[32] con el Cluster Kanbalam, mediante una petición especial para tener acceso a los 1024 Cores del Cluster, a la cual el Comité Técnico del mismo dio acceso después de concluir un reparación mayor del equipo que obligo a apagar el Cluster, este acceso sólo se dio por unas horas y de forma exclusiva, lo cual agradezco enormemente, ya que el Cluster tiene una gran demanda dentro y fuera de la UNAM.

Las pruebas realizadas se hicieron usando desde 32 hasta 1024 Cores, para las descomposiciones de 31×33 y 150×150 —$4,194,304$ grados de libertad—, 31×33 y 200×200 —$40,920,000$ grados de libertad— y 31×33 y 250×250 —$63,937,500$ grados de libertad— se obtienen los siguientes tiempos de ejecución.

	Cores					
Subdominio	32	64	128	256	512	1024
31×33 y 150×150	7315 s	4016 s	2619 s	1941 s	1541 s	1298 s
31×33 y 200×200	ND	16037 s.	4916 s	3166 s	2688 s	2295 s
31×33 y 250×250	ND	ND	26587 s	8716 s	6388 s	ND

En donde si usamos las métricas de aceleración y eficiencia relativas obtenemos los siguientes resultados

Subdominio	Aceleración	Aceleración esperada	Eficiencia
31×33 y 150×150	$S_{1024}^{32} = 5.6$	$S_{1024}^{32} = 32$	$E_{1024}^{32} = 0.2$
31×33 y 200×200	$S_{1024}^{64} = 5.9$	$S_{1024}^{32} = 8$	$E_{1024}^{64} = 0.7$
31×33 y 250×250	$S_{1024}^{128} = 4.1$	$S_{1024}^{32} = 4$	$E_{1024}^{128} = 1.0$

De esta última tabla —$E_{1024}^{128} = 1.0$ para la descomposición 31×33 y 250×250—, se desprende que los algoritmos desarrollados son altamente escalables en equipos paralelos, ya que es posible obtener una alta eficiencia al encontrar descomposiciones adecuadas al Hardware. Y que pueden usarse para resolver problemas que involucren una gran cantidad de grados de libertad.

[31] El costo o trabajo de resolver un problema en paralelo es el producto del tiempo de cálculo en paralelo T_p por el número de procesadores usado P y se representa por $C_p = P * T_p$.

[32] No todas las pruebas que se plantearon fueron posibles de realizar, en algunos casos la limitante fue las características físicas de los equipos computacionales, en otros es el acceso limitado y de corta duración —para el uso de 256, 512 y 1024 Cores en el Cluster Kanbalam sólo se dispuso de unas cuantas horas de cómputo y en las cuales, varios Cores del Cluster presentaron fallas de hardware por lo que algunas corridas no se concluyeron de forma satisfactoria—.

8.6 Criterios Integrales para Evaluar el Esquema DVS

En el desarrollo e implementación numérica de los distintos métodos de descomposición de dominio, es necesario medir de alguna forma la eficiencia de los diversos métodos entre sí, algunos criterios comúnmente usados son:

1. Dado un dominio Ω y una descomposición fija, usar el número de iteraciones como criterio de eficiencia.

2. Dado un dominio Ω y haciendo refinamientos de la partición, usar el número de iteraciones como criterio de eficiencia.

3. Dado un dominio Ω y una descomposición del mismo, buscar aquella partición en la que el tiempo de ejecución sea mínimo al variar las particiones posibles.

En principio, estas formas de medir la eficiencia de los diversos métodos no deberían de ser excluyentes entre sí, por el contrario, juntas dan un criterio robusto de la eficiencia de un método de descomposición de dominio para un problema en particular implementado en un equipo de cómputo en las cuales ciertas descomposiciones son posibles —ya sea por limitaciones fenomenológicas o por cuestiones computacionales—.

Para mostrar las implicaciones de las distintas formas de medir la eficiencia, se hace un análisis de las diversas opciones para cada uno de los casos.

1.- Dado un dominio Ω y una descomposición fija, usar el número de iteraciones como criterio de eficiencia En este caso, se usa la Ec.(8.1) como simétrica, tomando una descomposición del dominio Ω en tres dimensiones en la cual se toma una malla gruesa$10 \times 10 \times 10$ que genera $10,000$ subdominios y en la que cada subdominio es descompuesto en $10 \times 10 \times 10$ elementos, los grados de libertad asociados al sistema son $970,299$, donde el número de vértices primales usados es de $22,599$. Obteniendo los siguientes resultados:

	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
Iteraciones:	8	8	8	8

Aquí, lo único que se observa, es que todos los métodos obtienen la misma eficiencia global en cuanto al número de iteraciones, pero nada dice de los tiempos involucrados en la ejecución. Si ahora se toma en cuenta los tiempos de ejecución en un procesador se obtiene:

	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
Tiempo:	1,380s	1,387s	1,490s	1,520s

Y si se usan varios procesadores de un Cluster —en este ejemplo se usó el Cluster Pohualli— se obtiene:

Cores	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
3	966s	965s	930s	953s
11	184s	186s	175s	181s
101	28s	29s	27s	27s

Esta forma integral de medir la eficiencia, da una idea más realista de la eficiencia de los métodos, pero nuevamente hay que tomar en cuenta que los tiempos de ejecución dependen directamente de la arquitectura de cómputo en la que se realicen las pruebas, en especial del balanceo de la carga de trabajo, de la infraestructura de red que interconecten los nodos del Cluster y si estos son Cores virtuales o reales.

2.- Dado un dominio Ω y haciendo refinamientos de la partición, usar el número de iteraciones como criterio de eficiencia En este caso, se usa la Ec.(8.1) como simétrica, se toma una descomposición del dominio Ω en dos dimensiones, en la primer tabla se muestra la descomposición usada, el número de subdominios, los grados de libertad asociados al sistema y el número de vértices primales usados:

Ejemplo	Partición	Subdominios	Grados Libertad	Primales
1	22×22 y 22×22	484	233,289	441
2	24×24 y 24×24	576	330,625	529
3	26×26 y 26×26	676	455,625	625
4	28×29 y 28×28	784	613,089	729
5	30×30 y 30×30	900	808,201	841

En la segunda tabla se muestra el número de iteraciones requeridas para alcanzar la tolerancia solicitada al método:

Ejemplo	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
1	13	14	15	16
2	14	14	15	15
3	14	14	15	15
4	14	14	15	15
5	15	14	15	15

En la siguiente tabla se muestra el tiempo de ejecución en un procesador para concluir las iteraciones:

Ejemplo	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
1	8s	7s	14s	27s
2	13s	13s	21s	40s
3	19s	19s	33s	61s
4	25s	27s	44s	85s
5	36s	38s	61s	116s

En la última tabla se muestra el tiempo de ejecución en 4 procesadores para concluir las iteraciones:

Ejemplo	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
1	2.9s	2.95s	2.93s	2.99s
2	4.80s	4.89s	4.81s	4.85s
3	7.2s	7.4s	7.3s	7.4s
4	8.92s	8.95s	8.91s	8.93s
5	13.02s	13.05s	13.02s	13.3s

Nuevamente, esta forma integral de medir la eficiencia, da una idea más realista de la eficiencia de los métodos, pero nuevamente hay que tomar en cuenta que los tiempos de ejecución dependen directamente de la arquitectura de cómputo en la que se realicen las pruebas, en especial del balanceo de la carga de trabajo, de la infraestructura de red que interconecten los nodos del Cluster y si estos son Cores virtuales o reales.

3.- Dado un dominio Ω y una descomposición del mismo, buscar aquella partición en la que el tiempo de ejecución sea mínimo al variar las particiones posibles Por último, supóngase que deseo resolver la Ec.(7.1) con un dominio Ω mediante una discretización de 1024×1024 nodos ($1,048,576$ grados de libertad) mediante el algoritmo NN-NP-PRIMAL#1 dado por la Ec.(4.21), de manera inmediata surgen las siguientes preguntas: ¿cuáles son las posibles descomposiciones validas? y ¿en cuántos procesadores se pueden resolver cada descomposición?. Para este ejemplo en particular, sin hacer la tabla exhaustiva, se tiene

Partición	Subdominios	Procesadores
2x2 y 512x512	4	2,3,5
4x4 y 256x256	16	2,3,5,9,17
8x8 y 128x128	64	2,3,5,9,17,33,65
16x16 y 64x64	256	2,3,5,9,17,33,65,129,257
32x32 y 32x32	1024	2,3,5,9,17,33,65,129,..,1025
64x64 y 16x16	4096	2,3,5,9,17,33,65,129,...,4097
128x128 y 8x8	16384	2,3,5,9,17,33,65,129,...,16385
256x256 y 4x4	65536	2,3,5,9,17,33,65,129,...,65537
512x512 y 2x2	262144	2,3,5,9,17,33,65,129,...,262145

De esta tabla es posible seleccionar las descomposiciones que se adecuen a las características del equipo paralelo con que se cuente, para evaluar el tiempo de ejecución de este ejemplo use la PC Antipolis, obteniendo resultados mostrados en la tabla de la sección (8.5.1).

De estos resultados, se desprende que, dependiendo del tamaño de la malla gruesa —número de subdominios a trabajar— y de la malla fina, es siempre posible encontrar una descomposición de dominio en que el tiempo de cálculo sea mínimo, tanto al usar un solo Core —programa secuencial 26 segundos—, como al usar múltiples Cores interconectados mediante la biblioteca de paso de mensajes MPI —el tiempo mínimo se obtuvo usando 6 Cores en 21 segundos—, pero es también notorio el efecto que genera el mal balanceo de carga, el cual

se refleja en que no disminuye el tiempo de ejecución al aumentar el número de procesadores y en algunos casos el tiempo aumenta conforme se agregan más Cores.

Nótese que conforme la partición en los subdominios se hace más fina, se incrementa notablemente el tiempo de cálculo necesario para resolver los sistemas lineales asociados a los subdominios, en particular en la resolución del sistema lineal asociado a $\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}$, si el número de nodos por subdominio es grande puede que exceda la cantidad de memoria que tiene a su disposición el Core y por el contrario, un número pequeño de nodos generarían una infrautilización del poder computacional de los nodos esclavos.

Por otro lado, el refinamiento de la malla gruesa, involucra un aumento considerable del número de objetos subdominio en los nodos esclavos, con los que el nodo maestro tendrá comunicación, incrementando la granularidad de las comunicaciones, con la consecuente degradación en la eficiencia.

De todo lo anterior se pueden hacer algunas observaciones importantes Para una evaluación objetiva e integra de la eficiencia de los diversos métodos de descomposición de dominio —en particular de los desarrollados— y su implementación computacional en una arquitectura de cómputo particular, es necesario tomar en cuanta los siguientes factores:

- Número de iteraciones para una descomposición dada.
- Número de iteraciones para diferentes particiones de una descomposición dada.
- Elección de la partición que genere el menor tiempo de ejecución para un problema en una arquitectura de cómputo específica.

Estas formas de medir la eficiencias en su conjunto, dan una idea realista de la eficiencia de los métodos, pero hay que tomar en cuenta que los tiempos de ejecución dependen directamente de la arquitectura de cómputo en la que se realicen las pruebas, en especial del balanceo de la carga de trabajo, de la infraestructura de red que interconecten los nodos del Cluster y si estos son Cores virtuales o reales.

9 Conclusiones y Trabajo Futuro

Los modelos matemáticos (véase [37] y [10]) de muchos sistemas de interés, incluyendo una gran cantidad de sistemas importantes de Ciencias de la Tierra e Ingeniería, conducen a una gran variedad de ecuaciones diferenciales parciales cuyos métodos de solución están basados en el procesamiento de sistemas algebraicos de gran escala. Además, la increíble expansión experimentada por el Hardware y Software computacional existente, ha hecho posible el efectivo tratamiento de problemas de un cada vez mayor incremento de diversidad y complejidad que poseen las aplicaciones Científicas y de Ingeniería.

El cómputo en paralelo destaca entre las nuevas herramientas computacionales y para hacer uso efectivo de los equipos de cómputo de alto desempeño más avanzados disponibles actualmente, el Software que aproveche al máximo el equipo paralelo es requerido. Los métodos de descomposición de dominio han sido desarrollados precisamente para hacer un efectivo tratamiento de ecuaciones diferenciales parciales en paralelo.

Idealmente, el objetivo principal en la investigación de los métodos de descomposición de dominio es producir algoritmos capaces de obtener *la solución global por la resolución de problemas locales exclusivamente*, pero hasta ahora, esto solo ha sido una inspiración, que es, un fuerte deseo para lograr dichas propiedades y por eso lo llamamos *el paradigma DDM*. En la anterior década, los algoritmos competitivos de descomposición de dominio desarrollados han sido los métodos precondicionados sin traslape y necesariamente incorporan restricciones, los cuales poseen un reto adicional para el paradigma DDM.

En el presente trabajo introducimos un grupo de ocho algoritmos, de los cuales, cuatro son precondicionados, a los que nos referiremos como los *algoritmos DVS* (véase [36], [38]), los cuales satisfacen el paradigma DDM. De ellos se derivan los eficientes y bien conocidos algoritmos BDDC y FETI-DP, los cuales fueron incorporados en un nuevo marco de trabajo; el método de descomposición de dominio en el espacio de vectores derivados (DVS).

Así, para poner en perspectiva nuestros desarrollos, el presente trabajo se inicio con una revisión de las formulaciones Dirichlet-Dirichlet y Neumann-Neumann a nivel continuo, para después dar paso a la derivación del método de descomposición de dominio en el espacio de vectores derivados y se muestra como este esquema es puesto dentro de las formulaciones continuas, completando así el modelo matemático del esquema DVS.

Con el modelo matemático del esquema DVS derivado, se procede ha presentar la formulación numérica —que esta estrechamente ligada a la implementación computacional— y detallar las características de la implementación computacional tanto en su forma secuencial como paralela; y mediante ejemplos numéricos, se realiza el análisis y discusión de resultados (véase [39]) para algunos problemas emblemáticos que generan sistemas algebraicos simétricos, no simétricos e indefinidos. Finalmente, en la presente sección se dan las conclusiones de los logros alcanzados en esta tesis y se esboza lo que se considera pueden ser sus perspectivas.

9.1 Conclusiones

La formulación del método de descomposición de dominio en espacio de vectores derivados (véase [36], [38]), es un esquema que involucra 8 algoritmos, este es un marco unificador de dos de las formulaciones algebraicas más comúnmente usados en los métodos de descomposición de dominio sin traslapes, estos son: Finite Element Tearing and Interconnect Dual-Primal (FETI-DP) y Balancing Domain Decomposition by Constraints (BDDC) (véase [18], [42], [43], [53], [54], [55], [57] y [58]).

Un breve y efectivo resumen de los ocho métodos de descomposición de dominio sin traslape en el espacio de vectores derivados, de los cuales cuatro son formulaciones primales y los otros cuatro son formulaciones duales, es dado a continuación:

1. Las fórmulas no precondicionadas son:

   - Formulación Dirichlet-Dirichlet

   $$\left\{ \underline{\underline{aS}}\underline{u}_\Delta = \underline{f}_\Delta \quad \text{y} \quad \underline{\underline{j}}\underline{u}_\Delta = 0, \quad \text{(PRIMAL\#1)} \right. \tag{9.1}$$

   - Formulación Neumann-Neumann

   $$\left\{ \begin{array}{l} \underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\lambda}_\Delta = \underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{\underline{a}}\underline{\lambda}_\Delta = 0, \quad \text{(DUAL\#1)} \\ \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{v}_\Delta = \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{\underline{aS}}\underline{v}_\Delta = 0, \quad \text{(PRIMAL\#2)} \\ \underline{\underline{Sa}}\underline{\mu}_\Delta = \underline{\underline{SaS}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \text{ y } \underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\mu}_\Delta = 0, \quad \text{(DUAL\#2)} \end{array} \right. \tag{9.2}$$

2. Las fórmulas precondicionadas son:

   - Formulación Dirichlet-Dirichlet DVS BDDC

   $$\underline{\underline{aS}}^{-1}\underline{\underline{aS}}\underline{u}_\Delta = \underline{\underline{aS}}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{\underline{j}}\underline{u}_\Delta = 0, \quad \text{(PRIMAL\#1)} \tag{9.3}$$

   - Formulación Neumann-Neumann DVS FETI-DP

   $$\begin{array}{l} \underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\lambda}_\Delta = \underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{\underline{a}}\underline{\lambda}_\Delta = 0, \quad \text{(DUAL\#1)} \\ \text{donde } \underline{u}_\Delta = \underline{\underline{aS}}^{-1}\left(\underline{f}_\Delta - \underline{\underline{j}}\underline{\lambda}_\Delta\right) \end{array} \tag{9.4}$$

   - Formulación Neumann-Neumann DVS-PRIMAL

   $$\begin{array}{l} \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{v}_\Delta = \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{\underline{aS}}\underline{v}_\Delta = 0, \text{(PRIMAL\#2)} \\ \text{donde } \underline{u}_\Delta = \underline{\underline{aS}}^{-1}\left(\underline{f}_\Delta - \underline{\underline{j}}\underline{\underline{S}}\underline{v}_\Delta\right) \end{array} \tag{9.5}$$

   - Formulación Neumann-Neumann DVS-DUAL

   $$\begin{array}{l} \underline{\underline{SaS}}^{-1}\underline{\underline{a}}\underline{\mu}_\Delta = \underline{\underline{SaS}}^{-1}\underline{\underline{aS}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\mu}_\Delta = 0, \text{(DUAL\#2)} \\ \text{donde } \underline{u}_\Delta = \underline{\underline{aS}}^{-1}\left(\underline{f}_\Delta + \underline{\mu}_\Delta\right) \end{array} \tag{9.6}$$

El paradigma DVS, constituido por 8 algoritmos con características similares, a los cuales nos hemos referido como los algoritmos DVS. Donde cada uno de los algoritmos precondicionados posee las siguientes conspicuas características:

- Las formulaciones Dual y Primal de dos de los métodos comúnmente usados —FETI-DP y BDDC— han sido derivadas de una manera unificada. El esquema desarrollado incluye formulaciones algebraicas para matrices simétricas, no simétricas e indefinidas —i.e. no positivas y no negativas definidas—. Además se detallan las condiciones que tales matrices deben de satisfacer para que los algoritmos generales sean aplicables.

- El esquema DVS permite aplicar técnicas de descomposición de dominio directamente al sistema de matrices que son obtenidas después de que la ecuación diferencial o sistema de tales ecuaciones han sido discretizadas. La aplicación de tales procedimientos no requieren del conocimiento acerca de la ecuación diferencial que originó las matrices.

- Los algoritmos tienen una aplicabilidad general, ya que ellos pueden ser aplicables a problemas de valor en la frontera asociados a una sola ecuación diferencial o a sistemas de ecuaciones.

- Para cada uno de los algoritmos, se han desarrollado formulaciones explícitas en términos de matrices; ellos están dados en las Ecs.(4.4, 4.21, 4.30, 4.43, 4.11, 4.29, 4.36 y 4.50).

- Tales formulas permiten desarrollar códigos que satisfacen el paradigma DDM, i.e. *en el cual la solución del problema global es obtenida exclusivamente por resolución de problemas locales.*

- Son precondicionados y con restricciones. En el caso de matrices no simétricas, la eficiencia numérica de los algoritmos precondicionados están en el mismo orden como los algoritmos de descomposición de dominio del estado del arte (véase [34] y [35]).

Además, hay varias propiedades numéricas y computacionales de los algoritmos DVS que destacan, entre las más importantes se tienen:

- El código es independiente de la geometría.

- El código que se obtiene es robusto, ya que con ligeras modificaciones es aplicado a problemas en dos y tres dimensiones; además soportan matrices simétricas, no simétricas e indefinidas

- El mismo código puede ser aplicado a una sola ecuación elíptica y a sistemas de ecuaciones de elasticidad lineal, con cambios mínimos en comportamientos específicos.

- Los algoritmos desarrollados son paralelizables y escalables con una alta eficiencia computacional.

- El código soporta diferentes métodos de solución de sistemas lineales —directos e iterativos— en los subdominios.
- El algoritmo global es débilmente acoplado a los subdominios.
- El desarrollo del código orientado a objetos simplifica la programación, permitiendo que ha partir de la implementación secuencial se genere la implementación paralela con un mínimo de cambios, con la consecuente facilidad de expansión y mantenimiento del código.
- Los códigos desarrollados fueron aplicados para resolver diversos problemas con valores en la frontera que existen en el modelado de ciertos fenómenos geofísicos, tales como el transporte de solutos en fluidos libres como en fluidos en medios porosos. También presentamos resultados para problemas de elasticidad estática, de este modo ilustramos la aplicación de los algoritmos desarrollados a sistemas de ecuaciones diferenciales.

Por otro lado, las formulaciones FETI-DP y BDDC son óptimas en el sentido de que el número de condicionamiento κ de estos problemas de interfases convergen asintóticamente como (véase [27], [46] y [47])

$$\kappa = O\left(1 + \log^2\left(H/h\right)\right) \tag{9.7}$$

—los términos H y h son tomados según definiciones de la sección (3.1)— nuestras formulaciones de los algoritmos DVS FETI-DP y DVS BDDC en el espacio de vectores derivados muestran un desempeño similar cuando usan el mismo conjunto de restricciones primales.

Estas propiedades hacen de los algoritmos DVS muy adecuados como herramientas usadas en la construcción de Software masivamente paralelizables, necesario para ser usada en la programación de las computadoras paralelas de alto desempeño disponibles actualmente[33].

9.2 Trabajo Futuro

De los algoritmos desarrollados, dos hasta donde se tiene conocimiento son totalmente diferentes a cualquiera de los reportados anteriormente y deben ser motivo de investigaciones futuras, además de que los métodos que se mostraron hasta ahora sólo se ha aplicado a problemas elípticos escalares y se inicia a problemas vectoriales, pero es posible aplicarlos a problemas parabólicos escalares y vectoriales tanto lineales como no lineales, por ello el trabajo futuro puede ser esbozado como:

- Hacer una investigación sobre nuestros métodos no reportados en la literatura.

[33] Una versión de los códigos desarrollados de los algoritmos DVS está en linea en la página WEB http://www.mmc.geofisica.unam.mx/acl/DVS/

- Aplicar los métodos desarrollados y ampliar el código para soportar:

    1. Problemas elípticos con condiciones de frontera tipo Robin.
    2. Problemas parabólicos escalares y vectoriales.
    3. Problemas elípticos y parabólicos no lineales.

- Implementar un mecanismo computacional que permita definir ecuaciones o sistemas de ecuaciones, sus parámetros y condiciones de frontera para que el código sea independiente de ellas.

En cuanto a la implementación computacional de los métodos desarrollados, la paralelización se realiza mediante el paso de mensajes usando la biblioteca MPI en C++ para interconectar cada uno de los Cores del Cluster. Pero es posible usar eficientemente la interconexión de memoria compartida de los actuales equipos de cómputo mediante OpenMP y además usar los cada vez más numerosos Cores CUDA —que una sola tarjeta NVIDIA TESLA puede tener del orden de cientos ellos—.

Los métodos de descomposición de dominio sin traslape y en particular el esquema DVS implementado con sus ocho algoritmos, pueden hacer uso de esta forma integradora de paralelismo. Para ello, la interconexión de los equipos de memoria compartida, se realizaría mediante MPI y en cada equipo de memoria compartida se manipularían uno o más subdominios mediante OpenMP —al ser cada subdominio independiente de los demás— y la manipulación de matrices y operaciones entre matrices y vectores que requiere cada subdominio se realizarían en las tarjetas NVIDIA mediante los numerosos Cores CUDA.

Donde la integración de esta forma de paralelismo en el código, es un paso natural, que involucra hacer cambios mínimos al código, al ser necesario sólo cambiar los comportamientos locales que requieren otro tipo de paralelismo al implementado actualmente, ya que la jerarquía de clases del código desarrollado permite especializar los comportamientos que implementan las comunicaciones y esto queda de manifiesto al reutilizar toda la jerarquía de clases de la implementación secuencial en la paralela.

Esto permitiría tener códigos reutilizables en distintas arquitecturas de cómputo paralelo, además de hacer uso de equipos de cómputo cada vez más asequibles y de menor costo, pero con una creciente eficiencia computacional que pueden competir con los grandes equipos de cómputo de alto desempeño.

A Consideraciones Sobre la Formulación Numérica y su Implementación Computacional

Para realizar la implementación de cada uno de los métodos desarrollados de descomposición de dominio en el espacio de vectores derivados (DVS) —véase sección (6.2)—, es necesario trabajar con los operadores $\underline{\underline{a}}, \underline{\underline{j}}, \underline{\underline{S}}$ y $\underline{\underline{S}}^{-1}$, así como realizar las operaciones involucradas entre ellos.

Normalmente los operadores $\underline{\underline{S}}$ y $\underline{\underline{S}}^{-1}$ no se construyen —son operadores virtuales— porque su implementación generaría matrices densas, las cuales consumen mucha memoria y hacen ineficiente su implementación computacional. En vez de eso, sólo se realizan las operaciones que involucra la definición del operador, por ejemplo $\underline{\underline{S}}\underline{u_\Gamma}$

$$\underline{\underline{S}}\underline{u_\Gamma} = \left(\underline{\underline{A}}_{\Delta\Delta} - \underline{\underline{A}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{\underline{A}}_{\Pi\Delta}\right)\underline{u_\Gamma} \tag{A.1}$$

en donde, para su evaluación sólo involucra la operación de multiplicación matriz-vector y resta de vectores, haciendo la evaluación computacional eficiente.

En el presente apéndice se desarrollan las operaciones que se requieren para implementar a cada uno los operadores involucrados en los métodos desarrollados y que matrices en cada caso son necesarias de construir, siempre teniendo en cuenta el hacer lo más eficiente posible su implementación y evaluación en equipos de cómputo de alto desempeño.

A.1 Matrices Virtuales y Susceptibles de Construir

La gran mayoría de las matrices usadas en los métodos de descomposición de dominio son operadores virtuales, por ejemplo el operador $\underline{\underline{S}}$, que es definido como

$$\underline{\underline{S}} = \underline{\underline{A}}_{\Delta\Delta} - \underline{\underline{A}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{\underline{A}}_{\Pi\Delta} \tag{A.2}$$

y es formado por

$$\underline{\underline{S}} = \sum_{\alpha=1}^{E}\underline{\underline{S}}^{\alpha} \tag{A.3}$$

donde $\underline{\underline{S}}^{\alpha}$ a su vez está constituida por el complemento de Schur local al subdominio Ω_α

$$\underline{\underline{S}}^{\alpha} = \underline{\underline{A}}^{\alpha}_{\Delta\Delta} - \underline{\underline{A}}^{\alpha}_{\Delta\Pi}\left(\underline{\underline{A}}^{\alpha}_{\Pi\Pi}\right)^{-1}\underline{\underline{A}}^{\alpha}_{\Pi\Delta} \tag{A.4}$$

pero, de nuevo, las matrices locales $\underline{\underline{S}}^{\alpha}$ y $\left(\underline{\underline{A}}^{\alpha}_{\Pi\Pi}\right)^{-1}$ no se construyen ya que $\underline{\underline{A}}^{\alpha}_{\Pi\Pi}\underline{u_\Pi} = \underline{\underline{S}}^{\alpha}_{\pi}\underline{u_\Pi}$ donde $\underline{\underline{S}}^{\alpha}_{\pi}$ es definido como

$$\underline{\underline{S}}^{\alpha}_{\pi} = \underline{\underline{A}}^{\alpha}_{\pi\pi} - \underline{\underline{A}}^{\alpha}_{\pi I}\left(\underline{\underline{A}}^{\alpha}_{II}\right)^{-1}\underline{\underline{A}}^{\alpha}_{I\pi}. \tag{A.5}$$

Entonces, las matrices susceptibles de ser construidas en cada subdominio Ω_α son

$$\underline{\underline{A}}^\alpha_{II}, \underline{\underline{A}}^\alpha_{I\pi}, \underline{\underline{A}}^\alpha_{I\Delta}, \underline{\underline{A}}^\alpha_{\pi I}, \underline{\underline{A}}^\alpha_{\pi\pi}, \underline{\underline{A}}^\alpha_{\pi\Delta}, \underline{\underline{A}}^\alpha_{\Delta I}, \underline{\underline{A}}^\alpha_{\Delta\pi} \text{ y } \underline{\underline{A}}^\alpha_{\Delta\Delta} \tag{A.6}$$

pero $\underline{\underline{A}}^\alpha_{I\pi} = \left(\underline{\underline{A}}^\alpha_{\pi I}\right)^T$, $\underline{\underline{A}}^\alpha_{I\Delta} = \left(\underline{\underline{A}}^\alpha_{\Delta I}\right)^T$ y $\underline{\underline{A}}^\alpha_{\pi\Delta} = \left(\underline{\underline{A}}^\alpha_{\Delta\pi}\right)^T$, así, las únicas matrices susceptibles de construir son

$$\underline{\underline{A}}^\alpha_{II}, \underline{\underline{A}}^\alpha_{I\pi}, \underline{\underline{A}}^\alpha_{I\Delta}, \underline{\underline{A}}^\alpha_{\pi\pi}, \underline{\underline{A}}^\alpha_{\pi\Delta} \text{ y } \underline{\underline{A}}^\alpha_{\Delta\Delta} \tag{A.7}$$

donde todas ellas son locales a cada subdominio Ω_α, con una estructura acorde al tipo de discretización discutida en las secciones (3.5 y 6.1) y en este apéndice.

Por lo anterior, nótese que el operador $\underline{\underline{S}}^{-1}$ tampoco se construye, para evaluar $\underline{\underline{S}}^{-1}\underline{u_\Gamma}$ se usa el procedimiento indicado en la sección (A.3).

Otros operadores como son las matrices $\underline{\underline{a}}$ y $\underline{\underline{j}}$ pueden ser o no construidos, pero es necesario recordar que $\underline{\underline{j}} = \underline{\underline{I}} - \underline{\underline{a}}$, así que, en principio $\underline{\underline{j}}$ no sería necesario construir, por otro lado los operadores $\underline{\underline{a}}$ y $\underline{\underline{j}}$ sólo existen en el nodo maestro —donde se controla a los nodos duales y primales y no en los subdominios—; y estas matrices en caso de ser construidas serán dispersas, con pocos valores por renglón —según el número de subdominios que compartan a cada uno de los nodos de la frontera interior— y están sujetas al tipo de triangulación usada en la descomposición de la malla gruesa del dominio.

A.2 Evaluación de la Matriz $\underline{\underline{S}}$ con Nodos Primales Definidos

En todos los casos donde aparece el operador $\underline{\underline{S}}$, ya que, sólo interesa la evaluación de $\underline{\underline{S}}\underline{y_\Gamma}$, i.e. $\underline{v_\Gamma} = \underline{\underline{S}}\underline{y_\Gamma}$, entonces se reescribe al operador $\underline{\underline{S}}$ en términos de sus componentes por subdominio

$$\underline{u_\Gamma} = \left[\sum_{\alpha=1}^{E} \underline{\underline{S}}^\alpha\right] \underline{y_\Gamma} \tag{A.8}$$

para evaluar, el vector $\underline{y_\Gamma}$ se descompone en los subvectores $\underline{y_\Gamma}^\alpha$ correspondientes a cada subdominio $\Omega\alpha$. Así, para evaluar $\underline{\tilde{u}_\Gamma}^\alpha = \underline{\underline{S}}^\alpha \underline{y_\Gamma}^\alpha$ se usa el hecho de que

$$\underline{\underline{S}}^\alpha = \underline{\underline{A}}^\alpha_{\Delta\Delta} - \underline{\underline{A}}^\alpha_{\Delta\Pi}\left(\underline{\underline{A}}^\alpha_{\Pi\Pi}\right)^{-1}\underline{\underline{A}}^\alpha_{\Pi\Delta} \tag{A.9}$$

en donde, se tiene que evaluar

$$\underline{\tilde{u}_\Gamma}^\alpha = \left(\underline{\underline{A}}^\alpha_{\Delta\Delta} - \underline{\underline{A}}^\alpha_{\Delta\Pi}\left(\underline{\underline{A}}^\alpha_{\Pi\Pi}\right)^{-1}\underline{\underline{A}}^\alpha_{\Pi\Delta}\right)\underline{y_\Gamma}^\alpha \tag{A.10}$$

estas evaluaciones en cada subdominio Ω_α pueden realizarse en paralelo.

Para evaluar de forma eficiente esta expresión, se realizan las siguientes operaciones equivalentes

$$\begin{aligned} \underline{x1} &= \underline{\underline{A}}^{\alpha}_{\Delta\Delta}\,\underline{y_{\Gamma}}^{\alpha} \\ \underline{x2} &= \left(\underline{\underline{A}}^{\alpha}_{\Delta\Pi}\left(\underline{\underline{A}}^{\alpha}_{\Pi\Pi}\right)^{-1}\underline{\underline{A}}^{\alpha}_{\Pi\Delta}\right)\underline{y_{\Gamma}}^{\alpha} \\ \underline{\tilde{u}_{\Gamma}}^{\alpha} &= \underline{x1} - \underline{x2} \end{aligned} \tag{A.11}$$

la primera y tercera expresión no tienen ningún problema en su evaluación, para la segunda expresión se tiene que hacer

$$\underline{x3} = \underline{\underline{A}}^{\alpha}_{\Pi\Delta}\,\underline{y_{\Gamma}}^{\alpha} \tag{A.12}$$

con este resultado intermedio se debe calcular

$$\underline{x4} = \left(\underline{\underline{A}}^{\alpha}_{\Pi\Pi}\right)^{-1}\underline{x3} \tag{A.13}$$

pero como no se cuenta con $\left(\underline{\underline{A}}^{\alpha}_{\Pi\Pi}\right)^{-1}$ ya que seria una matriz densa, entonces se multiplica la expresión por $\underline{\underline{A}}^{\alpha}_{\Pi\Pi}$ obteniendo

$$\underline{\underline{A}}^{\alpha}_{\Pi\Pi}\underline{x4} = \underline{\underline{A}}^{\alpha}_{\Pi\Pi}\left(\underline{\underline{A}}^{\alpha}_{\Pi\Pi}\right)^{-1}\underline{x3} \tag{A.14}$$

al simplificar, se obtiene

$$\underline{\underline{A}}^{\alpha}_{\Pi\Pi}\underline{x4} = \underline{x3}. \tag{A.15}$$

Esta última expresión puede ser resuelta usando Gradiente Conjugado o alguna variante de GMRES. Una vez obtenido $\underline{x4}$, se puede calcular

$$\underline{x2} = \underline{\underline{A}}^{\alpha}_{\Delta\Pi}\underline{x4} \tag{A.16}$$

así

$$\underline{\tilde{u}_{\Gamma}}^{\alpha} = \underline{x1} - \underline{x2} \tag{A.17}$$

completando la secuencia de operaciones necesaria para obtener $\underline{\tilde{u}_{\Gamma}}^{\alpha} = \underline{\underline{S}}^{\alpha}\,\underline{y_{\Gamma}}^{\alpha}$.

Observación 8 *En el caso de la expresión dada por la Ec.(A.15) al aplicar un método iterativo, sólo interesará realizar el producto* $\underline{\underline{A}}^{\alpha}_{\Pi\Pi}\underline{x_{\Pi}}$, *o más precisamente* $\underline{\underline{S}}^{i}_{\pi}\underline{x_{\pi}}$, *donde*

$$\underline{\underline{S}}^{i}_{\pi} = \left(\underline{\underline{A}}^{i}_{\pi\pi} - \underline{\underline{A}}^{i}_{\pi I}\left(\underline{\underline{A}}^{i}_{II}\right)^{-1}\underline{\underline{A}}^{i}_{I\pi}\right) \tag{A.18}$$

entonces si se llama $\underline{x_{\pi}}^{i}$ *al vector correspondiente al subdominio* i, *se tiene*

$$\underline{\tilde{u}_{\pi}}^{i} = \left(\underline{\underline{A}}^{i}_{\pi\pi} - \underline{\underline{A}}^{i}_{\pi I}\left(\underline{\underline{A}}^{i}_{II}\right)^{-1}\underline{\underline{A}}^{i}_{I\pi}\right)\underline{x_{\pi}}^{i} \tag{A.19}$$

para evaluar de forma eficiente esta expresión, se realizan las siguientes operaciones equivalentes

$$\begin{aligned} \underline{u1} &= \underline{\underline{A}}^i_{\pi\pi}\underline{x}_i \\ \underline{u2} &= \left(\underline{\underline{A}}^i_{\pi I}\left(\underline{\underline{A}}^i_{II}\right)^{-1}\underline{\underline{A}}^i_{I\pi}\right)\underline{x}_i \\ \underline{\tilde{u}_\Gamma}^i &= \underline{u1} - \underline{u2} \end{aligned} \tag{A.20}$$

la primera y tercera expresión no tienen ningún problema en su evaluación, para la segunda expresión se tiene que hacer

$$\underline{u3} = \underline{\underline{A}}^i_{I\pi}\underline{x}_i \tag{A.21}$$

con este resultado intermedio se debe calcular

$$\underline{u3} = \left(\underline{\underline{A}}^i_{II}\right)^{-1}\underline{u4} \tag{A.22}$$

pero como no se cuenta con $\left(\underline{\underline{A}}^i_{II}\right)^{-1}$*, entonces se multiplica la expresión por* $\underline{\underline{A}}^i_{II}$ *obteniendo*

$$\underline{\underline{A}}^i_{II}\underline{u3} = \underline{\underline{A}}^i_{II}\left(\underline{\underline{A}}^i_{II}\right)^{-1}\underline{u4} \tag{A.23}$$

al simplificar, se obtiene

$$\underline{\underline{A}}^i_{II}\underline{u4} = \underline{u3}. \tag{A.24}$$

Esta última expresión puede ser resuelta usando métodos directos —Factorización LU o Cholesky— o iterativos —Gradiente Conjugado o alguna variante de GMRES— cada una de estas opciones tiene ventajas y desventajas desde el punto de vista computacional —como el consumo de memoria adicional o el aumento de tiempo de ejecución por las operaciones involucradas en cada caso— que deben ser evaluadas al momento de implementar el código para un problema particular. Una vez obtenido $\underline{u4}$*, se puede calcular*

$$\underline{u2} = \underline{\underline{A}}^i_{\pi I}\underline{u4} \tag{A.25}$$

así

$$\underline{\tilde{u}_\Gamma}^i = \underline{u1} - \underline{u2} \tag{A.26}$$

completando la secuencia de operaciones necesaria para obtener $\underline{\underline{S}}^i_\pi\underline{x}_i$*.*

A.3 Evaluación de la Matriz $\underline{\underline{S}}^{-1}$ con Nodos Primales Definidos

En los algoritmos desarrollados interviene el cálculo de $\underline{\underline{S}}^{-1}$, dado que la matriz $\underline{\underline{S}}$ no se construye, entonces la matriz $\underline{\underline{S}}^{-1}$ tampoco se construye. En lugar de ello, se procede de la siguiente manera: Se asume que en las operaciones anteriores al producto de $\underline{\underline{S}}^{-1}$, se ha obtenido un vector. Supóngase que es $\underline{v_\Gamma}$, entonces para hacer

$$\underline{u_\Gamma} = \underline{\underline{S}}^{-1}\underline{v_\Gamma} \tag{A.27}$$

se procede a multiplicar por $\underline{\underline{S}}$ a la ecuación anterior

$$\underline{\underline{S}}\underline{u_\Gamma} = \underline{\underline{S}}\underline{\underline{S}}^{-1}\underline{v_\Gamma} \tag{A.28}$$

obteniendo

$$\underline{\underline{S}}\underline{u_\Gamma} = \underline{v_\Gamma} \tag{A.29}$$

es decir, usando algún proceso iterativo —como CGM, GMRES— se resuelve el sistema anterior, de tal forma que en cada iteración de $\underline{u_\Gamma}^i$ se procede como se indicó en la sección del cálculo de $\underline{\underline{S}}$, resolviendo $\underline{\underline{S}}\underline{u_\Gamma} = \underline{v_\Gamma}$ mediante iteraciones de $\underline{u_\Gamma}^{i+1} = \underline{\underline{S}}\underline{u_\Gamma}^i$.

A.4 Cálculo de los Nodos Interiores

La evaluación de

$$\underline{u_\Pi} = -\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{-1}\underline{\underline{A}}_{\Pi\Delta}\underline{u_\Delta} \tag{A.30}$$

involucra de nuevo cálculos locales de la expresión

$$\underline{u_I}^\alpha = -\left(\underline{\underline{A}}^\alpha_{\Pi\Pi}\right)^{-1}\underline{\underline{A}}^\alpha_{\Pi\Delta}\underline{u_\Gamma}^\alpha \tag{A.31}$$

aquí otra vez está involucrado $\left(\underline{\underline{A}}^\alpha_{\Pi\Pi}\right)^{-1}$, por ello se debe usar el siguiente procedimiento para evaluar de forma eficiente esta expresión, realizando las operaciones equivalentes

$$\begin{aligned} \underline{x4} &= \underline{\underline{A}}^\alpha_{\Pi\Delta}\underline{u_\Gamma}^\alpha \\ \underline{u_I}^\alpha &= \left(\underline{\underline{A}}^\alpha_{\Pi\Pi}\right)^{-1}\underline{x4} \end{aligned} \tag{A.32}$$

multiplicando por $\underline{\underline{A}}^\alpha_{\Pi\Pi}$ la última expresión, se obtiene

$$\underline{\underline{A}}^\alpha_{\Pi\Pi}\underline{u_I}^\alpha = \underline{\underline{A}}^\alpha_{\Pi\Pi}\left(\underline{\underline{A}}^\alpha_{\Pi\Pi}\right)^{-1}\underline{x4} \tag{A.33}$$

simplificando, se obtiene

$$\underline{\underline{A}}^\alpha_{\Pi\Pi}\underline{u_I}^\alpha = \underline{x4} \tag{A.34}$$

esta última expresión puede ser resuelta usando métodos directos —como Factorización LU o Cholesky— o mediante métodos iterativos —como Gradiente Conjugado o alguna variante de GMRES— como se indica en la observación (8).

A.5 Descomposición de Schur sin Nodos Primales

En el caso de que en la descomposición no se usen nodos primales, la matriz virtual global $\underline{\underline{A}}$ queda como

$$\underline{\underline{A}} = \begin{pmatrix} \underline{\underline{A}}^1_{II} & \underline{\underline{0}} & \cdots & \underline{\underline{0}} & \underline{\underline{A}}^1_{I\Delta} \\ \underline{\underline{0}} & \underline{\underline{A}}^2_{II} & \cdots & \underline{\underline{0}} & \underline{\underline{A}}^2_{I\Delta} \\ \underline{\underline{0}} & \underline{\underline{0}} & \ddots & \underline{\underline{0}} & \vdots \\ \underline{\underline{0}} & \underline{\underline{0}} & \cdots & \underline{\underline{A}}^E_{II} & \underline{\underline{A}}^E_{I\Delta} \\ \underline{\underline{A}}^1_{\Delta I} & \underline{\underline{A}}^2_{\Delta I} & \cdots & \underline{\underline{A}}^E_{\Delta I} & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix} \tag{A.35}$$

de donde $\underline{\underline{A}}\underline{x} = \underline{b}$ se implementa como

$$\underline{\underline{A}} \begin{pmatrix} \underline{x}_I \\ \underline{x}_\Delta \end{pmatrix} = \begin{pmatrix} \underline{b}_I \\ \underline{b}_\Delta \end{pmatrix} \tag{A.36}$$

i.e.

$$\left(\sum_{\alpha=1}^{E} \left(\underline{\underline{A}}^\alpha_{\Delta\Delta} - \underline{\underline{A}}^\alpha_{\Delta I} \left(\underline{\underline{A}}^\alpha_{II} \right)^{-1} \underline{\underline{A}}^\alpha_{I\Delta} \right) \right) \underline{x}_\Delta = \underline{b}_\Delta - \sum_{\alpha=1}^{E} \left(\underline{\underline{A}}^\alpha_{\Delta I} \left(\underline{\underline{A}}^\alpha_{II} \right)^{-1} \underline{b}^\alpha_I \right) \tag{A.37}$$

una vez encontrado $\underline{x}_\Delta$, se encuentra $\underline{x}_I$ mediante

$$\underline{x}^\alpha_I = \left(\underline{\underline{A}}^\alpha_{II} \right)^{-1} \left(\underline{b}^\alpha_I - \underline{\underline{A}}^\alpha_{I\Delta} \underline{x}^\alpha_\Delta \right). \tag{A.38}$$

A.6 DVS para Ecuaciones Escalares y Vectoriales

Con el fin de ejemplificación, se supone una ecuación escalar en dos dimensiones definida en un dominio Ω, mediante una descomposición en subdominios usando una malla estructurada cartesiana como se muestra en la figura:

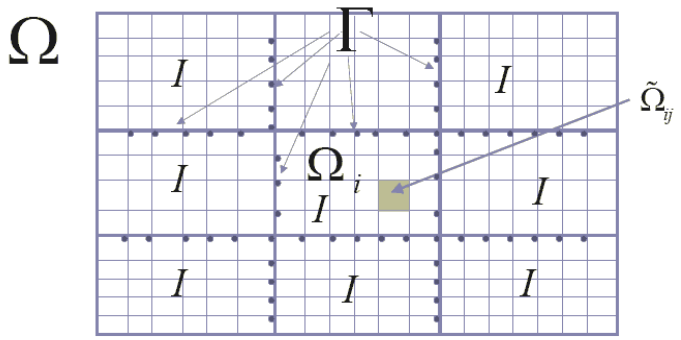

Figura 19: Dominio Ω descompuesto en una partición gruesa de 3×3 y cada subdominio Ω_i en una partición fina de 6×5.

En este caso, sería una descomposición del dominio Ω en 3×3 y 6×5 , la malla gruesa seria descompuesta en $3 \times 3 = 9$ subdominios Ω_α y donde cada

uno de ellos tiene una partición fina de 6×5 i.e. $42 = (6+1)*(5+1)$ grados de libertad por subdominio. En el caso vectorial, en el cual cada grado de libertad contiene C componentes, el número de grados de libertad total será igual a $((6+1)*(5+1)*C)$. Por ejemplo, en el caso de $C = 3$ se tienen 126 grados de libertad por subdominio (véase [45]).

EDP Vectoriales en Dos y Tres Dimensiones En el caso de una ecuación vectorial en dos —tres— dimensiones, si el dominio Ω es descompuesto en una malla estructurada cartesiana, donde la partición gruesa de $n \times m$ —ó $n \times m \times o$— subdominios Ω_α y donde cada uno es descompuesto en una partición fina de $r \times s$ —ó $r \times s \times t$—, en el cual cada grado de libertad contiene C componentes, el número de grados de libertad por subdominio es $(r+1)*(s+1)*C$ —ó $(r+1)*(s+1)*(t+1)*C$—.

A partir de la formulación de los métodos desarrollados, se generan las matrices locales en cada subdominio, de esta forma se obtiene la descomposición fina del dominio, generándose de manera virtual el sistema lineal definido por el método DVS que se este implementando.

La implementación computacional que se desarrolló tiene una jerarquía de clases en donde la clase *DPMethod* realiza la partición gruesa del dominio usando la clase *Interchange* y controla la partición de cada subdominio mediante un objeto de la clase de *RectSub* generando la partición fina del dominio. La resolución de los nodos de la frontera interior se hace mediante el método de CGM o alguna variante de GMRES.

El método de descomposición de dominio en el espacio de vectores derivados se implementó realizando las siguientes tareas:

A) La clase *DPMethod* genera la descomposición gruesa del dominio mediante la agregación de un objeto de la clase *Interchange,* se supone que se tiene particionado en $n \times m$ —ó $n \times m \times o$— subdominios.

B) Con esa geometría se construyen los objetos de *RectSub* —uno por cada subdominio Ω_α—, donde cada subdominio es particionado en $r \times s$ —ó $r \times s \times t$— subdominios y se regresan las coordenadas de los nodos de frontera del subdominio correspondiente a la clase *DPMethod*.

C) Con estas coordenadas, la clase *DPMethod* conoce a los nodos de la frontera interior —son estos los que resuelve el método de descomposición de dominio—. Las coordenadas de los nodos de la frontera interior se dan a conocer a los objetos *RectSub*, transmitiendo sólo aquellos que están en su subdominio.

D) Después de conocer los nodos de la frontera interior, cada objeto *RectSub* calcula las matrices locales sin realizar comunicación alguna. Al terminar de calcular las matrices se avisa a la clase *DPMethod* de la finalización de los cálculos.

E) Mediante la comunicación de vectores del tamaño del número de nodos de la frontera interior entre la clase *DPMethod* y los objetos *RectSub*, se prepara todo lo necesario para empezar el método de CGM o GMRES y resolver el sistema lineal virtual.

F) Para aplicar el método de CGM o GMRES, en cada iteración se transmite un vector del tamaño del número de nodos de la frontera interior para que en cada objeto se realicen las operaciones pertinentes y resolver así el sistema algebraico asociado, esta comunicación se realiza de ida y vuelta entre la clase *DPMethod* y los objetos *RectSub* tantas veces como iteraciones haga el método. Resolviendo con esto los nodos de la frontera interior $\underline{u_\Gamma}_i$.

G) Al término de las iteraciones, se pasa la solución $\underline{u_\Gamma}_i$ de los nodos de la frontera interior que pertenecen a cada subdominio dentro de cada objeto *RectSub* para que se resuelvan los nodos locales al subdominio $\underline{u_\pi}^\alpha = -\left(\underline{\underline{A}}^\alpha_{\Pi\Pi}\right)^{-1} \underline{\underline{A}}^\alpha_{\Pi\Delta} \underline{u_\Gamma}^\alpha$, sin realizar comunicación alguna en el proceso, al concluir se avisa a la clase *DDM* de ello.

I) La clase *DPMethod* mediante un último mensaje avisa que se concluya el programa, terminado así el esquema Maestro-Esclavo.

Análisis de Comunicaciones Para hacer un análisis de las comunicaciones en una ecuación vectorial con C componentes, entre el nodo principal y los nodos esclavos en el método DVS es necesario conocer qué se trasmite y su tamaño, es por ello que en la medida de lo posible se detallan las comunicaciones existentes —hay que hacer mención que entre los nodos esclavos no hay comunicación alguna—.

Tomando la descripción del algoritmo detallado anteriormente, en donde se supuso una partición del dominio Ω con una malla estructurada cartesiana en $n \times m$ —ó $n \times m \times o$ en tres dimensiones— para la malla gruesa y $r \times s$ —ó $r \times s \times t$— para la malla fina, las comunicaciones correspondientes a cada inciso son:

A) El nodo maestro transmite 2 coordenadas en dos —en tres– dimensiones correspondientes a la delimitación del subdominio.

B) $2 * (r + 1) * (s + 1) * C$ —ó $2 * (r + 1) * (s + 1) * (t + 1) * C$— coordenadas transmite cada subdominio al nodo maestro.

C) A lo más $n*m*2*(r+1)*(s+1)*C$ —ó $n*m*o*2*(r+1)*(s+1)*(t+1)*C$— coordenadas son las de los nodos de la frontera interior, y sólo aquellas correspondientes a cada subdominio son trasmitidas por el nodo maestro a los subdominios en los esclavos siendo estas a lo más $2 * (n * m) * (r * s) * C$ —ó $2 * (n * m * o) * (r * s * t) * C$— coordenadas.

D) Sólo se envía un aviso de la conclusión del cálculo de las matrices.

E) A lo más $2*(r+1)*(s+1)*C$ —ó $2*(r+1)*(s+1)*(t+1)*C$— coordenadas son transmitidas a los subdominios en los nodos esclavos desde el nodo maestro y los nodos esclavos transmiten al nodo maestro esa misma cantidad información.

F) A lo más $2*(r+1)*(s+1)*C$ —ó $2*(r+1)*(s+1)*(t+1)*C$— coordenadas son transmitidas a los subdominios en los nodos esclavos y estos retornan un número igual al nodo maestro por iteración del método de CGM o alguna variante de GMRES.

G) A lo más $2*(r+1)*(s+1)*C$ —ó $2*(r+1)*(s+1)*(t+1)*C$— valores de la solución de la frontera interior son transmitidas a los subdominios en los nodos esclavos desde el nodo maestro y cada objeto transmite un único aviso de terminación.

I) El nodo maestro manda un aviso a cada subdominio en los nodos esclavos para concluir con el esquema.

En todos los casos, la transmisión se realiza mediante paso de arreglos de enteros y números de punto flotante que varían de longitud pero siempre son cantidades pequeñas de estos y se transmiten en forma de bloque, por ello las comunicaciones son eficientes.

EDP Escalares como un Caso Particular de EDP Vectoriales En el caso de trabajar con ecuaciones escalares en dos o tres dimensiones con una malla estructurada cartesiana, en vez de ecuaciones vectoriales, todo el análisis anterior continua siendo válido y sólo es necesario tomar el número de componentes C igual a uno, manteniéndose la estructura del código intacta.

Esto le da robustez al código, pues permite trabajar con problemas escalares y vectoriales con un pequeño cambio en la estructura del programa, permitiendo resolver una gama más grande de problemas.

Tamaño de las Matrices Locales. Ahora, si se supone que el dominio $\Omega \subset \mathbb{R}^3$ es descompuesto con una malla estructurada cartesiana en una partición gruesa de $n \times m \times o$ subdominios Ω_α y cada subdominio Ω_α es descompuesto en una partición fina de $r \times s \times t$, con número de componentes igual a C, entonces el número de grados de libertad por subdominio es $(r+1)*(s+1)*(t+1)*C$. En la cual se usa un ordenamiento estándar en la numeración de los nodos dentro de cada subdominio

El número de nodos interiores es

$$N^I = (r-1)*(s-1)*(t-1) \tag{A.39}$$

el número de nodos primales, suponiendo que se usa restricción en los vértices es

$$N^\pi = (n-1)*(m-1)*(o-1) \tag{A.40}$$

y el número de nodos duales es a lo más

$$N^\Delta = 2*[((r-1)+(s-1))*(t-1)] \tag{A.41}$$

entonces se tiene que el tamaño y la estructura de cada una de las matrices locales

$$\underline{\underline{A}}^i_{II}, \underline{\underline{A}}^i_{I\pi}, \underline{\underline{A}}^i_{I\Delta}, \underline{\underline{A}}^i_{\pi I}, \underline{\underline{A}}^i_{\pi\pi}, \underline{\underline{A}}^i_{\pi\Delta}, \underline{\underline{A}}^i_{\Delta I}, \underline{\underline{A}}^i_{\Delta\pi} \text{ y } \underline{\underline{A}}^i_{\Delta\Delta}$$

generadas por la descomposición fina del subdominio será:

- $\underline{\underline{A}}^i_{II}$ es una matriz cuadrada de $N^I \times N^I$ nodos, con una estructura bandada
- $\underline{\underline{A}}^i_{I\pi}$ es una matriz rectangular de $N^I \times N^\pi$ nodos, con una estructura dispersa
- $\underline{\underline{A}}^i_{\pi I}$ es una matriz rectangular de $N^\pi \times N^I$ nodos, con una estructura dispersa y transpuesta de $\underline{\underline{A}}^i_{I\pi}$
- $\underline{\underline{A}}^i_{I\Delta}$ es una matriz rectangular de $N^I \times N^\Delta$ nodos, con una estructura dispersa
- $\underline{\underline{A}}^i_{\Delta I}$ es una matriz rectangular de $N^\Delta \times N^I$ nodos, con una estructura dispersa y transpuesta de $\underline{\underline{A}}^i_{I\Delta}$
- $\underline{\underline{A}}^i_{\pi\Delta}$ es una matriz rectangular de $N^\pi \times N^\Delta$ nodos, con una estructura dispersa
- $\underline{\underline{A}}^i_{\Delta\pi}$ es una matriz rectangular de $N^\pi \times N^\Delta$ nodos, con una estructura dispersa y transpuesta de $\underline{\underline{A}}^i_{\pi\Delta}$
- $\underline{\underline{A}}^i_{\pi\pi}$ es una matriz cuadrada de $N^\pi \times N^\pi$ nodos, con una estructura bandada
- $\underline{\underline{A}}^i_{\Delta\Delta}$ es una matriz cuadrada de $N^\Delta \times N^\Delta$ nodos, con una estructura bandada

Para información sobre la estructura de las matrices bandadas véase la sección (B.3.1) y para matrices dispersas véase la sección (B.3.2).

A.7 Método de Descomposición de Dominio de Subestructuración

La solución numérica por los esquemas tradicionales de discretización tipo Elemento Finito y Diferencias Finitas generan una discretización del problema, la cual es usada para generar un sistema de ecuaciones algebraicos $\underline{\underline{A}}\underline{u} = \underline{b}$. Este sistema algebraico en general es de gran tamaño para problemas reales.

En esta sección y sin pérdida de generalidad se considerarán problemas con valor en la frontera (VBVP) de la forma

$$\begin{aligned} \mathcal{L}u &= f \quad \text{en } \Omega \\ u &= g \quad \text{sobre } \partial\Omega \end{aligned} \tag{A.42}$$

donde

$$\mathcal{L}u = -\nabla \cdot \underline{\underline{a}} \cdot \nabla u + cu \tag{A.43}$$

con $\underline{\underline{a}}$ una matriz positiva definida, simétrica y $c \geq 0$, como un caso particular del operador elíptico de orden 2 y para ejemplificar se toma un dominio $\Omega \subset R^2$ con fronteras poligonales, es decir, Ω es un conjunto abierto acotado y conexo tal que su frontera $\partial\Omega$ es la unión de un número finito de polígonos.

Si multiplico a la ecuación $-\nabla \cdot \underline{\underline{a}} \cdot \nabla u + cu = f_\Omega$ por $v \in V = H_0^1(\Omega)$, se obtiene

$$-v\left(\nabla \cdot \underline{\underline{a}} \cdot \nabla u + cu\right) = v f_\Omega \tag{A.44}$$

aplicando el teorema de Green se obtiene

$$\int_\Omega \left(\nabla v \cdot \underline{\underline{a}} \cdot \nabla u + cuv\right) d\underline{x} = \int_\Omega v f_\Omega d\underline{x}. \tag{A.45}$$

Definiendo el operador bilineal

$$a(u,v) = \int_\Omega \left(\nabla v \cdot \underline{\underline{a}} \cdot \nabla u + cuv\right) d\underline{x} \tag{A.46}$$

y la funcional lineal

$$l(v) = \langle f, v\rangle = \int_\Omega v f_\Omega d\underline{x} \tag{A.47}$$

entonces se puede reescribir el problema en forma variacional, haciendo uso de la forma bilineal $a(\cdot,\cdot)$ y la funcional lineal $l(\cdot)$.

Donde las funciones lineales definidas por tramos en Ω_e en este caso serán polinomios de orden uno en cada variable de forma separada y cuya restricción de ϕ_i a Ω_e es $\phi_i^{(e)}$. Para simplificar los cálculos en esta etapa, se supone que la matriz $\underline{\underline{a}} = a\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, entonces se tiene que la integral del lado izquierdo de la Ec.(A.45) queda escrita como

$$\int_\Omega \left(a\nabla\phi_i \cdot \nabla\phi_j + c\phi_i\phi_j\right) dxdy = \int_\Omega f_\Omega \phi_j dxdy \tag{A.48}$$

donde

$$\begin{aligned} K_{ij} &= \int_\Omega \left(a\nabla\phi_i \cdot \nabla\phi_j + c\phi_i\phi_j\right) dxdy \qquad (A.49)\\ &= \sum_{e=1}^{E} \int_{\Omega_e} \left(a\nabla\phi_i^{(e)} \cdot \nabla\phi_j^{(e)} + c\phi_i^{(e)}\phi_j^{(e)}\right) dxdy \\ &= \sum_{e=1}^{E} \int_{\Omega_e} \left(a\left[\frac{\partial\phi_i^{(e)}}{\partial x}\frac{\partial\phi_j^{(e)}}{\partial x} + \frac{\partial\phi_i^{(e)}}{\partial y}\frac{\partial\phi_j^{(e)}}{\partial y}\right] + c\phi_i^{(e)}\phi_j^{(e)}\right) dxdy \end{aligned}$$

y el lado derecho como

$$\begin{aligned} F_j &= \int_\Omega f_\Omega \phi_j dxdy \\ &= \sum_{e=1}^{E} \int_{\Omega_e} f_\Omega \phi_j^{(e)} dxdy. \end{aligned} \tag{A.50}$$

Se considera el problema dado por la Ec.(A.42) en el dominio Ω, el cual es subdividido en E subdominios Ω_i, $i = 1, 2, ..., E$ sin traslape, también conocida como malla gruesa $\mathcal{T}_H$, es decir

$$\Omega_i \cap \Omega_j = \varnothing \quad \forall i \neq j \quad \text{y} \quad \overline{\Omega} = \bigcup_{i=1}^{E} \overline{\Omega}_i, \tag{A.51}$$

y al conjunto

$$\Gamma = \bigcup_{i=1}^{E} \Gamma_i, \quad \text{si } \Gamma_i = \partial\Omega_i \backslash \partial\Omega \tag{A.52}$$

lo llamo la frontera interior del dominio Ω.

Un ejemplo de un dominio Ω y su descomposición en subdominios Ω_i y cada Ω_i a su vez descompuesto en Ω_e subdominios. Sin pérdida de generalidad se toma $g = 0$ en $\partial\Omega$, nótese que siempre es posible poner el problema de la Ec.(A.42) como uno con condiciones de frontera Dirichlet que se nulifiquen mediante la adecuada manipulación del término del lado derecho de la ecuación.

Primeramente sea $D \subset H_0^1(\Omega)$ un espacio lineal de funciones de dimensión finita N, en el cual esté definido un producto interior denotado para cada $u, v \in D$ por

$$u \cdot v = \langle u, v \rangle . \tag{A.53}$$

Considerando la existencia de los subconjuntos linealmente independientes

$$\begin{gathered} \mathcal{B} \subset \tilde{D}, \mathcal{B}_I \subset \tilde{D}_I, \mathcal{B}_\Gamma \subset \tilde{D}_\Gamma \\ \mathcal{B}_\Gamma \subset \tilde{D}_\Gamma, \mathcal{B}_{\Gamma J} \subset \tilde{D}_{\Gamma 1}, \mathcal{B}_{\Gamma M} \subset \tilde{D}_{\Gamma 2} \end{gathered} \tag{A.54}$$

los cuales satisfacen

$$\mathcal{B} = \mathcal{B}_I \cup \mathcal{B}_\Gamma \text{ y } \bar{\mathcal{B}}_\Gamma = \mathcal{B}_{\Gamma J} \cup \mathcal{B}_{\Gamma M} \tag{A.55}$$

el espacio generado por cada uno de los subconjuntos $\mathcal{B}_\Gamma$ y $\bar{\mathcal{B}}_\Gamma$ es $\tilde{D}_\Gamma$, sin embargo, nótese la propiedad de que los miembros de $\mathcal{B}_\Gamma$ tienen soporte local.

Se definen las bases

$$\mathcal{B}_I = \left\{ w_I^1, ..., w_I^{\tilde{N}_I} \right\}, \mathcal{B}_{\Gamma M} = \left\{ w_M^1, ..., w_M^{\tilde{N}_\Gamma} \right\} \text{ y } \mathcal{B}_{\Gamma J} = \left\{ w_J^1, ..., w_J^{\tilde{N}_\Gamma} \right\} \tag{A.56}$$

de las funcionales lineales ϕ_i en Ω.

Entonces definiendo para toda $\delta = 1, ..., K$, la matriz de $N_\delta \times N_\delta$

$$\underline{\underline{A}}_{II}^{\delta} \equiv \left[\left\langle w_I^i, w_I^j \right\rangle \right] \tag{A.57}$$

que sólo esta definida en cada subespacio (subdominio Ω_δ). Entonces, la matriz virtual $\underline{\underline{A}}_{II}$ es dada por la matriz diagonal de la forma

$$\underline{\underline{A}}_{II} \equiv \begin{bmatrix} \underline{\underline{A}}_{II}^{1} & & & \\ & \underline{\underline{A}}_{II}^{2} & & \\ & & \ddots & \\ & & & \underline{\underline{A}}_{II}^{E} \end{bmatrix} \tag{A.58}$$

donde el resto de la matriz fuera de la diagonal en bloques es cero.

De forma similar se define

$$\underline{\underline{A}}_{I\Gamma}^{\delta} \equiv \left[\left\langle w_I^i, w_\Gamma^\alpha \right\rangle \right], \quad \underline{\underline{A}}_{\Gamma I}^{\delta} \equiv \left[\left\langle w_\Gamma^\alpha, w_I^i \right\rangle \right] \tag{A.59}$$

y

$$\underline{\underline{A}}_{\Gamma\Gamma}^{\delta} \equiv \left[\langle w_\Gamma^\alpha, w_\Gamma^\alpha \rangle \right] \tag{A.60}$$

para toda $\delta = 1, ..., E$, obsérvese que como $\bar{\mathcal{B}}_\Gamma = \mathcal{B}_{\Gamma J} \cup \mathcal{B}_{\Gamma M}$ entonces

$$\underline{\underline{A}}_{\Gamma\Gamma}^{\delta} = \left[\langle w_\Gamma^\alpha, w_\Gamma^\alpha \rangle \right] = \left[\left\langle w_{\Gamma_J}^\alpha, w_{\Gamma_J}^\alpha \right\rangle \right] + \left[\left\langle w_{\Gamma_M}^\alpha, w_{\Gamma_M}^\alpha \right\rangle \right] \tag{A.61}$$

también que $\underline{\underline{A}}_{I\Gamma}^{\delta} = \left(\underline{\underline{A}}_{\Gamma I}^{\delta} \right)^T$. Entonces las matrices virtuales $\underline{\underline{A}}_{\Gamma I}, \underline{\underline{A}}_{I\Gamma}$ y $\underline{\underline{A}}_{\Gamma\Gamma}$ quedarán definidas como

$$\underline{\underline{A}}_{I\Gamma} \equiv \begin{bmatrix} \underline{\underline{A}}_{I\Gamma}^{1} \\ \underline{\underline{A}}_{I\Gamma}^{2} \\ \vdots \\ \underline{\underline{A}}_{I\Gamma}^{E} \end{bmatrix} \tag{A.62}$$

$$\underline{\underline{A}}_{\Gamma I} \equiv \begin{bmatrix} \underline{\underline{A}}_{\Gamma I}^{1} & \underline{\underline{A}}_{\Gamma I}^{2} & \cdots & \underline{\underline{A}}_{\Gamma I}^{E} \end{bmatrix} \tag{A.63}$$

y

$$\underline{\underline{A}}_{\Gamma\Gamma} \equiv \left[\sum_{i=1}^{E} \underline{\underline{A}}_{\Gamma\Gamma}^{i} \right] \tag{A.64}$$

donde $\left[\sum_{i=1}^{E} \underline{\underline{A}}_{\Gamma\Gamma}^{i} \right]$ es construida sumando las $\underline{\underline{A}}_{\Gamma\Gamma}^{i}$ según el orden de los nodos globales versus los nodos locales.

También se considera al vector $\underline{u} \equiv (u_1, .., u_E)$ el cual se escribe como $\underline{u} = (\underline{u_I}, \underline{u_\Gamma})$ donde $\underline{u_I} = (u_1, ..., _{N_I})$ y $\underline{u_\Gamma} = (u_1, ..., _{N_\Gamma})$.

Así, el sistema virtual

$$\begin{aligned} \underline{\underline{A}}_{II} \underline{u_I} + \underline{\underline{A}}_{I\Gamma} \underline{u_\Gamma} &= \underline{b_I} \\ \underline{\underline{A}}_{\Gamma I} \underline{u_I} + \underline{\underline{A}}_{\Gamma\Gamma} \underline{u_\Gamma} &= \underline{b_\Gamma} \end{aligned} \tag{A.65}$$

quedando expresado como

$$\begin{bmatrix} \underline{\underline{A}}^1_{II} & & \\ & \ddots & \\ & & \underline{\underline{A}}^E_{II} \end{bmatrix} \begin{bmatrix} \underline{u_I}_1 \\ \vdots \\ \underline{u_I}_E \end{bmatrix} + \begin{bmatrix} \underline{\underline{A}}^1_{I\Gamma} \\ \vdots \\ \underline{\underline{A}}^E_{I\Gamma} \end{bmatrix} \begin{bmatrix} \underline{u_\Gamma}_1 \\ \vdots \\ \underline{u_\Gamma}_E \end{bmatrix} = \begin{bmatrix} \underline{b_I}_1 \\ \vdots \\ \underline{b_I}_E \end{bmatrix}$$

$$\begin{bmatrix} \underline{\underline{A}}^1_{\Gamma I} & \cdots & \underline{\underline{A}}^E_{\Gamma I} \end{bmatrix} \begin{bmatrix} \underline{u_I}_1 \\ \vdots \\ \underline{u_I}_E \end{bmatrix} + \begin{bmatrix} \underline{\underline{A}}^{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} \underline{u_\Gamma}_1 \\ \vdots \\ \underline{u_\Gamma}_E \end{bmatrix} = \begin{bmatrix} \underline{b_\Gamma}_1 \\ \vdots \\ \underline{b_\Gamma}_E \end{bmatrix}$$

o más compactamente como $\underline{\underline{A}}\underline{u} = \underline{b}$, nótese que las matrices $\underline{\underline{A}}^i_{\Gamma\Gamma}, \underline{\underline{A}}^i_{\Gamma I}, \underline{\underline{A}}^i_{I\Gamma}$ y $\underline{\underline{A}}^i_{II}$ son matrices bandadas.

Si ahora despejo $\underline{u_I}$ de la primera ecuación del sistema dado por la Ec.(A.65), se obtiene

$$\underline{u_I} = \left(\underline{\underline{A}}_{II}\right)^{-1} \left(\underline{b_I} - \underline{\underline{A}}_{I\Gamma}\underline{u_\Gamma}\right) \tag{A.66}$$

si sustituyo $\underline{u_I}$ en la segunda ecuación del sistema dado por la Ec.(A.65) entonces se obtiene

$$\left(\underline{\underline{A}}_{\Gamma\Gamma} - \underline{\underline{A}}_{\Gamma I}\left(\underline{\underline{A}}_{II}\right)^{-1}\underline{\underline{A}}_{I\Gamma}\right)\underline{u_\Gamma} = \underline{b_\Gamma} - \underline{\underline{A}}_{\Gamma I}\left(\underline{\underline{A}}_{II}\right)^{-1}\underline{b_I} \tag{A.67}$$

en la cual los nodos interiores no figuran en la ecuación y todo queda en función de los nodos de la frontera interior $\underline{u_\Gamma}$.

A la matriz formada por $\underline{\underline{A}}_{\Gamma\Gamma} - \underline{\underline{A}}_{\Gamma I}\left(\underline{\underline{A}}_{II}\right)^{-1}\underline{\underline{A}}_{I\Gamma}$ se le conoce como el complemento de Schur global y se le denota como

$$\underline{\underline{S}} = \underline{\underline{A}}_{\Gamma\Gamma} - \underline{\underline{A}}_{\Gamma I}\left(\underline{\underline{A}}_{II}\right)^{-1}\underline{\underline{A}}_{I\Gamma}. \tag{A.68}$$

En este caso, como estoy planteando todo en términos de subdominios Ω_i, con $i = 1, ..., E$, entonces las matrices $\underline{\underline{A}}^i_{\Gamma\Gamma}, \underline{\underline{A}}^i_{\Gamma I}, \underline{\underline{A}}^i_{I\Gamma}$ y $\underline{\underline{A}}^i_{II}$ quedan definidas de manera local, así que se procede a definir el complemento de Schur local como

$$\underline{\underline{S}}_i = \underline{\underline{A}}^i_{\Gamma\Gamma} - \underline{\underline{A}}^i_{\Gamma I}\left(\underline{\underline{A}}^i_{II}\right)^{-1}\underline{\underline{A}}^i_{I\Gamma} \tag{A.69}$$

de forma adicional se define

$$\underline{b}_i = \underline{b_\Gamma}_i - \underline{\underline{A}}^i_{\Gamma I}\left(\underline{\underline{A}}^i_{II}\right)^{-1}\underline{b_I}_i. \tag{A.70}$$

El sistema dado por la Ec.(A.67) se escribe como

$$\underline{\underline{S}}\underline{u_\Gamma} = \underline{b} \tag{A.71}$$

y queda definido de manera virtual a partir de

$$\left[\sum_{i=1}^{E} \underline{\underline{S}}_i\right] \underline{u_\Gamma} = \left[\sum_{i=1}^{E} \underline{b}_i\right] \tag{A.72}$$

donde $\left[\sum_{i=1}^{E} \underline{\underline{S}}_i\right]$ y $\left[\sum_{i=1}^{E} \underline{b}_i\right]$ podrían ser construida sumando las S_i y b_i respectivamente según el orden de los nodos globales versus los nodos locales.

El sistema lineal virtual obtenido de la Ec.(A.71) se resuelve —dependiendo de si es o no simétrico— eficientemente usando el método de Gradiente Conjugado o alguna variante de GMRES, para ello no es necesario construir la matriz $\underline{\underline{S}}$ con las contribuciones de cada S_i correspondientes al subdominio i, lo que se hace es pasar a cada subdominio el vector $\underline{u_\Gamma}_i$ correspondiente a la i-ésima iteración del método iterativo usado, para que en cada subdominio se evalué $\underline{\tilde{u}_\Gamma}^i = \underline{\underline{S}}_i \underline{u_\Gamma}_i$ localmente y con el resultado se forma el vector $\underline{\tilde{u}_\Gamma} = \sum_{i=1}^{E} \underline{\tilde{u}_\Gamma}_i$ y se continué con los demás pasos del método. Esto es ideal para una implementación en paralelo del método de Gradiente Conjugado o variante de GMRES.

Una vez resuelto el sistema de la Ec.(A.72) en el que se ha encontrado la solución para los nodos de la frontera interior $\underline{u_\Gamma}$, entonces debo resolver localmente los $\underline{u_I}_i$ correspondientes a los nodos interiores para cada subespacio Ω_i, para esto empleo

$$\underline{u_I}_i = \left(\underline{\underline{A}}^i_{II}\right)^{-1} \left(\underline{b_I}_i - \underline{\underline{A}}^i_{I\Gamma} \underline{u_\Gamma}_i\right) \tag{A.73}$$

para cada $i = 1, 2, ..., E$, quedando así resuelto el problema $\underline{\underline{A}}\underline{u} = \underline{b}$ tanto en los nodos interiores $\underline{u_I}_i$ como en los de la frontera interior $\underline{u_\Gamma}_i$ correspondientes a cada subespacio Ω_i.

Observación 9 *Nótese que normalmente las matrices locales* $\underline{\underline{S}}_i$ *y* $\left(\underline{\underline{A}}^i_{II}\right)^{-1}$ *no se construyen, ya que estas serian matrices densas y su construcción es computacionalmente costosa, y como sólo interesa el producto* $\underline{\underline{S}}\underline{y_\Gamma}$*, o más precisamente* $\left[\sum_{i=1}^{E} \underline{\underline{S}}_i\right] \underline{y_\Gamma}$*, entonces si llamo* $\underline{y_\Gamma}_i$ *al vector correspondiente al subdominio* i*, entonces se obtiene*

$$\underline{\tilde{u}_\Gamma}^i = \left(\underline{\underline{A}}^i_{\Gamma\Gamma} - \underline{\underline{A}}^i_{\Gamma I} \left(\underline{\underline{A}}^i_{II}\right)^{-1} \underline{\underline{A}}^i_{I\Gamma}\right) \underline{y_\Gamma}_i. \tag{A.74}$$

Para evaluar de forma eficiente esta expresión, se realizan las siguientes operaciones equivalentes

$$\begin{aligned} \underline{x1} &= \underline{\underline{A}}^i_{\Gamma\Gamma} \underline{y_\Gamma}_i \\ \underline{x2} &= \left(\underline{\underline{A}}^i_{\Gamma I} \left(\underline{\underline{A}}^i_{II}\right)^{-1} \underline{\underline{A}}^i_{I\Gamma}\right) \underline{y_\Gamma}_i \\ \underline{\tilde{u}_\Gamma}^i &= \underline{x1} - \underline{x2} \end{aligned} \tag{A.75}$$

la primera y tercera expresión no tienen ningún problema en su evaluación, para la segunda expresión se debe evaluar

$$\underline{x3} = \underline{\underline{A}}^i_{I\Gamma} \underline{y_\Gamma}_i \tag{A.76}$$

con este resultado intermedio se debe calcular

$$\underline{x4} = \left(\underline{\underline{A}}^i_{II}\right)^{-1} \underline{x3} \tag{A.77}$$

pero como no se cuenta con $\left(\underline{\underline{A}}^i_{II}\right)^{-1}$*, entonces multiplico la expresión por* $\underline{\underline{A}}^i_{II}$ *obteniendo*

$$\underline{\underline{A}}^i_{II} \underline{x4} = \underline{\underline{A}}^i_{II} \left(\underline{\underline{A}}^i_{II}\right)^{-1} \underline{x3} \tag{A.78}$$

al simplificar, se obtiene

$$\underline{\underline{A}}^i_{II} \underline{x4} = \underline{x3}. \tag{A.79}$$

Esta última expresión puede ser resuelta usando Factorización LU, Gradiente Conjugado o alguna variante de GMRES —cada una de estas opciones tiene ventajas y desventajas computacionales que deben ser evaluadas al momento de implementar el código para un problema particular—. Una vez obtenido $\underline{x4}$*, se puede calcular*

$$\underline{x2} = \underline{\underline{A}}^i_{\Gamma I} \underline{x4} \tag{A.80}$$

así

$$\underline{\tilde{u}_\Gamma}^i = \underline{x1} - \underline{x2} \tag{A.81}$$

completando la secuencia de operaciones necesarias para obtener $\underline{\underline{S}}_i \underline{y_\Gamma}_i$.

Observación 10 *En el caso del cálculo de*

$$\underline{b}_i = \underline{b_\Gamma}_i - \underline{\underline{A}}^i_{\Gamma I} \left(\underline{\underline{A}}^i_{II}\right)^{-1} \underline{b_I}_i \tag{A.82}$$

algo análogo al comentario anterior deberá de hacerse, ya que de nuevo está involucrado $\left(\underline{\underline{A}}^i_{II}\right)^{-1}$*, por ello se debe usar el siguiente procedimiento para evaluar de forma eficiente esta expresión, realizando las operaciones equivalentes*

$$\underline{y1} = \left(\underline{\underline{A}}^i_{II}\right)^{-1} \underline{b_I}_i \tag{A.83}$$

multiplicando por $\underline{\underline{A}}^i_{II}$ *a la última expresión, se obtiene*

$$\underline{\underline{A}}^i_{II} \underline{y1} = \underline{\underline{A}}^i_{II} \left(\underline{\underline{A}}^i_{II}\right)^{-1} \underline{b_I}_i \tag{A.84}$$

simplificando, se obtiene

$$\left(\underline{\underline{A}}^i_{II}\right) \underline{y1} = \underline{b_I}_i \tag{A.85}$$

donde esta última expresión puede ser resuelta usando Factorización LU, Gradiente Conjugado o alguna variante de GMRES, luego se hace

$$\underline{y2} = \underline{\underline{A}}^i_{\Gamma I}\underline{y1} \tag{A.86}$$

y para finalizar el cálculo, se obtiene

$$\underline{b}_i = \underline{b}_{\Gamma_i} - \underline{y2}. \tag{A.87}$$

Observación 11 *En la evaluación de*

$$\underline{u}_{I_i} = \left(\underline{\underline{A}}^i_{II}\right)^{-1}\left(\underline{b}_{I_i} - \underline{\underline{A}}^i_{I\Gamma}\underline{u}_{\Gamma_i}\right) \tag{A.88}$$

esta de nuevo involucrado $\left(\underline{\underline{A}}^i_{II}\right)^{-1}$*, por ello debo de usar el siguiente procedimiento para evaluar de forma eficiente esta expresión, realizando las operaciones equivalentes*

$$\begin{aligned} \underline{x4} &= \underline{b}_{I_i} - \underline{\underline{A}}^i_{I\Gamma}\underline{u}_{\Gamma_i} \\ \underline{u}_{I_i} &= \left(\underline{\underline{A}}^i_{II}\right)^{-1}\underline{x4} \end{aligned} \tag{A.89}$$

multiplicando por $\underline{\underline{A}}^i_{II}$ *a la última expresión, se obtiene*

$$\underline{\underline{A}}^i_{II}\underline{u}_{I_i} = \underline{\underline{A}}^i_{II}\left(\underline{\underline{A}}^i_{II}\right)^{-1}\underline{x4} \tag{A.90}$$

simplificando, se obtiene

$$\underline{\underline{A}}^i_{II}\underline{u}_{I_i} = \underline{x4} \tag{A.91}$$

esta última expresión puede ser resuelta usando Factorización LU, Cholesky, Gradiente Conjugado o alguna variante de GMRES.

Como se indico en las últimas observaciones, para resolver el sistema $\underline{\underline{A}}^i_{II}\underline{x} = \underline{b}$ se puede usar Factorización LU, Factorización Cholesky, Gradiente Conjugado, alguna variante de GMRES o cualquier otro método para resolver sistemas lineales, pero deberá de usarse aquel que proporcione la mayor velocidad en el cálculo o que consuma la menor cantidad de memoria —ambas condicionantes son mutuamente excluyentes—, por ello la decisión de que método usar deberá de tomarse al momento de tener que resolver un problema particular en un equipo dado y básicamente el condicionante es el tamaño de la matriz $\underline{\underline{A}}^i_{II}$.

Para usar el método de Factorización LU, se deberá primeramente de factorizar la matriz bandada $\underline{\underline{A}}^i_{II}$ en una matriz $\underline{\underline{LU}}$, la cual es bandada pero incrementa el tamaño de la banda a más del doble, pero esta operación sólo se deberá de realizar una vez en cada subdominio, y para solucionar los diversos sistemas lineales $\underline{\underline{A}}^i_{II}\underline{x} = \underline{b}$ sólo será necesario evaluar los sistemas

$$\begin{aligned} \underline{\underline{L}}\underline{y} &= \underline{b} \\ \underline{\underline{U}}\underline{x} &= \underline{y} \end{aligned} \tag{A.92}$$

en donde $\underline{y}$ es un vector auxiliar. Esto proporciona una manera eficiente de evaluar el sistema lineal pero el consumo en memoria para un problema particular puede ser excesivo.

Por ello, si el problema involucra una gran cantidad de nodos interiores y el equipo en el que se implantará la ejecución del programa tiene una cantidad de memoria limitada, es recomendable usar el método de Gradiente Conjugado o alguna variante de GMRES, este consume una cantidad de memoria adicional pequeña, pero puede consumir muchas iteraciones con el consecuente aumento de tiempo de cómputo para alcanzar la precisión de la Factorización LU.

De esta forma, es posible adaptar el código para tomar en cuenta desde la implementación al equipo de cómputo al que se tenga acceso y poder sacar el máximo provecho al método de Subestructuración en la resolución de problemas elípticos de gran envergadura.

El número de condicionamiento del complemento de Schur sin precondicionamiento puede ser estimado, para ello:

Definición 12 *Introduzco una norma-L^2 equivalente sobre Γ mediante*

$$\left\|\underline{u_\Gamma}\right\|_\Gamma^2 = \sum_{i=1}^{E} \left\|\underline{u_\Gamma}\right\|_{L^2(\partial\Omega_i)}^2 . \tag{A.93}$$

Teorema 13 *Sea $\underline{u_\Gamma}$ la traza de funciones de elemento finito en V^h sobre Γ, asumo que los coeficientes de la ecuación diferencial parcial $\rho_i = 1$, $i = 1, 2, ..., E$, y que la malla fina $\mathcal{T}_h$ y la malla gruesa $\mathcal{T}_H$ sea cuasi-uniforme. Entonces existen dos constantes positivas c y C, independientes de h y H, tal que*

$$cH \left\|\underline{u_\Gamma}\right\|_\Gamma^2 \leq s(\underline{u_\Gamma}, \underline{u_\Gamma}) \leq Ch^{-1} \left\|\underline{u_\Gamma}\right\|_\Gamma^2 \tag{A.94}$$

de este modo

$$\kappa = cond(\underline{\underline{S}}) \leq \frac{C}{Hh}. \tag{A.95}$$

Por analogía al método de subestructuración desarrollado anteriormente, dado un sistema lineal $\underline{\underline{M}}\underline{x} = \underline{f}$ que proviene de la discretización de algún método tipo Diferencias Finitas, Elemento Finito o Volumen Finito, siempre es posible reacomodarlo como (véase [7])

$$\begin{pmatrix} \underline{\underline{A}} & \underline{\underline{B}} \\ \underline{\underline{C}} & \underline{\underline{D}} \end{pmatrix} \begin{pmatrix} \underline{u} \\ \underline{v} \end{pmatrix} = \begin{pmatrix} \underline{a} \\ \underline{b} \end{pmatrix} \tag{A.96}$$

con $\underline{\underline{M}} = \begin{pmatrix} \underline{\underline{A}} & \underline{\underline{B}} \\ \underline{\underline{C}} & \underline{\underline{D}} \end{pmatrix}$, $\underline{x} = \begin{pmatrix} \underline{u} \\ \underline{v} \end{pmatrix}$ y $\underline{f} = \begin{pmatrix} \underline{a} \\ \underline{b} \end{pmatrix}$, en la cual la matriz $\underline{\underline{A}}$ sea invertible, entonces

$$\left(\underline{\underline{D}} - \underline{\underline{C}}\underline{\underline{A}}^{-1}\underline{\underline{B}}\right)\underline{v} = \underline{b} - \underline{\underline{C}}\underline{\underline{A}}^{-1}\underline{a} \tag{A.97}$$

y donde

$$\underline{u} = \underline{\underline{A}}^{-1}\left(\underline{a} - \underline{\underline{B}}\underline{v}\right). \tag{A.98}$$

Así, he transformado el sistema $\underline{\underline{M}}\underline{x} = \underline{f}$, en otro equivalente

$$\underline{\underline{N}}\underline{v} = \underline{g} \tag{A.99}$$

pero con un menor número de grados de libertad, donde

$$\underline{\underline{N}} = \left(\underline{\underline{D}} - \underline{\underline{C}}\underline{\underline{A}}^{-1}\underline{\underline{B}}\right), \qquad \underline{g} = \underline{b} - \underline{\underline{C}}\underline{\underline{A}}^{-1}\underline{a} \tag{A.100}$$

a esta descomposición matricial se le conoce como la descomposición de Schur.

B Consideraciones Sobre la Implementación de Métodos de Solución de Grandes Sistemas de Ecuaciones Lineales

Los modelos matemáticos de muchos sistemas en Ciencia e Ingeniería y en particular una gran cantidad de sistemas continuos geofísicos requieren el procesamiento de sistemas algebraicos de gran escala. Es este trabajo se muestra como proceder para transformar un problema de ecuaciones diferenciales parciales en un sistema algebraico virtual de ecuaciones lineales; y así, poder hallar la solución a dicho problema al resolver el sistema lineal asociado al esquema DVS. La solución de este sistema virtual, involucra la solución acoplada de muchos sistemas lineales locales —uno por cada subdominio—, cada uno de estos sistemas lineales puede ser expresado en la forma matricial siguiente

$$\underline{\underline{A}}\,\underline{u} = \underline{f} \tag{B.1}$$

donde la matriz $\underline{\underline{A}}$ es de tamaño $n \times n$ y generalmente bandada, cuyo tamaño de banda es b.

Los métodos de resolución del sistema algebraico de ecuaciones $\underline{\underline{A}}\,\underline{u} = \underline{f}$ se clasifican en dos grandes grupos (véase [14]): los métodos directos y los métodos iterativos. En los métodos directos la solución $\underline{u}$ se obtiene en un número fijo de pasos y sólo están sujetos a los errores de redondeo. En los métodos iterativos, se realizan iteraciones para aproximarse a la solución $\underline{u}$ aprovechando las características propias de la matriz $\underline{\underline{A}}$, tratando de usar un menor número de pasos que en un método directo (véase [10], [11], [12] y [14]).

Por lo general, es conveniente usar librerías[34] para implementar de forma eficiente a los vectores, matrices —bandadas y dispersas— y resolver los sistemas lineales locales asociados al método DVS, pero se decidió[35] hacer una jerarquía de clases propia, que implementa los algoritmos necesarios para este trabajo, los cuales corren en cualquier equipo de cómputo que tenga un compilador de C++ y la librería de paso de mensajes MPI; y en caso de querer usar librerías para la manipulación de sistemas lineales, sólo es necesario especializar las clases desarrolladas con las implementaciones particulares de estas; y así, ocultar el uso de dichas librerías sin afectar al resto del código. Siendo sencillo, adaptar el código para usar una o más librerías o versiones de estas, según sea necesario para correr el programa en un equipo de cómputo particular.

[34] Algunas de las librerías más usadas para resolver sistemas lineales usando matrices bandadas y dispersar son PETCs, HYPRE, ATLAS, LAPACK++, LAPACK, EISPACK, LINPACK, BLAS, entre muchas otras alternativas, tanto para implementaciones secuenciales como paralelas y más recientemente para hacer uso de los procesadores CUDA en las GPU de nVidia.

[35] La variedad de librerías y las diferentes versiones que existen —muchas de ellas, difieren en la forma de programar entre versiones sucesivas— y pueden usarse es grande, pero cada una de ellas requiere de una implementación especifica en el programa y una configuración particular del equipo. Esto restringe al código desarrollado a una plataforma y versión particular de la librería seleccionada.

Así, para poder operar con los diversos métodos numéricos directos o iterativos que resuelven sistemas lineales reales y virtuales, se implemento una jerarquía de clases, la cual se muestra a continuación:

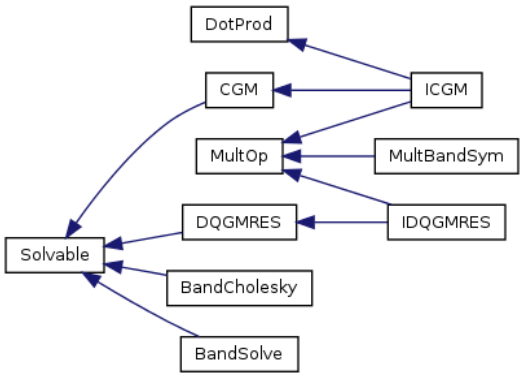

Figura 20: Jerarquía de clases para la resolución de sistemas lineales

Esta jerarquía permite que los métodos DVS le soliciten a la clase abstracta Solvable[36] que resuelva un determinado sistema lineal sin importar el método numérico subyacente —directos o iterativos—, si la matriz es real —existe como tal— o es virtual —esta dispersa en los distintos procesadores del Cluster— y es reutilizable tanto en la implementación secuencial como paralela. La clase Solvable tiene la siguiente estructura:

```
class Solvable
{
    private:
        int iter;
    public:
        virtual int getIter(void)=0;
        virtual void solve(double *x, double *y)=0;
};
```

B.1 Métodos Directos

En los métodos directos (véase [10] y [13]), la solución $\underline{u}$ se obtiene en un número fijo de pasos y sólo están sujetos a errores de redondeo. Entre los métodos más importantes se puede considerar: Factorización LU —para matrices simétricas y no simétricas— y Factorización Cholesky —para matrices simétricas—. En

[36] En general los comportamientos virtuales de las clases abstractas, pueden no ser eficientes si son llamadas una gran cantidad de veces durante la ejecución del programa. Para el caso del esquema DVS, en el cual se usa CGM o GMRES para resolver el sistema lineal virtual, este sólo se llama una sola vez; y el proceso de solución del sistema lineal asociado consume la mayoría del tiempo de ejecución, por eso se considera eficiente.

todos los casos la matriz original $\underline{\underline{A}}$ es modificada y en caso de usar la Factorización LU el tamaño de la banda b crece a $2b+1$ si la factorización se realiza en la misma matriz.

B.1.1 Factorización LU

Sea $\underline{\underline{U}}$ una matriz triangular superior obtenida de $\underline{\underline{A}}$ por eliminación bandada. Entonces $\underline{\underline{U}} = \underline{\underline{L}}^{-1}\underline{\underline{A}}$, donde $\underline{\underline{L}}$ es una matriz triangular inferior con unos en la diagonal. Las entradas de $\underline{\underline{L}}^{-1}$ pueden obtenerse de los coeficientes $\underline{\underline{L}}_{ij}$ y pueden ser almacenados estrictamente en las entradas de la diagonal inferior de $\underline{\underline{A}}$ ya que estas ya fueron eliminadas. Esto proporciona una Factorización $\underline{\underline{LU}}$ de $\underline{\underline{A}}$ en la misma matriz $\underline{\underline{A}}$ ahorrando espacio de memoria, donde el ancho de banda cambia de b a $2b+1$.

En el algoritmo de Factorización LU, se toma como datos de entrada del sistema $\underline{\underline{A}}\underline{u} = \underline{f}$, a la matriz $\underline{\underline{A}}$, la cual será factorizada en la misma matriz, esta contendrá a las matrices $\underline{\underline{L}}$ y $\underline{\underline{U}}$ producto de la factorización, quedando el método numérico esquemáticamente como:

$$
\begin{array}{l}
A_{ii} = 1, \text{ para } i = 1, ..., N \\
\text{Para } J = 1, 2, ..., N \ \{ \\
\qquad \text{Para } i = 1, 2, ..., j \\
\qquad\qquad A_{ij} = A_{ij} - \displaystyle\sum_{k=1}^{i-1} A_{ik}A_{kj} \\
\qquad \text{Para } i = j+1, j+2, ..., N \\
\qquad\qquad A_{ij} = \frac{1}{A_{jj}} \left(A_{ij} - \displaystyle\sum_{k=1}^{j-1} A_{ik}A_{kj} \right) \\
\}
\end{array}
\tag{B.2}
$$

El problema original $\underline{\underline{A}}\underline{u} = \underline{f}$ se escribe como $\underline{\underline{LU}}\underline{u} = \underline{f}$, donde la búsqueda de la solución $\underline{u}$ se reduce a la solución sucesiva de los sistemas lineales triangulares

$$
\underline{\underline{L}}\underline{y} = \underline{f} \quad \text{y} \quad \underline{\underline{U}}\underline{u} = \underline{y}. \tag{B.3}
$$

i,.e.

$$
\underline{\underline{L}}\underline{y} = \underline{f} \Leftrightarrow
\begin{cases}
y_1 = f_1/l_{11} \\
y_i = \frac{1}{A_{ii}} \left(f_i - \displaystyle\sum_{j=1}^{i-1} A_{ii}y_i \right) \text{ para toda } i = 1, ..., n
\end{cases}
\tag{B.4}
$$

y

$$
\underline{\underline{U}}\underline{u} = \underline{y} \Leftrightarrow
\begin{cases}
x_n = y_n/u_{nn} \\
x_i = \frac{1}{A_{ii}} \left(y_i - \displaystyle\sum_{j=i+1}^{n} A_{ij}x_j \right) \text{ para toda } i = n-1, ..., 1
\end{cases}
\tag{B.5}
$$

La descomposición $\underline{\underline{LU}}$ requiere $N^3/3$ operaciones aritméticas para la matriz llena, pero sólo Nb^2 operaciones aritméticas para la matriz con un ancho de banda de b siendo esto más económico computacionalmente.

Implementación LU Orientada a Objetos en C++ En este caso se desarrollo una jerarquía de clases para poder operar con matrices bandadas y dispersas, llamada BandSolve la cual se muestra a continuación:

```
class BandSolve: public Solvable
{
    private:
        double **AK;
    protect:
        factorLU(void);
    public:
        BanSolve(int n, double **A);
        BanSolve(int n, Matriz *A);
        void solve(double *x, double *y);
        void convertBand(void);
        int getIter(void);
};
```

B.1.2 Factorización Cholesky

Cuando la matriz es simétrica y definida positiva, se obtiene la descomposición $\underline{\underline{LU}}$ de la matriz $\underline{\underline{A}} = \underline{\underline{LDU}} = \underline{\underline{LDL}}^T$ donde $\underline{\underline{D}} = diag(\underline{\underline{U}})$ es la diagonal con entradas positivas.

En el algoritmo de Factorización Cholesky, se toma como datos de entrada del sistema $\underline{\underline{A}}\underline{u} = \underline{f}$, a la matriz $\underline{\underline{A}}$, la cual será factorizada en la misma matriz y contendrá a las matrices $\underline{\underline{L}}$ y $\underline{\underline{L}}^T$ producto de la factorización, quedando el método numérico esquemáticamente como:

$$\begin{aligned} &\text{para } i = 1, 2, ..., n \text{ y } j = i+1, ..., n \\ &A_{ii} = \sqrt{\left(A_{ii} - \sum_{k=1}^{i-1} A_{ik}^2 \right)} \\ &A_{ji} = \left(A_{ji} - \sum_{k=1}^{i-1} A_{jk} A_{ik} \right) / A_{ii} \end{aligned} \tag{B.6}$$

El problema original $\underline{\underline{A}}\underline{u} = \underline{f}$ se escribe como $\underline{\underline{LL}}^T\underline{u} = \underline{b}$, donde la búsqueda de la solución $\underline{u}$ se reduce a la solución sucesiva de los sistemas lineales triangulares

$$\underline{\underline{L}}\underline{y} = \underline{f} \quad \text{y} \quad \underline{\underline{L}}^T\underline{u} = \underline{y} \tag{B.7}$$

usando la formulación equivalente dada por las Ec.(B.4) y (B.5) para la descomposición LU.

La mayor ventaja de esta descomposición es que, en el caso en que es aplicable, el costo de cómputo es sustancialmente reducido, ya que requiere de $N^3/6$ multiplicaciones y $N^3/6$ sumas.

Implementación Cholesky Orientada a Objetos en C++ En este caso se desarrollo una jerarquía de clases para poder operar con matrices bandadas y dispersas, llamada BandCholesky la cual se muestra a continuación:

```
class BandCholesky: public Solvable
{
    private:
        double **AK;
    protect:
        factorLU(void);
    public:
        BanCholesky(int n, double **A);
        BanCholesky(int n, Matriz *A);
        void solve(double *x, double *y);
        void convertBand(void);
        int getIter(void);
};
```

B.2 Métodos Iterativos

En los métodos iterativos, se realizan iteraciones para aproximarse a la solución $\underline{u}$ aprovechando las características propias de la matriz $\underline{\underline{A}}$, tratando de usar un menor número de pasos que en un método directo (véase [10] y [13]).

En los métodos iterativos tales como Jacobi, Gauss-Seidel y de Relajación Sucesiva (SOR) en el cual se resuelve el sistema lineal

$$\underline{\underline{A}}\underline{u} = \underline{f} \tag{B.8}$$

comienza con una aproximación inicial $\underline{u}^0$ a la solución $\underline{u}$ y genera una sucesión de vectores $\left\{u^k\right\}_{k=1}^{\infty}$ que converge a $\underline{u}$. Los métodos iterativos traen consigo un proceso que convierte el sistema $\underline{\underline{A}}\underline{u} = \underline{f}$ en otro equivalente mediante la iteración de punto fijo de la forma $\underline{u} = \underline{\underline{T}}\underline{u} + \underline{c}$ para alguna matriz fija $\underline{\underline{T}}$ y un vector $\underline{c}$. Luego de seleccionar el vector inicial $\underline{u}^0$ la sucesión de los vectores de la solución aproximada se genera calculando

$$\underline{u}^k = \underline{\underline{T}}\underline{u}^{k-1} + \underline{c} \quad \forall k = 1, 2, 3, ... \tag{B.9}$$

La convergencia a la solución la garantiza el siguiente teorema (véase [14]).

Teorema 14 *Si* $\left\|\underline{\underline{T}}\right\| < 1$*, entonces el sistema lineal* $\underline{u} = \underline{\underline{T}}\underline{u} + \underline{c}$ *tiene una solución única* $\underline{u}^*$ *y las iteraciones* $\underline{u}^k$ *definidas por la fórmula* $\underline{u}^k = \underline{\underline{T}}\underline{u}^{k-1} + \underline{c} \quad \forall k = 1, 2, 3, ...$ *convergen hacia la solución exacta* $\underline{u}^*$ *para cualquier aproximación inicial* $\underline{u}^0$.

Nótese que, mientras menor sea la norma de la matriz $\underline{\underline{T}}$, más rápida es la convergencia, en el caso cuando $\left\|\underline{\underline{T}}\right\|$ es menor que uno, pero cercano a uno, la convergencia es lenta y el número de iteraciones necesario para disminuir el error depende significativamente del error inicial. En este caso, es deseable proponer al vector inicial $\underline{u}^0$ de forma tal que sea mínimo el error inicial. Sin embargo, la elección de dicho vector no tiene importancia si la $\left\|\underline{\underline{T}}\right\|$ es pequeña, ya que la convergencia es rápida.

Como es conocido, la velocidad de convergencia de los métodos iterativos dependen de las propiedades espectrales de la matriz de coeficientes del sistema de ecuaciones, cuando el operador diferencial $\mathcal{L}$ de la ecuación del problema a resolver es auto-adjunto se obtiene una matriz simétrica y positivo definida y el número de condicionamiento de la matriz $\underline{\underline{A}}$, es por definición

$$cond(\underline{\underline{A}}) = \frac{\lambda_{\max}}{\lambda_{\min}} \geq 1 \tag{B.10}$$

donde $\lambda_{\max}$ y $\lambda_{\min}$ es el máximo y mínimo de los eigen-valores de la matriz $\underline{\underline{A}}$. Si el número de condicionamiento es cercano a 1 los métodos numéricos al solucionar el problema convergerá en pocas iteraciones, en caso contrario se requerirán muchas iteraciones.

Frecuentemente al usar el método de Elemento Finito, Diferencias Finitas, entre otros, se tiene una velocidad de convergencia de $O\left(\frac{1}{h^2}\right)$ y en el caso de métodos de descomposición de dominio sin precondicionar se tiene una velocidad de convergencia de $O\left(\frac{1}{h}\right)$, donde h es la máxima distancia de separación entre nodos continuos de la partición, es decir, que poseen una pobre velocidad de convergencia cuando $h \to 0$ (véase [8], [9], [10] y [14]).

Los métodos Jacobi, Gauss-Seidel y de Relajación Sucesiva (SOR) son usualmente menos eficientes que los métodos discutidos en el resto de esta sección basados en el espacio de Krylov (véase [51] y [13]). Estos métodos minimizan, en la k-ésima iteración alguna medida de error sobre el espacio afín $\underline{x_0} + \mathcal{K}_k$, donde $\underline{x_0}$ es la iteración inicial y $\mathcal{K}_k$ es el k-ésimo subespacio de Krylov

$$\mathcal{K}_k = Generado\left\{\underline{r_0}, \underline{\underline{A}}\underline{r_0}, ..., \underline{\underline{A}}^{k-1}\underline{r_0}\right\} \text{ para } k \geq 1. \tag{B.11}$$

El residual es $\underline{r} = \underline{b} - \underline{\underline{A}}\underline{x}$, tal $\left\{\underline{r_k}\right\}_{k\geq 0}$ denota la sucesión de residuales

$$\underline{r_k} = \underline{b} - \underline{\underline{A}}\underline{x_k}. \tag{B.12}$$

Entre los métodos más usados definidos en el espacio de Krylov para el tipo de problemas tratados en el presente trabajo se puede considerar: Método de Gradiente Conjugado —para matrices simétricas— y GMRES —para matrices no simétricas—.

B.2.1 Método de Gradiente Conjugado

Si la matriz generada por la discretización es simétrica —$\underline{\underline{A}} = \underline{\underline{A}}^T$— y definida positiva —$\underline{u}^T\underline{\underline{A}}\underline{u} > 0$ para todo $\underline{u} \neq 0$—, entonces es aplicable el método de Gradiente Conjugado —Conjugate Gradient Method (CGM)—. La idea básica en

que descansa el método del Gradiente Conjugado consiste en construir una base de vectores ortogonales espacio de Krylov $\mathcal{K}_n\left(\underline{\underline{A}}, \underline{v}^n\right)$ y utilizarla para realizar la búsqueda de la solución en forma lo más eficiente posible.

Tal forma de proceder generalmente no sería aconsejable porqué la construcción de una base ortogonal utilizando el procedimiento de Gramm-Schmidt requiere, al seleccionar cada nuevo elemento de la base, asegurar su ortogonalidad con respecto a cada uno de los vectores construidos previamente. La gran ventaja del método de Gradiente Conjugado radica en que cuando se utiliza este procedimiento, basta con asegurar la ortogonalidad de un nuevo miembro con respecto al último que se ha construido, para que automáticamente esta condición se cumpla con respecto a todos los anteriores.

En el algoritmo de Gradiente Conjugado, se toma a la matriz $\underline{\underline{A}}$ como simétrica y positiva definida, y como datos de entrada del sistema $\underline{\underline{A}}\underline{u} = \underline{f}$, el vector de búsqueda inicial $\underline{u}^0$ y se calcula $\underline{r}^0 = \underline{f} - \underline{\underline{A}}\underline{u}^0$, $\underline{p}^0 = \underline{r}^0$, quedando el método numérico esquemáticamente como:

$$
\begin{array}{l}
\alpha^n = \frac{\left\langle \underline{p}^n, \underline{p}^n \right\rangle}{\left\langle \underline{p}^n, \underline{\underline{A}}\underline{p}^n \right\rangle} \\
\underline{u}^{n+1} = \underline{u}^n + \alpha^n \underline{p}^n \\
\underline{r}^{n+1} = \underline{r}^n - \alpha^n \underline{\underline{A}}\underline{p}^n \\
\text{Prueba de convergencia} \\
\beta^n = \frac{\left\langle \underline{r}^{n+1}, \underline{r}^{n+1} \right\rangle}{\left\langle \underline{r}^n, \underline{r}^n \right\rangle} \\
\underline{p}^{n+1} = \underline{r}^{n+1} + \beta^n \underline{p}^n \\
n = n + 1
\end{array}
\tag{B.13}
$$

donde $\langle \cdot, \cdot \rangle = (\cdot, \cdot)$ será el producto interior adecuado al sistema lineal en particular, la solución aproximada será $\underline{u}^{n+1}$ y el vector residual será $\underline{r}^{n+1}$.

En la implementación numérica y computacional del método es necesario realizar la menor cantidad de operaciones posibles por iteración, en particular en $\underline{\underline{A}}\underline{p}^n$, una manera de hacerlo queda esquemáticamente como:

Dado el vector de búsqueda inicial $\underline{u}$, calcula $\underline{r} = \underline{f} - \underline{\underline{A}}\underline{u}$, $\underline{p} = \underline{r}$ y $\mu = \underline{r} \cdot \underline{r}$.

Para $n = 1, 2, ...,$Mientras $(\mu < \varepsilon)$ {
$v = \underline{\underline{A}}\underline{p}$
$\alpha = \frac{\mu}{\underline{p} \cdot \underline{v}}$
$\underline{u} = \underline{u} + \alpha \underline{p}$
$\underline{r} = \underline{r} - \alpha \underline{v}$
$\mu' = \underline{r} \cdot \underline{r}$
$\beta = \frac{\mu'}{\mu}$
$\underline{p} = \underline{r} + \beta \underline{p}$
$\mu = \mu'$
}

La solución aproximada será $\underline{u}$ y el vector residual será $\underline{r}$.

Si se denota con $\{\lambda_i, V_i\}_{i=1}^N$ las eigen-soluciones de $\underline{\underline{A}}$, i.e. $\underline{\underline{A}}V_i = \lambda_i V_i$, $i = 0, 1, 2, ..., N$. Ya que la matriz $\underline{\underline{A}}$ es simétrica, los eigen-valores son reales y

se pueden ordenar $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_N$. Se define el número de condición por $Cond(\underline{\underline{A}}) = \lambda_N/\lambda_1$ y la norma de la energía asociada a $\underline{\underline{A}}$ por $\|\underline{u}\|^2_{\underline{\underline{A}}} = \underline{u} \cdot \underline{\underline{A}}\underline{u}$ entonces

$$\left\|\underline{u} - \underline{u}^k\right\|_{\underline{\underline{A}}} \leq \left\|\underline{u} - \underline{u}^0\right\|_{\underline{\underline{A}}} \left[\frac{1 - \sqrt{Cond(\underline{\underline{A}})}}{1 + \sqrt{Cond(\underline{\underline{A}})}}\right]^{2k}. \tag{B.14}$$

El siguiente teorema da idea del espectro de convergencia del sistema $\underline{\underline{A}}\underline{u} = \underline{b}$ para el método de Gradiente Conjugado.

Teorema 15 *Sea $\kappa = cond(\underline{\underline{A}}) = \frac{\lambda_{\max}}{\lambda_{\min}} \geq 1$, entonces el método de Gradiente Conjugado satisface la $\underline{\underline{A}}-$norma del error dado por*

$$\frac{\|e^n\|}{\|e^0\|} \leq \frac{2}{\left[\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^n + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{-n}\right]} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n \tag{B.15}$$

donde $\underline{e}^m = \underline{u} - \underline{u}^m$ del sistema $\underline{\underline{A}}\underline{u} = \underline{b}$.

Nótese que para κ grande se tiene que

$$\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \simeq 1 - \frac{2}{\sqrt{\kappa}} \tag{B.16}$$

tal que

$$\left\|\underline{e}^n\right\|_{\underline{\underline{A}}} \simeq \left\|\underline{e}^0\right\|_{\underline{\underline{A}}} \exp\left(-2\frac{n}{\sqrt{\kappa}}\right) \tag{B.17}$$

de lo anterior se puede esperar un espectro de convergencia del orden de $O(\sqrt{\kappa})$ iteraciones (véase [14] y [51]).

Definición 16 *Un método iterativo para la solución de un sistema lineal es llamado óptimo, si la razón de convergencia a la solución exacta es independiente del tamaño del sistema lineal.*

Definición 17 *Un método para la solución del sistema lineal generado por métodos de descomposición de dominio es llamado escalable, si la razón de convergencia no se deteriora cuando el número de subdominios crece.*

La gran ventaja de los métodos de descomposición de dominio precondicionados entre los que destacan a FETI-DP y BDDC y por ende DVS es que estos métodos pueden ser óptimos y escalables según el tipo de precondicionador a priori que se use en ellos.

Consideraciones sobre la Matriz y el producto punto en el CGM Es común al trabajar en métodos de descomposición de dominio que se requieran usar más de un producto interior $\langle \cdot, \cdot \rangle$ en distintas implementaciones del mismo algoritmo de Gradiente Conjugado. Por ello se diseña el algoritmo para soportar en forma de parámetro el objeto que implemente el producto punto.

```
class DotProd
{
    public:
        virtual double dot(double *x, double *y)=0;
};
```

En este caso, al usar el esquema DVS la matriz con la que trabajara el método de Gradiente Conjugado puede estar definida como tal o ser virtual; en el caso más general, puede estar en un procesador o en múltiples procesadores, por ello, en el diseño del algoritmo se decidió implementar un objeto el cual pueda realizar la multiplicación de la matriz por el vector, sin importar el tipo de matriz —real o virtual—.

```
class MultOp
{
    public:
        virtual void multOp(double *x, double *y)=0;
        virtual int getSize(void)=0;
};
```

Así, se diseña un solo método de Gradiente Conjugado robusto y flexible que soporta diferentes productos interiores y trabaja con matrices reales o virtuales, adaptándose sin ningún cambio a diversas necesidades de implementación tanto secuencial como paralela.

Implementación CGM Orientada a Objetos en C++ Dado el sistema lineal $\underline{\underline{A}}\underline{u} = \underline{f}$, donde $\underline{\underline{A}}$ sea una matriz real o virtual. La entrada al método será una elección de $\underline{u}^0$ como condición inicial, $\varepsilon > 0$ como la tolerancia del método, N como el número máximo de iteraciones además de un objeto para realizar la multiplicación de la matriz por un vector y otro objeto para el producto interior, el algoritmo del método de Gradiente Conjugado orientado a objetos en C++ queda como:

```
class CGM: public Solvable
{
    protect:
        MultOp *A;
        DotProd *dotP;
        double norm(double *x);
    public:
```

```
        CGM(MulOp &A, DotProd &dotP, double eps);
        void solve(double *x, double *y);
        int getIter(void);
};
```

B.2.2 Método Residual Mínimo Generalizado

Si la matriz generada por la discretización es no simétrica, entonces una opción, es el método Residual Mínimo Generalizado —Generalized Minimum Residual Method (GMRES)—, este representa una formulación iterativa común satisfaciendo una condición de optimización. La idea básica detrás del método se basa en construir una base ortonormal

$$\left\{\underline{v}^1, \underline{v}^2, ..., \underline{v}^n\right\} \tag{B.18}$$

para el espacio de Krylov $\mathcal{K}_n\left(\underline{\underline{A}}, \underline{v}^n\right)$. Para hacer $\underline{v}^{n+1}$ ortogonal a $\mathcal{K}_n\left(\underline{\underline{A}}, \underline{v}^n\right)$, es necesario usar todos los vectores previamente construidos $\left\{\underline{v}^{n+1j}\right\}_{j=1}^{n}$ —en la práctica sólo se guardan algunos vectores anteriores— en los cálculos. Y el algoritmo se basa en una modificación del método de Gram-Schmidt para la generación de una base ortonormal. Sea $\underline{\underline{V}}_n = \left[\underline{v}^1, \underline{v}^2, ..., \underline{v}^n\right]$ la cual denota la matriz conteniendo $\underline{v}^j$ en la j-ésima columna, para $j = 1, 2, ..., n$, y sea $\underline{\underline{H}}_n = [h_{i,j}], 1 \leq i, j \leq n$, donde las entradas de $\underline{\underline{H}}_n$ no especificadas en el algoritmo son cero. Entonces, $\underline{\underline{H}}_n$ es una matriz superior de Hessenberg. i.e. $h_{ij} = 0$ para $j < i - 1$, y

$$\begin{aligned} \underline{\underline{AV}}_n &= \underline{\underline{V}}_n \underline{\underline{H}}_n + h_{n+1,n}\left[0, ..., 0, \underline{v}^{n+1}\right] \\ \underline{\underline{H}}_n &= \underline{\underline{H}}_n^T \underline{\underline{AV}}_n. \end{aligned} \tag{B.19}$$

En el algoritmo del método Residual Mínimo Generalizado, la matriz $\underline{\underline{A}}$ es tomada como no simétrica, y como datos de entrada del sistema

$$\underline{\underline{A}}\underline{u} = \underline{f} \tag{B.20}$$

el vector de búsqueda inicial $\underline{u}^0$ y se calcula $\underline{r}^0 = \underline{f} - \underline{\underline{A}}\underline{u}^0$, $\beta^0 = \left\|r^0\right\|$, $\underline{v}^1 = \underline{r}^0/\beta^0$, quedando el método esquemáticamente como:

$$\begin{array}{l} \text{Para } n = 1, 2, ..., \text{Mientras } \beta^n < \tau\beta^0 \ \{ \\ \quad \underline{w}_0^{n+1} = \underline{\underline{A}}\underline{v}^n \\ \quad \text{Para } l = 1 \text{ hasta } n \ \{ \\ \qquad h_{l,n} = \left\langle \underline{w}_l^{n+1}, \underline{v}^l \right\rangle \\ \qquad \underline{w}_{l+1}^{n+1} = \underline{w}_l^{n+1} - h_{l,n}\underline{v}^l \\ \quad \} \\ \quad h_{n+1,n} = \left\|\underline{w}_{n+1}^{n+1}\right\| \\ \quad \underline{v}^{n+1} = \underline{w}_{n+1}^{n+1}/h_{n+1,n} \\ \quad \text{Calcular } \underline{y}^n \text{ tal que } \beta^n = \left\|\beta^0\underline{e}_1 - \underline{\underline{\hat{H}}}_n\underline{y}^n\right\| \text{ es mínima} \\ \} \end{array} \tag{B.21}$$

donde $\underline{\underline{\hat{H}}}_n = [h_{ij}]_{1\leq i\leq n+1, 1\leq j\leq n}$, la solución aproximada será $\underline{u}^n = \underline{u}^0 + \underline{\underline{V}}_n \underline{y}^n$, y el vector residual será

$$\underline{r}^n = \underline{r}^0 - \underline{\underline{AV}}_n y^n = \underline{\underline{V}}_{n+1} \left(\beta^0 \underline{e}_1 - \underline{\underline{\hat{H}}}_n \underline{y}^n \right). \tag{B.22}$$

Teorema 18 *Sea $\underline{u}^k$ la iteración generada después de k iteraciones de GMRES, con residual $\underline{r}^k$. Si la matriz $\underline{\underline{A}}$ es diagonalizable, i.e. $\underline{\underline{A}} = \underline{\underline{V}} \Lambda \underline{\underline{V}}^{-1}$ donde Λ es una matriz diagonal de eigen-valores de $\underline{\underline{A}}$, y $\underline{\underline{V}}$ es la matriz cuyas columnas son los eigen-vectores, entonces*

$$\frac{\left\| r^k \right\|}{\left\| r^0 \right\|} \leq \kappa \left(V \right) \min_{p_\kappa \in \Pi\kappa, p_k(0)=1} \max_{\lambda_j} \left| p_\kappa \left(\lambda_j \right) \right| \tag{B.23}$$

donde $\kappa(V) = \frac{\left\| \underline{\underline{V}} \right\|}{\left\| \underline{\underline{V}}^{-1} \right\|}$ es el número de condicionamiento de $\underline{\underline{V}}$.

Consideraciones sobre la Matriz en el GMRES En este caso al usar DVS la matriz con la que trabajara el método GMRES puede estar definida como tal o ser virtual, o en el caso más general, puede estar en un procesador o en múltiples procesadores, por ello en el diseño del algoritmo se decidió implementar un objeto el cual pueda realizar la multiplicación de la matriz por el vector, sin importar el tipo de matriz —real o virtual—.

```
class MultOp
{
    public:
        virtual void multOp(double *x, double *y)=0;
        virtual int getSize(void)=0;
};
```

Así, se diseña un sólo método de GMRES robusto y flexible que trabaje con matrices reales o virtuales, adaptándose sin ningún cambio a diversas necesidades de implementación tanto secuencial como paralela.

Implementación GMRES Orientada a Objetos en C++ Dado el sistema lineal $\underline{\underline{Au}} = \underline{f}$, donde $\underline{\underline{A}}$ sea una matriz real o virtual. La entrada al método será una elección de $\underline{u}^0$ como condición inicial, $\varepsilon > 0$ como la tolerancia del método, N como el número máximo de iteraciones y k el número de vectores a guardar además de un objeto para realizar la multiplicación de la matriz por un vector, el algoritmo del método de GMRES orientado a objetos en C++ queda como:

```
class DQGMRES: public Solvable
{
    protect:
        MultOp *A;
```

```
    public:
        DQGMRES(MulOp &A, int k);
        void solve(double *x, double *y);
        int getIter(void);
};
```

B.3 Estructura Óptima de las Matrices en su Implementación Computacional

Una parte fundamental de la implementación computacional de los métodos numéricos de resolución de sistemas algebraicos, es utilizar una forma óptima de almacenar, recuperar y operar las matrices, tal que, facilite los cálculos que involucra la resolución de grandes sistemas de ecuaciones lineales cuya implementación puede ser secuencial o paralela (véase [13]).

El sistema lineal puede ser expresado en la forma matricial $\underline{\underline{A}}\underline{u} = \underline{f}$, donde la matriz $\underline{\underline{A}}$ —que puede ser real o virtual— es de tamaño $n \times n$ con banda b, pero el número total de datos almacenados en ella es a los más $n*b$ números de doble precisión, en el caso de ser simétrica la matriz, el número de datos almacenados es menor a $(n*b)/2$. Además si el problema que la originó es de coeficientes constantes el número de valores almacenados se reduce drásticamente a sólo el tamaño de la banda b.

En el caso de que el método para la resolución del sistema lineal a usar sea del tipo Factorización LU o Cholesky, la estructura de la matriz cambia, ampliándose el tamaño de la banda de b a $2*b+1$ en la factorización, en el caso de usar métodos iterativos tipo CGM o GMRES la matriz se mantiene intacta con una banda b.

Para la resolución del sistema lineal virtual asociada a los métodos de descomposición de dominio, la operación básica que se realiza de manera reiterada, es la multiplicación de una matriz por un vector $\underline{v} = \underline{\underline{C}}\underline{u}$, la cual es necesario realizar de la forma más eficiente posible.

Un factor determinante en la implementación computacional, para que esta resulte eficiente, es la forma de almacenar, recuperar y realizar las operaciones que involucren matrices y vectores, de tal forma que la multiplicación se realice en la menor cantidad de operaciones y que los valores necesarios para realizar dichas operaciones queden en la medida de lo posible contiguos para ser almacenados en el Cache[37] del procesador.

[37] Nótese que la velocidad de acceso a la memoria principal (RAM) es relativamente lenta con respecto al Cache, este generalmente está dividido en sub-Caches L1 —de menor tamaño y el más rápido—, L2 y hasta L3 —el más lento y de mayor tamaño— los cuales son de tamaño muy reducido con respecto a la RAM.

Por ello, cada vez que las unidades funcionales de la Unidad de Aritmética y Lógica requieren un conjunto de datos para implementar una determinada operación en los registros, solicitan los datos primeramente a los Caches, estos consumen diversa cantidad de ciclos de reloj para entregar el dato si lo tienen —pero siempre el tiempo es menor que solicitarle el dato a la memoria principal—; en caso de no tenerlo, se solicitan a la RAM para ser cargados a los caches y poder implementar la operación solicitada.

Dado que la multiplicación de una matriz $\underline{\underline{C}}$ por un vector $\underline{u}$, dejando el resultado en $\underline{v}$ se realiza mediante el algoritmo

```
for (i=0; i<ren; i++)
{
    s = 0.0;
    for (j=0; j < col; j++)
    {
        s += C[i][j]*u[j];
    }
    v[i] = s;
}
```

Para lograr una eficiente implementación del algoritmo anterior, es necesario que el gran volumen de datos desplazados de la memoria al Cache y viceversa sea mínimo. Por ello, los datos se deben agrupar para que la operación más usada —en este caso multiplicación matriz por vector— se realice con la menor solicitud de datos a la memoria principal, si los datos usados —renglón de la matriz— se ponen contiguos minimizará los accesos a la memoria principal, pues es más probable que estos estarán contiguos en el Cache al momento de realizar la multiplicación.

Por ejemplo, en el caso de matrices bandadas de tamaño de banda b, el algoritmo anterior se simplifica a

```
for (i=0; i<ren; i++)
{
    s= 0.0;
    for (k=0; k < ban; k++)
    {
        if ((Ind[k] + i) >= 0 && (Ind[k]+i) < ren)
            s += Dat[i][k]*u[Ind[k]+i];
    }
    v[i]=s;
}
```

Si, la solicitud de memoria para Dat[i] se hace de tal forma que los datos del renglón estén continuos —son b números de punto flotante—, esto minimizará los accesos a la memoria principal en cada una de las operaciones involucradas en el producto, como se explica en las siguientes secciones.

B.3.1 Matrices Bandadas

En el caso de las matrices bandadas de banda b —sin pérdida de generalidad y para propósitos de ejemplificación se supone pentadiagonal— típicamente tiene

la siguiente forma

$$\underline{\underline{A}} = \begin{bmatrix} a_1 & b_1 & & & c_1 & & & & \\ d_2 & a_2 & b_2 & & & c_2 & & & \\ & d_3 & a_3 & b_3 & & & c_3 & & \\ & & d_4 & a_4 & b_4 & & & c_4 & \\ e_5 & & & d_5 & a_5 & b_5 & & & c_5 \\ & e_6 & & & d_6 & a_6 & b_6 & & \\ & & e_7 & & & d_7 & a_7 & b_7 & \\ & & & e_8 & & & d_8 & a_8 & b_8 \\ & & & & e_9 & & & d_9 & a_9 \end{bmatrix} \tag{B.24}$$

la cual puede ser almacenada usando el algoritmo (véase [13]) Compressed Diagonal Storage (CDS), optimizado para ser usado en C++, para su almacenamiento y posterior recuperación. Para este ejemplo en particular, se hará uso de un vector de índices

$$\underline{Ind} = [-5, -1, 0, +1, +5] \tag{B.25}$$

y los datos serán almacenados usando la estructura

$$\underline{\underline{Dat}} = \begin{bmatrix} 0 & 0 & a_1 & b_1 & c_1 \\ 0 & d_2 & a_2 & b_2 & c_2 \\ 0 & d_3 & a_3 & b_3 & c_3 \\ 0 & d_4 & a_4 & b_4 & c_4 \\ e_5 & d_5 & a_5 & b_5 & c_5 \\ e_6 & d_6 & a_6 & b_6 & 0 \\ e_7 & d_7 & a_7 & b_7 & 0 \\ e_8 & d_8 & a_8 & b_8 & 0 \\ e_9 & d_9 & a_9 & 0 & 0 \end{bmatrix} \tag{B.26}$$

de tal forma que la matriz $\underline{\underline{A}}$ puede ser reconstruida de forma eficiente. Para obtener el valor $A_{i,j}$, calculo $ind = j - i$, si el valor ind esta en la lista de índices $\underline{Ind}$ —supóngase en la columna k—, entonces $A_{i,j} = Dat_{ik}$, en otro caso $A_{i,j} = 0$.

Casos Particulares de la Matriz Bandada $\underline{\underline{A}}$ Básicamente dos casos particulares surgen en el tratamiento de ecuaciones diferenciales parciales: El primer caso es cuando el operador diferencial parcial es simétrico y el otro, en el que los coeficientes del operador sean constantes.

Para el primer caso, al ser la matriz simétrica, sólo es necesario almacenar la parte con índices mayores o iguales a cero, de tal forma que se buscara el índice que satisfaga $ind = |j - i|$, reduciendo el tamaño de la banda a $b/2$ en la matriz $\underline{\underline{A}}$.

Para el segundo caso, al tener coeficientes constantes el operador diferencial, los valores de los renglones dentro de cada columna de la matriz son iguales, y sólo es necesario almacenarlos una sola vez, reduciendo drásticamente el tamaño de la matriz de datos.

Implementación de Matrices Bandadas Orientada a Objetos en C++
Una forma de implementar la matriz bandada $\underline{\underline{A}}$ de banda b en C++ es mediante:

```
// Creacion
double **Dat;
Dat = new double*[ren];
for (i = 0; i < ren; i++) Dat[i] = new double[b];
int *Ind;
Ind = new int[b];
// Inicializacion
for (i = 0; i < ren; i++)
      for (j = 0; j < b; j++) Dat[i][j] = 0.0;
for (i = 0; i < b; i++) Ind[i] = 0;
```

B.3.2 Matrices Dispersas

Las matrices dispersas de a lo más b valores distintos por renglón —sin pérdida de generalidad y para propósitos de ejemplificación se supone $b = 3$— que surgen en métodos de descomposición de dominio para almacenar algunas matrices, típicamente tienen la siguiente forma

$$\underline{\underline{A}} = \begin{bmatrix} a_1 & & & & b_1 & & & c_1 \\ & a_2 & & b_2 & & & c_2 & \\ & & & a_3 & & & b_3 & c_3 \\ a_4 & & & b_4 & & & & \\ & & a_5 & & b_5 & & & c_5 \\ a_6 & b_6 & c_6 & & & & & \\ & & & & & a_7 & b_7 & c_7 \\ & & & a_8 & & b_8 & c_8 & \\ & & & & & a_9 & b_9 & \end{bmatrix} \tag{B.27}$$

la cual puede ser almacenada usando el algoritmo (véase [13]) Jagged Diagonal Storage (JDC), optimizado para ser usado en C++. Para este ejemplo en particular, se hará uso de una matriz de índices

$$\underline{\underline{Ind}} = \begin{bmatrix} 1 & 6 & 9 \\ 2 & 5 & 8 \\ 5 & 8 & 9 \\ 1 & 4 & 0 \\ 3 & 6 & 9 \\ 1 & 2 & 3 \\ 7 & 8 & 9 \\ 4 & 7 & 8 \\ 7 & 8 & 0 \end{bmatrix} \tag{B.28}$$

y los datos serán almacenados usando la estructura

$$\underline{\underline{Dat}} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \\ a_4 & b_4 & 0 \\ a_5 & b_5 & c_5 \\ a_6 & b_6 & c_6 \\ a_7 & b_7 & c_7 \\ a_8 & b_8 & c_8 \\ a_9 & b_9 & 0 \end{bmatrix} \tag{B.29}$$

de tal forma que la matriz $\underline{\underline{A}}$ puede ser reconstruida de forma eficiente. Para obtener el obtener el valor $A_{i,j}$, busco el valor j en la lista de índices $\underline{Ind}$ dentro del renglón i, si lo encuentro en la posición k, entonces $A_{i,j} = Dat_{ik}$, en otro caso $A_{i,j} = 0$.

Casos Particulares de la Matriz Dispersa $\underline{\underline{A}}$ Si la matriz $\underline{\underline{A}}$, que al ser almacenada, se observa que existen a lo más r diferentes renglones con valores distintos de los n con que cuenta la matriz y si $r << n$, entonces es posible sólo guardar los r renglones distintos y llevar un arreglo que contenga la referencia al renglón almacenado.

Implementación de Matrices Dispersas Orientada a Objetos en C++
Una forma de implementar la matriz bandada $\underline{\underline{A}}$ de banda b en C++ es mediante:

```
// Creación
double **Dat;
Dat = new double*[ren];
for (i = 0; i < ren; i++) Dat[i] = new double[b];
int **Ind;
Ind = new int*[ren];
for (i = 0; i < ren; i++) Ind[i] = new int[b];
// Inicialización
for (i = 0; i < ren; i++)
     for (j = 0; j < b; j++) Dat[i][j] = 0.0, Ind[i][j] = -1;
```

B.3.3 Multiplicación Matriz-Vector

Los métodos de descomposición de dominio requieren por un lado la resolución de al menos un sistema lineal y por el otro lado requieren realizar la operación de multiplicación de matriz por vector, i.e. $\underline{\underline{C}}\underline{u}$ de la forma más eficiente posible, por ello los datos se almacenan de tal forma que la multiplicación se realice en la menor cantidad de operaciones.

Dado que la multiplicación de una matriz $\underline{\underline{C}}$ por un vector $\underline{u}$, dejando el resultado en $\underline{v}$ se realiza mediante el algoritmo:

```
for (i=0; i<ren; i++)
{
    s = 0.0;
    for (j=0; j < col; j++)
    {
        s += C[i][j]*u[j];
    }
    v[i] = s;
}
```

En el caso de matrices bandadas, se simplifica a:

```
for (i=0; i<ren; i++)
{
    s= 0.0;
    for (k=0; k < ban; k++)
    {
        if ((Ind[k] + i) >= 0 && (Ind[k]+i) < ren)
            s += Dat[i][k]*u[Ind[k]+i];
    }
    v[i]=s;
}
```

De forma similar, en el caso de matrices dispersas, se simplifica a:

```
for (i=0; i<ren; i++)
{
    s = 0.0, k = 0
    while (Ind[i][k] != -1)
    {
        s += Dat[i][k]*u[Ind[i][k]];
        k++;
        if (k >= b) break;
    }
    v[i] = s;
}
```

De esta forma, al tomar en cuenta la operación de multiplicación de una matriz por un vector, donde el renglón de la matriz involucrado en la multiplicación queda generalmente en una región contigua del Cache, se hace óptima la operación de multiplicación de matriz por vector.

C Bibliografía

Referencias

[1] K. Hutter y K Jöhnk, *Continuum Methods of Physical Modeling*, Springer-Verlag Berlin Heidelberg New York, 2004.

[2] J. L. Lions y E. Magenes, *Non-Homogeneous Bounduary Value Problems and Applicarions* Vol. I, Springer-Verlag Berlin Heidelberg New York, 1972.

[3] A. Quarteroni y A. Valli, *Domain Decomposition Methods for Partial Differential Equations*. Clarendon Press Oxford, 1999.

[4] A. Quarteroni y A. Valli; *Numerical Approximation of Partial Diferential Equations*. Springer, 1994.

[5] B. Dietrich, *Finite Elements: Theory, Fast Solvers*, and Applications in Solid Mechanics, Cambridge University, 2001.

[6] B. F. Smith, P. E. Bjørstad, W. D. Gropp; *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.

[7] Fuzhen Zhang, *The Schur Complement and its Aplications*, Springer, Numerical Methods and Algorithms, Vol. 4, 2005.

[8] B. I. Wohlmuth; *Discretization Methods and Iterative Solvers Based on Domain Decomposition*. Springer, 2003.

[9] L. F. Pavarino, A. Toselli; *Recent Developments in Domain Decomposition Methods*. Springer, 2003.

[10] M.B. Allen III, I. Herrera & G. F. Pinder; *Numerical Modeling in Science And Engineering*. John Wiley & Sons, Inc . 1988.

[11] R. L. Burden y J. D. Faires; *Análisis Numérico*. Math Learning, 7 ed. 2004.

[12] S. Friedberg, A. Insel, and L. Spence; *Linear Algebra*, 4th Edition, Prentice Hall, Inc. 2003.

[13] Y. Saad; *Iterative Methods for Sparse Linear Systems*. SIAM, 2 ed. 2000.

[14] Y. Skiba; *Métodos y Esquemas Numéricos, un Análisis Computacional*. UNAM, 2005.

[15] W. Gropp, E. Lusk, A. Skjellem, *Using MPI, Portable Parallel Programming Whit the Message Passing Interface*. Scientific and Engineering Computation Series, 2ed, 1999.

[16] I. Foster; *Designing and Building Parallel Programs*. Addison-Wesley Inc., Argonne National Laboratory, and the NSF, 2004.

[17] Jorge L. Ortega-Arjona, *Patterns for Parallel Software Design*, Wiley series in Software Design Patterns, 2010.

[18] DDM Organization, *Proceedings of International Conferences on Domain Decomposition Methods*, 1988-2012.

http://www.ddm.org and http://www.domain-decomposition.com

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

[20] Farhat, C. and Roux, F. X. *A Method of Finite Element Tearing and Interconnecting and its Parallel Solution Algorithm.* Int. J. Numer. Meth. Engng., 32:1205-1227, 1991.

[21] Mandel J. & Tezaur R. *Convergence of a Substructuring Method with Lagrange Multipliers*, Numer. Math. 73 (1996) 473-487.

[22] Farhat C., Lesoinne M. Le Tallec P., Pierson K. & Rixen D. *FETI-DP a Dual-Primal Unified FETI method*, *Part 1: A Faster Alternative to the two-level FETI Method*, Int. J. Numer. Methods Engrg. 50 (2001) 1523-1544.

[23] Farhat C., Lesoinne M., Pierson K. *A Scalable Dual-Primal Domain Decomposition Method*, Numer. Linear Algebra Appl. 7 (2000) 687-714.

[24] Mandel J. & Tezaur R. *On the Convergence of a Dual-Primal Substructuring Method*, Numer. Math. 88(2001), pp. 5443-558.

[25] Mandel, J. *Balancing Domain Decomposition.* Comm. Numer. Meth. Engrg., 9:233-241, 1993.

[26] Mandel J., & Brezina M., *Balancing Domain Decomposition for Problems with Large Jumps in Coefficients,* Math. Comput. 65 (1996) 1387-1401.

[27] Dohrmann C., *A Preconditioner for Substructuring Based on Constrained Energy Minimization.* SIAM J. Sci. Comput. 25 (2003) 246-258.

[28] Mandel J. & Dohrmann C., *Convergence of a Balancing Domain Decomposition by Constraints and Energy Minimization.* Numer. Linear Algebra Appl. 10 (2003) 639-659.

[29] Da Conceição, D. T. Jr., *Balancing Domain Decomposition Preconditioners for Non-symetric Problems*, Instituto Nacional de Matemática pura e Aplicada, Agencia Nacional do Petróleo PRH-32, Rio de Janeiro, May. 9, 2006.

[30] J. Li and O. Widlund, *FETI-DP, BDDC and block Cholesky Methods*, Int. J. Numer. Methods Engrg. 66, 250-271, 2005.

[31] Herrera, I., *Theory of Differential Equations in Discontinuous Piecewise-Defined-Functions*, NUMER METH PART D.E., 23(3): 597-639, 2007 OI 10.1002/num.20182.

[32] 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, May 2008 DOI 10.1002/num.20293.

[33] Herrera I. and R. Yates, *Unified Multipliers-Free Theory of Dual Primal Domain Decomposition Methods*. NUMER. METH. PART D. E. 25(3): 552-581, May 2009, (Published on line May 13, 2008) DOI 10.1002/num.20359.

[34] Herrera, I. & Yates R. A., *The Multipliers-Free Domain Decomposition Methods*, NUMER. METH. PART D. E., 26(4): pp 874-905, July 2010. (Published on line: 23 April 2009, DOI 10.1002/num.20462)

[35] Herrera, I., Yates R. A., *The multipliers-Free Dual Primal Domain Decomposition Methods for Nonsymmetric Matrices*. NUMER. METH. PART D. E. DOI 10.1002/num.20581 (Published on line April 28, 2010).

[36] Herrera, I., Carrillo-Ledesma A. and Alberto Rosas-Medina, *A Brief Overview of Non-Overlapping Domain Decomposition Methods*, Geofisica Internacional, Vol, 50, 4, October-December, 2011.

[37] Herrera, I. and Pinder, G. F., *Mathematical Modelling in Science and Engineering: An Axiomatic Approach*, Wiley, 243p., 2012.

[38] Ismael Herrera and Alberto A. Rosas-Medina, *The Derived-Vector Space Framerwork and Four General purposes massively parallel DDM Algorithms*, Engineering Analysis with Boundary Elements, 20013, in press.

[39] Antonio Carrillo-Ledesma, Herrera, I, Luis M. de la Cruz, *Parallel Algorithms for Computational Models of Geophysical Systems*, Geofísica Internacional, en prensa, 2013.

[40] Farhat Ch., Lesoinne M., Le Tallec 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*. Internal. J. Numer. Methods Engrg., 50:1523-1544, 2001.

[41] Rixen, D. and Farhat Ch. *A Simple and Efficient Extension of a Class of Substructure Based Preconditioners to Heterogeneous Structural Mechanics Problems*. Internal. J. Numer. Methods Engrg., 44:489-516, 1999.

[42] J. Mandel, C. R. Dohrmann, and R. Tezaur, *An Algebraic Theory for Primal and Dual Substructuring Methods by Constraints*, Appl. Numer. Math., 54 (2005), pp. 167-193.

[43] A. Klawonn, O. B. Widlund, and M. Dryja, *Dual-primal FETI Methods for Three-Dimensional Elliptic Problems with Heterogeneous Coefficients*, SIAM J. Numer. Anal., 40 (2002), pp. 159-179.

[44] Alberto Rosas Medina, *Métodos de Estabilización para Problemas de Advección-Difusión*, Trabajo de Investigación para Sustentar el Examen de Candidaura al Doctorado, Postgrado en Ciencias de la Tierra, UNAM, 2011.

[45] Iván Germán Contreras Trejo, *Métodos de Precondicionamiento para Sistemas de Ecuaciones Diferenciales Parciales*, Trabajo de Tesis Doctoral en Proceso, Postgrado en Ciencias e Ingeniería de la Computación, UNAM, 2012.

[46] Klawonn A. and Widlund O.B., *FETI and Neumann-Neumann Iterative Substructuring Methods: Connections and New Results*. Comm. Pure and Appl. Math. 54(1): 57-90, 2001.

[47] Tezaur R., *Analysis of Lagrange Multipliers Based Domain Decomposition*. P.H. D. Thesis, University of Colorado, Denver, 1998.

[48] Herrera I. & Rubio E., *Unified Theory of Differential Operators Acting on Discontinuos Functions and of Matrices Acting on Discontinuous Vectors*, 19th International Conference on Domain Decomposition Methods, Zhangjiajie, China 2009. (Oral presentation). Internal report #5, GMMC-UNAM, 2011.

[49] Valeri I. Agoshkov, *Poincaré-Steklov Operators and Domain Decomposition Methods in Finite Dimensional Spaces*. First International Symposium on Domain Decomposition Methods for Partial Differential Equations, pages 73-112, Philadelphia, PA, 1988. SIAM. Paris, France, Junuary 7-9, 1987.

[50] Toselli, A., *FETI Domain Decomposition Methods for Escalar Advection-Diffusion Problems*. Compututational Methods Appl. Mech. Engrg. 190. (2001), 5759-5776.

[51] C.T. Keller, *Iterative Methods for Linear and Nonlinear Equations*, Societe for Industrial and Applied Mathematics, 1995.

[52] Manoj Bhardwaj, David Day, Charbel Farhat, Michel Lesoinne, Kendall Pierson, and Daniel Rixen. *Application of the PETI Method to ASCI Problems: Scalability Results on One Thousand Processors and Discussion of Highly Heterogeneous Problems*. Intemat. J. Numer. Methods Engrg., 47:513-535, 2000.

[53] Zdengk Dostdl and David Hordk. *Scalability and FETI Based Algorithm for Large Discretized Variational Inequalities*. Math. Comput. Simulation, 61(3-6): 347-357, 2003. MODELLING 2001 (Pilsen).

[54] Charbel Farhat, Michel Lesoinne, and Kendall Pierson. *A Scalable Dual-Primal Domain Decomposition Method*. Numer. Linear Algebra Appl., 7(7-8):687-714, 2000.

[55] Yannis Fragakis and Manolis Papadrakakis, *The Mosaic of High Performance Domain Decomposition Methods for Structural Mechanics: Formulation, Interrelation and Numerical Efficiency of Primal and Dual Methods*. Comput. Methods Appl. Mech. Engrg, 192(35-36):3799-3830, 2003.

[56] Kendall H. Pierson, *A family of Domain Decomposition Methods for the Massively Parallel Solution of Computational Mechanics Problems*. PhD thesis, University of Colorado at Boulder, Aerospace Engineering, 2000.

[57] Manoj Bhardwaj, Kendall H. Pierson, Garth Reese, Tim Walsh, David Day, Ken Alvin, James Peery, Charbel Farhat, and Michel Lesoinne. Salinas, *A Scalable Software for High Performance Structural and Mechanics Simulation*. In ACM/IEEE Proceedings of SC02: High Performance Networking and Computing. Gordon Bell Award, pages 1-19, 2002.

[58] Klawonn, A.; Rheinbach, O., *Highly Scalable Parallel Domain Decomposition Methods with an Application to Biomechanics*, Journal of Applied Mathematics and Mechanics 90 (1): 5-32, doi:10.1002/zamm.200900329.

[59] Petter E. Bjørstad and Morten Skogen. *Domain Decomposition Algorithms of Schwarz Type, Designed for Massively Parallel Computers*. In David E. Keyes, Tony F. Chan, Gerard A. Meurant, Jeffrey S. Scroggs, and Robert G. Voigt, editors. Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, pages 362-375, Philadelphia, PA, 1992. SIAM. Norfolk, Virgina, May 6-8, 1991.

[60] Yau Shu Wong and Guangrui Li. *Exact Finite Difference Schemes for Solving Helmholtz Equation at any Wavenumber*. International Journal of Numerical Analysis and Modeling, Series B, Volume 2, Number 1, Pages 91-108, 2011.

[61] Holger Brunst, Bernd Mohr. *Performance Analysis of Large-Scale OpenMP and Hybrid MPI/OpenMP Applications with Vampir NG*. IWOMP 2005: 5-14.

[62] S.J. Pennycook, S.D. Hammond, S.A. Jarvis and G.R. Mudalige, *Performance Analysis of a Hybrid MPI/CUDA Implementation of the NAS-LU Benchmark*. ACM SIGMETRICS Perform. Eval. Rev. 38 (4). ISSN 0163-5999, (2011).

[63] Doxygen, *Gererate Documentation from Source Code*.
http://www.stack.nl/∼dimitri/doxygen/

[64] XMPI, *A Run/Debug GUI for MPI*.
http://www.lam-mpi.org/software/xmpi/

[65] VAMPIR, *Performance Optimization*.
http://www.vampir.eu/

Declaro terminado este trabajo sufrido, ideado y llevado a cabo entre los años 2007 al 2013, aún y a pesar de impedimentos tales como: la mala suerte, la desventura, el infortunio, la incomprensión, la gripe, las horas de frío, la tristeza, la desesperanza, el cansancio, el presente, el pasado y mi futuro, el que dirán, la vergüenza, mis propias incapacidades y limitaciones, mis aversiones, mis temores, mis dudas y en fin, todo aquello que pudiera ser tomado por mi, o por cualquiera, como obstáculo en este tiempo de mentiras, verdades, de incredulidad e ignorancia o negación de la existencia real y física de la mala fe.

Atentamente

Antonio Carrillo Ledesma