UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

Doctor en Ciencias (computación)

PRESENTA:

Iván Germán Contreras Trejo

TUTOR

Dr. Ismael Herrera Revilla

Instituo de Geofísica

Ciudad Universitaria, enero de 2016.

Procesamiento en paralelo de sistemas de EDP 's

Autor: Mtro. Iván Germán Contreras Trejo
Director de Tesis: Dr. Ismael Herrera Revilla

Índice general

1. Introducción **3**
- 1.1. Antecedentes . . . 3
- 1.2. Objetivos de la tesis . . . 5
  - 1.2.1. Objetivos generales . . . 5
  - 1.2.2. Objetivos particulares . . . 5

2. DDM: Ideas generales **6**
- 2.1. Problema continuo . . . 7
- 2.2. Problema discontinuo . . . 8
  - 2.2.1. Una ecuación para u_Γ (Complemento de Schur) . . . 9
  - 2.2.2. Una ecuación para el flujo: λ_Γ . . . 10

3. FEM aplicado al Operador de Elasticidad **12**
- 3.1. Formulación Variacional. . . . 12
- 3.2. Aproximación de la solución. . . . 13

4. Descripción General del DVS **16**
- 4.1. Nociones preliminares y notación . . . 17
  - 4.1.1. Cómo construir discretizaciones sin traslape . . . 21
  - 4.1.2. Los Algoritmos DVS Precondicionados con restricciones . 23
  - 4.1.3. El Algoritmo DVS-BDDC . . . 23
  - 4.1.4. El Algoritmo DVS-Primal . . . 24
  - 4.1.5. El Algoritmo DVS-FETI-DP . . . 24
  - 4.1.6. El Algoritmo DVS-Dual . . . 24
  - 4.1.7. Algoritmos DVS aplicados a elasticidad . . . 24

5. Visión General del Software DVS **27**
- 5.1. Las piezas elementales del software DVS . . . 27
- 5.2. Construcción del software DVS . . . 28
- 5.3. Construcción del software local del DVS . . . 29
  - 5.3.1. La aplicación de $\underline{\underline{a}}$ . . . 30
  - 5.3.2. El software DVS para $\underline{\underline{S}}$ y $\underline{\underline{S}}^{\sim\mathbf{1}}$ . . . 30
  - 5.3.3. El software DVS local para $\underline{\underline{S}}$ . . . 31
  - 5.3.4. El software DVS local para $(\underline{\underline{A}}^{\sim\mathbf{1}}_{\mathbf{\Pi\Pi}}$ . . . 32

5.3.5. El software DVS local para $\underline{\underline{S}}^{\sim \mathbf{1}}$ 33
5.3.6. Aplicaciones del Método de Gradiente Conjugado (CGM) 33

6. DDM con DVS y su implementación computacional. 34
6.1. Análisis del sistema general 35
6.2. Paralelización 37
6.3. Geometría del problema 38
6.4. Método de discretización numérico (FEM) 40
6.5. Diseño del DVS 42
6.6. Arquitectura del software 44
6.7. Descripción del software 46
6.7.1. Descripción del comportamiento dinámico del software. . 47

7. Experimentos Numéricos 51

8. Conclusiones 56

A. Algoritmos 62
A.0.1. Algoritmo Dirichlet-Neumann 62
A.1. Algoritmo Neumann-Neumann 63
A.2. Algoritmo Dirichlet-Dirichlet o FETI 63

B. Complemento de Schur 65
B.1. Sistema de ecuaciones 65
B.2. Formulación convencional 66
B.3. Formulación con funciones discontinuas 66
B.4. Solución del sistema en paralelo usando CGM 67
B.4.1. Formulación Convencional 68
B.4.2. Formulación con funciones discontinuas 70

Capítulo 1

Introducción

1.1. Antecedentes

Los modelos matemáticos que aparecen en ciencia e ingeniería llevan a sistemas de ecuaciones diferenciales parciales (PDE's) [1] cuyos métodos de solución están basados en el procesamiento computacional de sistemas algebraicos de gran escala y el avance de muchas áreas, particularmente la de las ciencias de la tierra, depende de la aplicación del hardware computacional más potente a esos sistemas [2].

El cómputo en paralelo sobresale entre las herramientas computacionales, especialmente en el presente cuando los incrementos en la velocidad del hardware aparentemente han alcanzado barreras que parecían insuperables. Como es bien sabido, las principales dificultades del cómputo en paralelo están asociadas con la coordinación de muchos procesadores que llevan a cabo diferentes tareas así como de la transmisión de información. De manera ideal, dada una tarea, estas dificultades desaparecen cuando "dicha tarea es efectuada por procesadores que trabajan independientemente unos de otros". Nos referiremos a esta última condición como "el paradigma del software del cómputo en paralelo".

El surgimiento del cómputo en paralelo promovió en buena parte la comunidad de la modelación computacional un esfuerzo continuo y sistemático con el propósito de dirigirlo con el fin de resolver modelos matemáticos de sistemas de la ciencia y de la ingeniería. Muy tempranamente justo después de que ese esfuerzo comenzó, se reconoció que los métodos de descomposición de dominio (DDM por sus siglas en inglés) eran las técnicas más efectivas para la aplicación del cómputo en paralelo a la solución de ecuaciones diferenciales parciales [3], debido a que dicha aproximación simplifica drásticamente la coordinación de muchos procesadores que llevan a cabo las diferentes tareas y también reduce bastante los requerimientos de transmisión de la información entre ellos [4]-[11].

Cuando algún DDM es aplicado, primero se lleva a cabo la discretización de un modelo matemático en una malla fina y, una vez hecho esto, se introduce una malla gruesa, lo cual propiamente constituye la descomposición de dominio.

'El paradigma del DDM', un paradigma para los métodos de descomposición de dominio concomitante con el paradigma del software del cómputo en paralelo [12], consiste en 'obtener la solución global a través de la solución de problemas locales exclusivamente (un problema local es uno definido separadamente en un subdominio de la malla gruesa). Dicho de una manera sencilla, la idea básica es que, cuando el paradigma de los DDM se satisface, una completa paralelización puede ser llevada a cabo por la asignación de cada subdominio a un procesador diferente.

Cuando comenzó la etapa más intensa de la investigación de los DDM se le dio mucho más atención a los DDM con traslape, pero muy pronto la atención cambio a los DDM sin traslape. Cuando el paradigma de los DDM se tomó en cuenta, esta evolución pareció muy natural porque es más fácil desacoplar los problemas locales cuando los subdominios no se traslapan. Sin embargo, aún en este tipo de métodos los distintos subdominios están ligados por los nodos de interface que son compartidos por muchos subdominios y, por lo tanto, los DDM sin traslape son de hecho traslapados cuando son vistos desde la perspectiva de los nodos usados en la discretización. Así, un desacoplamiento mayor de los problemas locales y ventajas computacionales muy significativas deberían esperarse si fuera posible llevar a cabo la discretización de las ecuaciones diferenciales en un sistema de nodos sin traslape [12] ; esto es, un conjunto de nodos con la propiedad de que cada uno de ellos pertenezca a un y solo un subdominio de la malla gruesa (esta malla que constituye la descomposición de dominio). En [12], así como en todo lo que sigue, a los métodos de discretización que satisfacen estas condiciones se les referirá como discretizaciones sin traslape.

En una línea de investigación, a la que pertenece este artículo, Herrera y colaboradores abordaron este problema y lo resolvieron habiendo desarrollado un marco –el marco del DVS- detalladamente formulado usando una discretización sin traslape de las ecuaciones diferenciales parciales originales. Debido a las propiedades de las discretizaciones sin traslape en dichos algoritmos los enlaces entre los diferentes procesadores son mucho más relajados, y también la transmisión de información requerida entre ellos se reduce. Dichas propiedades, también conocidas como el análisis preliminar de los algoritmos, indican que son muy adecuados para programar los procedimientos para resolver las ecuaciones diferenciales parciales que aparecen en los modelos de la ciencia y la ingeniería con el hardware más altamente paralelizado de la actualidad.

A pesar que los algoritmos DVS ya han sido significativamente desarrollados y algunos ejemplos han sido tratados previamente [12]-[**?**], hasta ahora ningún software que tomara ventaja de los algoritmos DVS había sido desarrollado. Es claro que para tener una amplia ganancia de esos avances es esencial desarrollar un software, cuidadosamente codificado, el cual permite aplicar efectivamente los algoritmos DVS a problemas de interés de la ciencia y la ingeniería. Como una notable contribución de esos avances, en este trabajo, por primera vez presentamos y probamos un software de esas características.

1.2. Objetivos de la tesis

En este trabajo de tesis se presenta el enfoque DVS y los métodos de descomposición de dominio que de ahí surgen pero adecuados para sistemas de Ecuaciones Diferenciales Parciales. Se demuestra que los métodos que originalmente aplican a una EDP también son validos para los sistemas de ecuaciones con ligeras modificaciones que son necesarias como el manejo entre funciones valuadas escalarmente y las valuadas vectorialmente. La ecuación que se toma es la de elastoestática que es un operador valuado vectorialmente y del cual surge un sistema de ecuaciones algebraico con un número muy alto de grados de libertad.

1.2.1. Objetivos generales

- Revisar los métodos de Descomposición de Dominio para establecer las semejanzas y diferencias con el resto de las metodologías tomando como referencia los métodos que provienen de los DVS.
- Dar las modificaciones necesarias para la adecuación de los algoritmos obtenidos en el enfoque DVS al caso de sistemas de ecuaciones diferenciales parciales para demostrar su generalidad.
- Implementar dichos algoritmos en máquinas de alto rendimiento, específicamente en un cluster, para demostrar la eficiencia computacional que puede alcanzarse.

1.2.2. Objetivos particulares

1. Resolver un caso sencillo de la ecuación de elasticidad en tres dimensiones.
2. Demostrar la convergencia, la escalabilidad y la eficiencia de los algoritmos implementados.
3. El uso de cómputo en paralelo para resolver sistemas con un alto número de incógnitas.
4. Acceder a recursos robustos de cómputo en paralelo para verificar el código y medir su eficiencia.

Capítulo 2

DDM: Ideas generales

Obtener una solución numérica de una Ecuación Diferencial Parcial (EDP) en un dominio puede ser muy costoso computacionalmente. Esto debido a que resolver los sistemas lineales algebraicos que surgen de la aplicación de algún método de discretización requiere un número de operaciones proporcionales a la dimensión de la matriz que se obtuvo de la discretización de la EDP.

La idea de resolver una EDP en distintos dominios locales y posteriormente obtener una solución global con base en las soluciones locales obtenidas no es nueva [4]. Aunque estas ideas surgen muchos años antes de la aparición de las computadoras digitales y su interés era para dominios que por su forma irregular pudieran separarse en dominios con formas regulares , esas mismas ideas son usadas actualmente para dividir el dominio discretizado en subdominios. Estas técnicas son denominadas Métodos de Descomposición de Dominio (DDM por sus siglas en inglés).

Con el auge del cómputo en paralelo se han retomado las ideas para resolver numéricamente EDP aplicando esos procedimientos , técnicas y metodologías que optimicen el uso de los procesadores así como el de las comunicaciones.

El procedimiento por el cual se generan los subdominos consiste en introducir una partición que separe los nodos de la discretización original en subconjuntos ajenos. A dichas discretizaciones más gruesas se les denomina subdominios. Esto a su vez genera una frontera interior denominada Γ que es la interface entre los subdominios. A su vez se construyen matrices en cada subdominio, que son equivalentes a las del sistema global.

Los procedimientos que se usan para obtener las soluciones de los sistemas de ecuaciones locales se puede recurrir tanto a métodos directos como a métodos iterativos. Mientras que la solución en los nodos de interface se hace exclusivamente con un método iterativo.

Como se explica más adelante, la idea fundamental es generar una clasificación de los nodos de la discretización en nodos que pertenecen a la interface y los que no y usar eliminación Gaussiana por bloques para llevar a cabo los despejes los cuales determinarán la factorización del sistema. Después se generan precondicionadores que aceleran la convergencia del método iterativo.

Las metodologías convencionales actúan de la siguiente manera, dada una discretización obtenida con el método de elemento finito (FEM) o con el método de diferencia finitas (FDM) en cada subdominio se obtiene la solución local tomando condiciones de frontera Dirichlet o Neumann y después se genera la solución del sistema sobre los nodos que están sobre Γ por un método iterativo ajustando las soluciones locales sobre los subdominios y de esta forma se continúa alternando hasta la convergencia de la solución.

2.1. Problema continuo

Consideremos el operador diferencial de Poisson $(-\triangle)$ en una EDP que se cumple en un dominio Ω dado con condiciones de frontera Dirichlet homogéneas en la frontera $\partial\Omega$ del dominio.

Dado el problema

$$-\triangle u = f \text{ en } \Omega \tag{2.1.1}$$

$$u = 0 \text{ en } \partial\Omega \tag{2.1.2}$$

Se lleva a cabo la descomposición del dominio la cual consiste en generar subdominios sin traslape y con fronteras artificiales. Para las aproximaciones numéricas se tienen dos casos con lo que respecta a la discretización de los subdominios, el caso en que las mallas coinciden y en el que no coinciden. En este trabajo de investigación consideramos únicamente el primer caso. Así la discretización esta dada como

$$\Omega = \{\Omega_1 \cap \Omega_2 \cap \cdots \cap \Omega_n\} \tag{2.1.3}$$

Con ello se genera una frontera interior Γ además de la frontera exterior $\partial\Omega$. Si consideramos $\bar{\Omega} = \Omega \cup \partial\Omega$ y como Γ esta en el interior de Ω entonces $\Gamma_i \subset \partial\Omega_i$.

La ecuación original en dos subdominios es equivalente al siguiente sistema de ecuaciones

$$-\triangle u_1 = f \text{ en}\Omega_{\ 1} \tag{2.1.4}$$

$$u_1 = 0 \text{ en } \partial\Omega_1|\Gamma \tag{2.1.5}$$

$$u_1 = u_2 \text{ sobre } \Gamma \tag{2.1.6}$$

$$\frac{\partial u_1}{\partial n_1} = -\frac{\partial u_2}{\partial n_2} \text{ sobre } \Gamma \tag{2.1.7}$$

$$-\triangle u_2 = f \text{ en}\Omega_{\ 2} \tag{2.1.8}$$

$$u_2 = 0 \text{ en } \partial\Omega_2\ \Gamma \tag{2.1.9}$$

2.2. Problema discontinuo

(2.1.4) A partir de la representación del problema en cada subdominio en términos de funciones y operadores continuos seremos capaces de reescribir el problema en forma matricial, mediante la discretización dada por FDM o FEM en cada subdominio como un sistema lineal algebraico de la forma

$$Au = f \tag{2.2.1}$$

donde A es una matriz simétrica y positiva definida.

La matriz se divide en grados de libertad para el interior deΩ_i y grados de libertad sobre la frontera interiorΓ , con condiciones Dirichlet en $\partial\Omega_i|\Gamma$ y condiciones Neumann enΓ.

Si consideramos únicamente dos subdominios entonces la matriz tiene la siguiente forma

$$A = \begin{bmatrix} A_{II}^{(1)} & 0 & A_{I\Gamma}^{(1)} \\ 0 & A_{II}^{(2)} & A_{I\Gamma}^{(2)} \\ A_{\Gamma I}^{(1)} & A_{\Gamma I}^{(2)} & A_{\Gamma\Gamma} \end{bmatrix}; u = \begin{bmatrix} u_I^{(1)} \\ u_I^{(2)} \\ u_\Gamma \end{bmatrix}; f = \begin{bmatrix} f_I^{(1)} \\ f_I^{(2)} \\ f_\Gamma \end{bmatrix}$$

Que de la misma forma se suman los vectores definidos sobre Γ pero en distintos subdominios.

$$A_{\Gamma\Gamma} = A_{\Gamma\Gamma}^1 + A_{\Gamma\Gamma}^2 \tag{2.2.2}$$

$$f_\Gamma = f_\Gamma^1 + f_\Gamma^2 \tag{2.2.3}$$

Las condiciones sobre Γ se llaman condiciones de transmisibilidad y son equivalentes a la igualdad con la combinación lineal de las trazas de las funciones y sus derivadas normales. A la derivada normal también se le conoce como el flujo.

Dada una expresión matricial donde se distinguen los nodos interiores de cada subdominio y los nodos sobreΓ , se busca una aproximación de las derivadas normales sobreΓ.

Si para el problema continuo se da una solución u_i exacta, es decir, que se satisface la ecuación enΩ_i, el flujo se puede definir como un funcional bilineal usando la fórmula de Green. Si damos una solución del Método de Elemento Finito en la funcional bilineal discretizando el término de más a la izquierda, tomando ϕ_j como función base en el nodo j y recorriendo todos los nodos sobre Γ obtenemos una expresión para la derivada normal

$$\int_\Gamma \frac{\partial u_i}{\partial n_i}\phi_j ds = \int_{\Omega_i} (\triangle u_i \phi_j + \nabla u_i \cdot \nabla \phi_j) dx = \int_{\Omega_i} (-f\phi_j + \nabla u_i \cdot \nabla \phi_j) dx \tag{2.2.4}$$

que al construir las matrices correspondientes se obtiene

$$\lambda^i = A_{\Gamma I}^{(1)} u_I^{(i)} + A_{\Gamma\Gamma}^i u_\Gamma^i - f_\Gamma^i \tag{2.2.5}$$

En este caso lo que ocurra en Γ tiene que ver con el flujo por lo tanto es un problema tipo Neumann.

$$A^1_{II}u^1_I + A^1_{I\Gamma}u^1_\Gamma = f^1_I \tag{2.2.6}$$
$$u^1_\Gamma = u^2_\Gamma = u_\Gamma \tag{2.2.7}$$
$$A^1_{\Gamma I}u^1_I + A^1_{\Gamma\Gamma}u^1_\Gamma - f^1_\Gamma = -A^2_{\Gamma I}u^2_I + A^2_{\Gamma\Gamma}u^2_\Gamma - f^2_\Gamma = \lambda_\Gamma \tag{2.2.8}$$
$$A^2_{II}u^2_I + A^2_{I\Gamma}u^2_\Gamma = f^2_I \tag{2.2.9}$$

Este problema tiene en cada subdominio condiciones de frontera Dirichlet sobre $\partial\Omega_i|\Gamma$ y condiciones Neumann enΓ . Además un problema local de Poisson en cada subdominio. Este sistema de ecuaciones es equivalente al problema original continuo , pero mientras este último es válido sólo si el lado derecho es lo suficientemente regular, su contraparte discreta es valida siempre, dado que puede ser obtenida directamente de FEM.

Lo anterior nos permite plantear problemas para resolver u_Γ o para λ_Γ. La base de los subsecuentes métodos es el sistema de ecuaciones (2.2.6 - 2.2.9) se considera de manera completa o se eliminan algunas ecuaciones de acuerdo con las condiciones de frontera interior impuestas.

2.2.1. Una ecuación para u_Γ (Complemento de Schur)

Al igual que otro métodos de descomposición de dominio iterativos, las incógnitas en el interior del subdominio son eliminadas del sistema. Lo anterior despejando en alguna ecuación estas incógnitas y sustituyendo en las restantes. Este procedimiento se puede hacer por bloques y se denomina Eliminación Gaussiana por bloques. Se toman tres ecuaciones del sistema anterior para obtener

$$A = \begin{bmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} u_I \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_I \\ f_\Gamma \end{bmatrix} \tag{2.2.10}$$

Despejando $u^{(i)}_I$

$$u^{(i)}_I = A^{(i)-1}_{II}(f^{(i)}_I - A^{(i)}_{I\Gamma}u^{(i)}_\Gamma) \tag{2.2.11}$$

obtenemos

y sustituyendo

$$(A^{(i)}_{\Gamma\Gamma} - A^{(i)}_{\Gamma I}A^{(i)-1}_{II}A^{(i)}_{I\Gamma})u^{(i)}_\Gamma = f^{(i)}_\Gamma - A^{(i)}_{\Gamma I}A^{(i)-1}_{II}f^{(i)}_I \tag{2.2.12}$$

Donde definimos el complemento de Schur, nuevamente con dos subdominios, como

$$S = A_{\Gamma\Gamma} - A^1_{\Gamma I}A^{(1)^{-1}}_{II}A^1_{I\Gamma} - A^2_{\Gamma I}A^{(2)^{-1}}_{II}A^2_{I\Gamma} \tag{2.2.13}$$

que es un despeje utilizando eliminación gaussiana por bloques y que es equivalente al operador Steklov-Poicaré el cual mapea de las trazas de las funciones

a las funciones en el interior de los subdominios , ya que lo hace de manera exacta las condiciones de flujo son cero. Mientras que el lado derecho esta dado como

$$g_\Gamma = \left(f_\Gamma^1 - A_{\Gamma I}^1 {A_{II}^{(1)}}^{-1} f_I^{(1)}\right) + \left(f_\Gamma^2 - A_{\Gamma I}^2 {A_{II}^{(2)}}^{-1} f_I^{(1)}\right) \tag{2.2.14}$$

del cual se despejaron las incógnitas que están sobre Γ y se eliminan, por sustitución, las que están sobre$\Omega_{\ i}$ Si reensamblamos el sistema anterior para dos subdominios obtenemos

$$\begin{bmatrix} A_{II}^{(1)} & 0 & A_{I\Gamma}^{(1)} \\ 0 & A_{II}^{(2)} & A_{I\Gamma}^{(2)} \\ 0 & 0 & S \end{bmatrix} u = \begin{bmatrix} f_I^{(1)} \\ f_I^{(2)} \\ g_\Gamma \end{bmatrix} \tag{2.2.15}$$

El sistema se resuelve primero con algún método iterativo como Gradiente Conjugado para la ecuación

$$Su_\Gamma = g_\Gamma \tag{2.2.16}$$

Para obtener los valores en el interior de los subdominios se utiliza la expresión

$$u_I^i = {A_{II}^{(1)}}^{-1}(f_I^{(i)} - A_{I\Gamma}^i u_\Gamma) \tag{2.2.17}$$

2.2.2. Una ecuación para el flujo: λ_Γ

A diferencia del método anterior, ahora sí consideramos que los valores de la función en la frontera interior Γ están determinados por la diferencia de los valores de u en el interior de dos subdominios adyacentes. Es decir por el flujo. Se consideran todas las ecuaciones lo que da el sistema de ecuaciones Además de las ecuaciones locales, para el flujo se considera la tercera ecuación la cual describe el flujo. Su condición de transmisibilidad esta dada por

$$\lambda_\Gamma = \lambda_\Gamma^{(1)} = -\lambda_\Gamma^{(2)} \tag{2.2.18}$$

Para el caso en dos subdominios se resuelven problemas locales tipo Neumann para encontrar $u^{(1)}$ y $u^{(2)}$.

$$\begin{bmatrix} A_{II}^{(i)} & A_{\Gamma I}^{(i)} \\ A_{I\Gamma}^{(i)} & A_{\Gamma\Gamma}^{(i)} \end{bmatrix} \begin{bmatrix} u_I^{(i)} \\ u_\Gamma^{(i)} \end{bmatrix} = \begin{bmatrix} f_I^{(i)} \\ f_\Gamma^{(i)} + \lambda_\Gamma^{(i)} \end{bmatrix} \tag{2.2.19}$$

Al igual que en el caso anterior aplicamos eliminación gaussiana por bloques (factorización) y se obtiene el valor de $u_\Gamma^{(i)}$ despejado.

$${S^{(i)}}^{-1}(g_\Gamma^i + \lambda_\Gamma^{(i)}) \tag{2.2.20}$$

Tomando la condición de transmisibilidad

$$u_\Gamma^{(1)} = u_\Gamma^{(2)} \tag{2.2.21}$$

y sustituyendo con (2.2.20) se obtiene

$$S^{(1)^{-1}}(g_\Gamma^1 + \lambda_\Gamma^{(1)}) = S^{(2)^{-1}}(g_\Gamma^2 + \lambda_\Gamma^{(2)}) \tag{2.2.22}$$

desarrollando

$$S^{(1)^{-1}} g_\Gamma^1 + S^{(1)^{-1}} \lambda_\Gamma^{(1)} = S^{(2)^{-1}} g_\Gamma^2 + S^{(2)^{-1}} \lambda_\Gamma^{(2)} \tag{2.2.23}$$

factorizando y considerando que el valor de $\lambda^{(1)}\Gamma = -\lambda^{(2)}\Gamma = \lambda_\Gamma$

$$(S^{(1)^{-1}} + S^{(2)^{-1}})\lambda_\Gamma = -(S^{(1)^{-1}} g_\Gamma^1 + S^{(2)^{-1}} g_\Gamma^2) \tag{2.2.24}$$

renombrando

$$F = S^{(1)^{-1}} + S^{(2)^{-1}} \tag{2.2.25}$$

$$d_\Gamma = -(S^{(1)^{-1}} g_\Gamma^1 + S^{(2)^{-1}} g_\Gamma^2) \tag{2.2.26}$$

Se obtiene el sistema

$$F\lambda_\Gamma = d_\Gamma \tag{2.2.27}$$

Una vez que λ_Γ es conocida podemos encontrar la solución para u^1 y u^2 resolviendo los problemas locales.

Capítulo 3

FEM aplicado al Operador de Elasticidad

3.1. Formulación Variacional.

Sea el operador diferencial

$$\mathscr{L}\underline{u} = -(\lambda+\mu)\nabla(\nabla\cdot\underline{u}) - \mu\triangle\underline{u} \text{ en } \Omega \tag{3.1.1}$$

Dada la ecuación diferencial

$$\mathscr{L}\underline{u} = \underline{f}_{\Omega} \tag{3.1.2}$$

$$\underline{u} \equiv u_{\partial} \text{ en } \partial\Omega \tag{3.1.3}$$

Si $\underline{w}$ es una función en Ω tal que $\nabla\underline{w}$ existe, entonces

$$\underline{w}\cdot\mathscr{L}\underline{u} = -(\lambda+\mu)(\underline{w}\cdot\nabla(\nabla\cdot\underline{u})) - \mu\underline{w}\cdot\triangle\underline{u} \tag{3.1.4}$$

Se detallan las operaciones involucradas para su mejor comprensión.

Sea la operación $\nabla(\nabla\cdot\underline{u})$ la cual es el gradiente de una divergencia, la expresamos en notación indexada como

$$\nabla(\nabla\cdot\underline{u}) = \frac{\partial}{\partial x_i}\frac{\partial u_j}{\partial x_j} \tag{3.1.5}$$

pesamos con $\underline{w}$ e integramos

$$\int_{\Omega} w_i\frac{\partial}{\partial x_i}\frac{\partial u_j}{\partial x_j}d\underline{x} = -\int_{\Omega}\frac{\partial w_i}{\partial x_i}\frac{\partial u_j}{\partial x_j}d\underline{x} + \int_{\partial\Omega} w_i\frac{\partial u_j}{\partial x_j}d\underline{x} \tag{3.1.6}$$

Mientras que la operación $\triangle\underline{u}$, siendo el laplaciano es la divergencia de un gradiente y expresada en notación indexada como

$$\triangle\underline{u} \equiv \nabla\nabla\underline{u} = \frac{\partial}{\partial x_i}\frac{\partial u_j}{\partial x_i} \tag{3.1.7}$$

Pesamos con $\underline{w}$

$$\underline{w}\triangle\underline{u} \equiv \sum_{i=1}^{3} w_i \triangle u_i \tag{3.1.8}$$

Integrando y expresando la fórmula componente a componente

$$\int_\Omega \underline{w}\triangle\underline{u}d\underline{x} = \int_\Omega w_i \frac{\partial^2 u_i}{\partial x_j \partial x_j} d\underline{x} \tag{3.1.9}$$

y después aplicando integración por partes

$$\int_\Omega w_i \frac{\partial^2 u_i}{\partial x_j \partial x_j} d\underline{x} = -\int_\Omega \frac{\partial u_i}{\partial x_j}\frac{\partial w_i}{\partial x_j} d\underline{x} + \int_{\partial\Omega} w_i \frac{\partial u_i}{\partial x_j} n_j d\underline{x} \tag{3.1.10}$$

Se toma en cuenta que las funciones de peso se anulan en $\partial\Omega$ por lo que los términos en (3.1.6) y (3.1.10) se anulan y al sustituir con los términos que restan de esas mismas ecuaciones en (3.1.4) obtenemos la formulación variacional a partir de la cual se modelará computacionalmente.

$$\int_\Omega \{(\lambda+\mu)(\nabla\cdot\underline{w})(\nabla\cdot\underline{u}) + \mu\nabla\underline{w} : \nabla\underline{u}\} d\underline{x} = \int_\Omega \underline{w}\cdot\underline{f}_\Omega d\underline{x} \tag{3.1.11}$$

Que es una formulación débil.

3.2. Aproximación de la solución.

Si $u_\partial(\underline{x})$ es una función que cumple las condiciones de frontera:

$$\underline{u}_\partial(\underline{x}) = \underline{g}(\underline{x}) \tag{3.2.1}$$

y si definimos

$$\underline{v}(\underline{x}) = \underline{u}(\underline{x}) - \underline{u}_\partial(\underline{x}) \tag{3.2.2}$$

entonces $\underline{v}(\underline{x}) = 0$ en $\partial\Omega$
Definimos el espacio de funciones

$$V = \{\underline{\phi}(\underline{x}) : \nabla\underline{\phi}(\underline{x}) \text{ existe en } \Omega \text{ y } \underline{\phi}(\underline{x}) = 0 \text{ cuando } \underline{x}\in\partial\Omega\} \tag{3.2.3}$$

Si $\underline{v},\underline{w}\in V$ y $\underline{v}(\underline{x}) = \underline{u}(\underline{x}) - \underline{u}_\partial(\underline{x})$ podemos decir que

$$A(\underline{u},\underline{w}) = \int_\Omega \underline{f}_\Omega\cdot\underline{w}d\underline{x} - A(\underline{u}_\partial,\underline{w}) \tag{3.2.4}$$

Ahora se construye un espacio de funciones $M_h \subset V$ de dimensión finita que sirve para aproximar a la solución la función $\underline{v}(\underline{x})$.
Las funciones de M_h son de la forma

$$\underline{\phi} = (\phi_1,\phi_2,\phi_3) \tag{3.2.5}$$

Asumimos que M_h cuenta con una base

$$M_h = << \underline{\psi}_1, \cdots, \underline{\psi}_N >> \tag{3.2.6}$$

Donde cada una de las funciones de M_h esta definida en un punto de Ω. Con esto se genera un espacio de funciones de dimensión finita.

Definimos así la función bilineal

$$A(\underline{u}, \underline{w}) = \int_\Omega \{(\lambda + \mu)(\nabla \cdot \underline{w})(\nabla \cdot \underline{u}) + \mu \nabla \underline{w} : \nabla \underline{u}\} d\underline{x} \tag{3.2.7}$$

para $\underline{u}, \underline{w} \in M_h$

Las funciones de la base son de la forma

$$\underline{\psi}_{\alpha 1} = (\psi_\alpha(\underline{x}), 0, 0) \tag{3.2.8}$$

$$\underline{\psi}_{\alpha 2} = (0, \psi_\alpha(\underline{x}), 0) \tag{3.2.9}$$

$$\underline{\psi}_{\alpha 3} = (0, 0, \psi_\alpha(\underline{x})) \tag{3.2.10}$$

A su vez se considera que $\psi_\alpha(\underline{x})$ es una función que cumple con la condición de que

$$\psi_\alpha(\underline{x}_\beta) = \delta_{\alpha\beta} \tag{3.2.11}$$

Que de manera explícita considerando los polinomios de Lagrange en cada eje dimensional se tiene que

$$\left(\underline{\psi}_{\alpha i}(\underline{x})\right)_j = \ell_\alpha(\underline{x})\, \delta_{ij} \tag{3.2.12}$$

en la que las funciones en cada componente se definen como

$$\ell_\alpha(\underline{x}) = l_{\alpha 1}(x_1) l_{\alpha 2}(x_2) l_{\alpha 3}(x_3) \tag{3.2.13}$$

y la definición en cada variable de esas funciones $l_{\alpha i}(x_i)$ valuadas escalarmente esta dada como

$$l_{\alpha i}(x_i) = \begin{cases} \frac{x_{i_{k+1}} - x_i}{x_{i_{k+1}} - x_{i_k}} & x_{i_k} \le x_i \le x_{i_{k+1}} \\ \frac{x_i - x_{i_{k-1}}}{x_{i_k} - x_{i_{k-1}}} & x_{i_{k-1}} \le x_i \le x_{i_k} \\ 0 & \text{de cualquier otra forma} \end{cases} \tag{3.2.14}$$

Entonces si $\underline{\hat{v}}(\underline{x})$ es el elemento que aproxima a $\underline{v}(\underline{x})$, podemos escribir:

$$\underline{\hat{v}}(\underline{x}) = \sum_{\alpha=1}^{N} \sum_{i=1}^{3} V_{\alpha i} \underline{\psi}_{\alpha i}(\underline{x}) \tag{3.2.15}$$

o tomando las componentes vectoriales

$$\hat{v}_j(\underline{x}) = \sum_{\alpha=1}^{N} \sum_{i=1}^{3} V_{\alpha i} \underline{\psi}_{\alpha i}(\underline{x}) \delta_{ij} \text{ para } j = 1, 2, 3 \tag{3.2.16}$$

Sustituyendo $\underline{u}$ con $\hat{\underline{v}}$ en (3.2.4) tenemos

$$\sum_{\alpha=1}^{N}\sum_{i=1}^{3} V_{\alpha i} A(\underline{\psi}_{\alpha i}, \underline{w}) = \int_{\Omega} \underline{f}_{\Omega} \cdot \underline{w} d\underline{x} - A(\underline{u}_{\partial}, \underline{w}) \tag{3.2.17}$$

sustituyendo $\underline{w} = \psi_{\alpha}$ la ecuación anterior la podemos expresar como

$$\sum_{\alpha=1}^{N}\sum_{i=1}^{3} V_{\alpha i} A(\underline{\psi}_{\alpha i}, \underline{\psi}_{\beta j}) = \int_{\Omega} \underline{f}_{\Omega} \cdot \underline{\psi}_{\beta j} d\underline{x} - A(\underline{u}_{\partial}, \underline{\psi}_{\beta j}) \text{ con } \beta = 1, \cdots, N; j = 1, 2, 3 \tag{3.2.18}$$

Que de forma concreta es

$$\sum_{\alpha=1}^{N}\sum_{i=1}^{3} V_{\alpha i} \int_{\Omega_{\alpha}} \{(\lambda + \mu)((\nabla \cdot \underline{\psi}_{\alpha_i})(\nabla \cdot \underline{\psi}_{\beta j})) + \mu(\nabla \underline{\psi}_{\alpha i} : \nabla \underline{\psi}_{\beta j})\} d\underline{x} \tag{3.2.19}$$

En forma indexada se expresa como

$$\sum_{\alpha=1}^{N}\sum_{i=1}^{3} V_{\alpha i} \int_{\Omega_{\alpha}} \{(\lambda+\mu)(\frac{\partial \psi_{\alpha}}{\partial x_i}\frac{\partial \psi_{\beta}}{\partial x_j}) + \mu(\frac{\partial \psi_{\alpha}}{\partial x_l}\frac{\psi_{\beta}}{\partial x_l}\delta_{ij})\} d\underline{x} = \int_{\Omega} f_j \psi_{\beta} \text{ con } \beta = 1, \cdots, N; j = 1, 2, 3 \tag{3.2.20}$$

Capítulo 4

Descripción General del DVS

El marco del *espacio de vectores derivados* (*el marco de trabajo del DVS*) trata con la matriz que se obtiene después de que una ecuación diferencial (EDP), o un sistema de dichas ecuaciones ha sido discretizado por medio de algún procedimiento de discretización estándar (esto es, por algún *método de discretización con traslape*). Al sistema de ecuaciones discretizado resultante se le referirá como el *sistema original*.

El procedimiento DVS sigue los siguientes pasos:
1. La ecuación diferencial parcial, o sistema de dichas ecuaciones, esta discretizado por algún método estándar ,que satisface los axiomas de la teoría establecida en 4.1.1, en una malla – llamada la *malla fina* – para obtener un problema discreto que está escrito como

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

A este se le llama el *problema original*, mientras que los nodos de la *malla fina* son llamados *nodos originales*. El símbolo $\widehat{\mathrm{X}}$ será usado para referirse a todo el conjunto de nodos originales; cualquier función definida en el conjunto $\widehat{\mathrm{X}}$ por definición es un *vector original*. Finalmente, la notación $\widehat{W}$ será usada para el espacio lineal generado por los vectores originales, el cual en su momento se le llamará espacio de vectores originales.
2. Se introduce una *malla gruesa*, la cual constituye una descomposición sin traslape del dominio del problema. El sistema de *nodos originales* aparece con traslape con respecto a dicha malla gruesa.
3. Un sistema de *nodos sin traslape* (*nodos derivados*), denotado por X, es construido aplicando el procedimiento explicado en artículos previos (véase sección

4.1). Las funciones definidas en todo el conjunto X son por definición los *vectores derivados* y la notación W ses usada para todo el espacio lineal de estos *vectores derivados*, el cual constituye el *espacio de vectores derivados*.
4. La teoría del *marco del DVS* provee una fórmula que permite trasformar la *discretización original* en una *discretización sin traslape*. Aplicando esta fórmula se obtiene la *discretización sin traslape*. Esta es otra formulación discreta que es equivalente al *problema original*, excepto que esta constituye una *discretización sin traslape*; y
5. Después de todo lo anterior, cada uno de los subdominios de la malla gruesa es asignado a un procesador diferente y el código es programado de forma separada en cada uno de los procesadores.

El marco teórico del DVS es muy elegante; en él, las operaciones algebraicas pueden ser llevadas a cabo sistemáticamente y con gran simplicidad. Además, muchos resultados algebraicos simplificados han sido obtenidos en trabajos previos. Para optimizar las comunicaciones y el tiempo de procesamiento se definen rutas críticas puramente algebraicas, las cuales dan una ganancia aún mayor de los resultados algebraicos obtenidos previamente. Entonces esta ruta crítica algebraica es transformada en código usando C++ y muchas otras técnicas computacionales bien establecidas como lo sería MPI.
Siguiendo los pasos indicados más arriba, en el presente trabajo se ha desarrollado un software para problemas de sólidos elásticos isotrópicos en equilibrio que se ha probado experimentalmente. La alta eficiencia en la paralelización del software así obtenido ha sido verificada experimentalmente. Para ser más específicos, solamente el algoritmo DVS-BDDC ha sido implementado para este problema. Sin embargo, por simples combinaciones de las rutinas ya desarrolladas los otros algoritmos DVS pueden ser implementados.

4.1. Nociones preliminares y notación

La aproximación DVS está basada en *discretizaciones sin traslape*, las cuales fueron introducidas durante el desarrollo de [12]. Una discretización es *sin traslape* cuando está basada en un sistema de nodos que es *sin traslape*; para distinguir los nodos de dicho sistema desde los *nodos originales*, estos son llamados *nodos derivados*.
Así, un sistema de nodos es sin traslape, con respecto a la malla gruesa (o descomposición de dominio), si cada uno de ellos pertenece a uno y solo uno de los subdominios. En general en el marco DVS, el espacio de vectores derivados (DVS) está constituido por todo el espacio lineal de funciones cuyo dominio es el total de los nodos derivados y que toma valores en $\mathbb{R}^n$.
En la presente tesis, donde son tratados los problemas de elasticidad que están gobernados por un sistema de tres EDP's, tomamos n=3. Usualmente, cuando el modelo matemático básico es gobernado por una sola ecuación diferencial, n es igual a 1.

Generalmente, cuando la malla gruesa es introducida algunos de los nodos de la malla fina caen en la cerradura de más de un subdominio de la malla gruesa. Cuando es el caso, un procedimiento general para transformar un conjunto de nodos con traslape en uno de nodos sin traslape se introdujo en [12]. Dicho procedimiento consiste en dividir cada nodo original en tantas piezas como subdominios haya. Para el caso en el cual la malla gruesa consiste de solamente cuatro subdominios, este proceso está ilustrado esquemáticamente en las siguientes figuras.

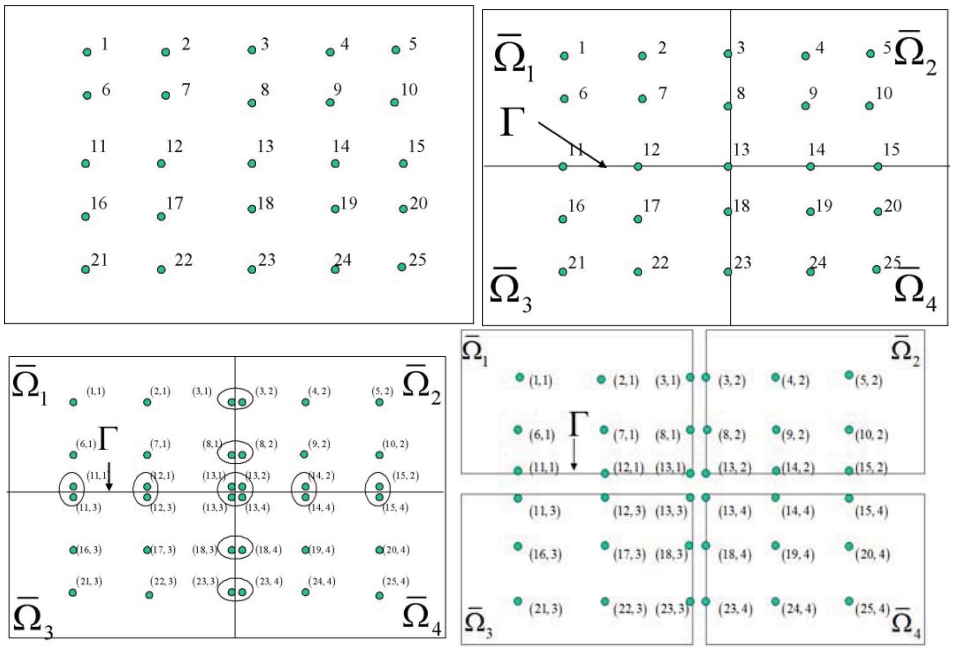

Figura 4.1: Nodos originales (arriba izquierda), Malla gruesa con nodos traslapados (arriba derecha), Generación de nodos sin traslape(abajo izquierda), Subdominios completamente sin traslape (abajo derecha)

Entonces, el resultado final es el sistema de nodos sin traslape mostrado en la figura 4. Cada uno de los nodos sin traslape está identificado de manera única por la pareja (p,α), donde p es el nodo original de donde proviene y α es el subdominio al cual pertenece. Usando esta notación, para cada $\beta = 1, ..., E$, es conveniente definir $\mathrm{X}^\beta \subset \mathrm{X}$ como sigue: Los nodos derivados (p,α) pertenecen a X^β, si y solo si, $\alpha = \beta$.
En lo que sigue, la familia de subconjuntos $\left\{\mathrm{X}^1, ..., \mathrm{X}^E\right\}$ sólo será referida como la *descomposición sin traslape del conjunto de nodos derivados*. Esto porque esta familia de subconjuntos de X posee la siguiente propiedad:

$$\mathrm{X} = \bigcup_{\alpha=1}^{E} \mathrm{X}^{\alpha} \; y \; \emptyset = \mathrm{X}^{\alpha} \cap \mathrm{X}^{\beta} \; cuando \; \alpha \neq \beta, \tag{4.1.1}$$

Una propiedad importante implicada por la ec. (4.1.1) es que el espacio de vectores derivados, , es la suma directa de la siguiente familia de subespacios de W: $\left\{W^1, ..., W^E\right\}$; esto es

$$W = W^1 \oplus ... \oplus W^E \tag{4.1.2}$$

Aquí, hemos escrito

$$W^{\alpha} \equiv W\left(\mathrm{X}^{\alpha}\right), \;\; \alpha = 1, ..., E \tag{4.1.3}$$

La notación $W\left(\mathrm{X}^{\alpha}\right)$, introducida previamente (ver, por ejemplo [12]), aquí es usada para representar el subespacio W cuyos vectores se desvanecen en cada *nodo derivado* que no pertenece a X^{α}. Una implicación importante, muy usada para el desarrollo de códigos en paralelo, es que cada vector derivado $\underline{w} \in W$ puede ser escrito de manera única en la forma

$$\underline{w} = \sum_{\alpha=1}^{E} \underline{w}^{\alpha}, \; con \; \underline{w}^{\alpha} \in W^{\alpha} \tag{4.1.4}$$

Como se acostumbra en los desarrollos de DDM, en la aproximación del DVS se hace una clasificación de los nodos de la discretización. Listamos los subconjuntos más usados y relevantes de X como sigue:

$$\begin{array}{l} \mathrm{I} \; nodos \; interiores \\ \Gamma \; nodos \; de \; interface \\ \pi \; nodos \; primales \\ \Delta \; nodos \; duales \\ \Pi \equiv \mathrm{I} \cup \pi \;\; nodos \; ' \; primales \; extendidos \; ' \\ \Sigma \equiv \mathrm{I} \cup \Delta \;\; nodos \; ' \; duales \; extendidos \; ' \end{array} \tag{4.1.5}$$

También observamos que cada una de las siguientes familias de conjuntos son disjuntos: $\{\mathrm{I}, \Gamma\}$, $\{\mathrm{I}, \pi, \Delta\}$, $\{\Pi, \Delta\}$ y $\{\Sigma, \pi\}$, mientras que

$$\mathrm{X} = \mathrm{I} \cup \Gamma = \mathrm{I} \cup \pi \cup \Delta = \Pi \cup \Delta = \Sigma \cup \pi \tag{4.1.6}$$

A continuación, resaltamos algunos de los aspectos más importantes de la notación y la nomenclatura usada en el marco del DVS; para mayores detalles el lector puede acudir a trabajos previos de esta línea de investigación (en particular [12], donde se dan referencias adicionales).

Cuando consideramos cualquier nodo derivado, el cual es identificado por la pareja de números(p,α), el número natural p (el cual corresponde a un nodo original) es llamado el ancestro del nodo derivado, mientras que α (cuyo rango va de 1 a E) identifica al subdominio al que pertenece. Además, para cada nodo

original $p \in \widehat{\mathrm{X}}$, la notación $\mathrm{Z}(p) \subset \mathrm{X}$ será usada para representar el conjunto de nodos derivados que provienen de él. Entonces, la multiplicidad de *p'*, $m(p)$, es la *cardinalidad* de $\mathrm{Z}(p)$. Observamos que la multiplicidad de p esta definida como una propiedad de cada nodo original p. Hay otro tipo de multiplicidad que es usada en el marco DVS, la cual está definida como una propiedad de cada pareja (p, q) de nodos originales y es también usada en la teoría del marco del DVS. Para introducirla, definimos

$$\delta_{pq}^{\alpha} \equiv \begin{cases} 1, \ si \ p, q \in \bar{\Omega}_{\alpha} \\ 0, \ de \ otra \ forma \end{cases}, \ \alpha = 1, ..., E; \ y \tag{4.1.7}$$

Entonces, la multiplicidad de una pareja (p, q) –escrita como $m(p, q)$ – está definida para que sea

$$m(p, q) \equiv \sum_{\alpha=1}^{E} \delta_{pq}^{\alpha} \tag{4.1.8}$$

Cuando $\underline{u}$ es un vector derivado, tal que $\underline{u}$ es una función definida en X, $\underline{u}(p,\alpha)$ se mantiene para los valores de $\underline{u}$ en el nodo derivado (p,α).
En particular, en aplicaciones del marco del DVS a problemas de elasticidad, esos valores son vectores en 3D y el número real $u(p, \alpha, i)$ -$i = 1, 2, 3$- será el $i - esimo$ componente del vector $\underline{u}(p,\alpha)$.
El espacio de vectores derivados está provisto con un producto interior, el producto interior euclidiano, el cual usando la notación de arriba para cada pareja de vectores derivados, $\underline{u}$ y $\underline{w}$, está definido por

$$\underline{u} \bullet \underline{w} \equiv \sum_{(p,\alpha)\in \mathrm{X}} \sum_{i=1}^{3} u(p, \alpha, i)\, w(p, \alpha, i) = \sum_{\alpha=1}^{E} \sum_{(p,\alpha)\in \mathrm{X}^{\alpha}} \sum_{i=1}^{3} u(p, \alpha, i)\, w(p, \alpha, i) \tag{4.1.9}$$

Para la paralelización de los algoritmos la relación

$$\underline{u} \bullet \underline{w} = \sum_{\alpha=1}^{E} \underline{u}^{\alpha} \bullet \underline{w}^{\alpha}, \ siempre \ y \ cuando \ \underline{u}^{\alpha}, \underline{w}^{\alpha} \in W^{\alpha} \tag{4.1.10}$$

será muy útil, porque las componentes del vector correspondiente a los diferentes subdominios serán manejados por diferentes procesadores en la implementación computacional.
Sea $p \in \widehat{\mathrm{X}}$ un nodo original y $\underline{u} \in W$ un vector derivado. Entonces $\underline{u} \in W$ se dice que es continuo en p cuando $\underline{u}(p,\alpha)$ es independiente de α, y se dice que es de promedio cero en cuando

$$\sum_{\alpha\in \mathrm{Z}(p)} \underline{u}(p,\alpha) = 0 \tag{4.1.11}$$

Cuando las correspondientes propiedades son satisfechas para cada $p \in \widehat{\mathrm{X}}$, el vector derivado $\underline{u}$ simplemente se dice que es continuo o de promedio cero.

Los subespacios lineales W_{12} y W_{11} de W, están constituidos por los vectores continuos y de promedio cero de W, respectivamente. Estos dos subespacios son complementos ortogonales uno del otro. Las matrices $\underline{\underline{a}}$ y $\underline{\underline{j}}$ son las proyecciones en W_{12} y en W_{11}, respectivamente. Estas matrices satisfacen:

$$\underline{\underline{a}} + \underline{\underline{j}} = \underline{\underline{\mathrm{I}}} \tag{4.1.12}$$

donde $\underline{\underline{\mathrm{I}}}$ es la matriz identidad. Para cualquier $\underline{w} \in W$, la evaluación de $\underline{v} \equiv \underline{\underline{a}}\,\underline{w}$ esta dada por

$$\underline{v}\,(p,\alpha\,) \equiv \frac{1}{m\,(p)} \sum_{(p,\beta)\in \mathrm{Z}(p)} \underline{w}\,(p,\beta\,) \tag{4.1.13}$$

Usando esta ecuación, la evaluación de $\underline{\underline{j}}\,\underline{w}$ es también de manera directa, dada la ec. (4.1.13) que implica

$$\underline{\underline{j}}\,\underline{w} = \underline{w} - \underline{\underline{a}}\,\underline{w} \tag{4.1.14}$$

La inyección natural de $\widehat{W}$ en W, escrita como $R : \widehat{W} \to W$, está definida para cada $\underline{\widehat{u}} \in \widehat{W}$ por $\underline{\widehat{u}} \in \widehat{W}$

$$\left(R\underline{\widehat{u}}\right)(p,\alpha\,) = \underline{\widehat{u}}\,(p)\,,\ \forall\,(p,\alpha\,) \in \mathrm{X} \tag{4.1.15}$$

Cuando $\underline{\widehat{u}} \in \widehat{W}$, $\left(R\underline{\widehat{u}}\right) \in W_{12}$ necesariamente. Observamos que $R\widehat{W} = W_{12}$. Además, se puede ver que R tiene inversa única en W_{12}; esto es, $R^{-1} : W_{12} \to \widehat{W}$ está bien definida.

4.1.1. Cómo construir discretizaciones sin traslape

Esta sección explica cómo transformar una discretización estandar (con traslape) en una discretización sin traslape. El procedimiento DVS aquí explicado permite transformar una discretización con traslape en discretizaciones sin traslape y genera directamente algoritmos precondicionados que están sujetos a restricciones. Puede ser aplicado siempre y cuando la siguiente suposición básica se satisfaga:

$$m\,(p,q) = 0 \Rightarrow M_{pq} = 0 \tag{4.1.16}$$

Aquí, el símbolo se mantiene para la implicación lógica y se entiende que es la matriz que aparece en la ec.(4.1.16).
Definimos la matriz $\underline{\underline{a}}'$ por su acción sobre cualquier vector de W: cuando $\underline{u} \in W$, tenemos

$$\underline{\underline{a}}'\,\underline{u} = \underline{u}_{\mathrm{I}} + \underline{u}_{\Delta} + \underline{\underline{a}}\,\underline{u}_{\pi} \tag{4.1.17}$$

Observamos que la acción de $\underline{\underline{a}}'$ puede ser llevada a cabo por la aplicación del operador $\underline{\underline{a}}'$ exclusivamente en los nodos primales. Entonces, definimos *el espacio restringido* como

$$W^r \equiv \underline{\underline{a}}' W \tag{4.1.18}$$

Claramente, $W^r \subset W$ es un espacio lineal de W y para cualquier $\underline{u} \in W$, $\underline{\underline{a}}' \underline{u}$ es la proyección de $\underline{u}$, en W^r.
Ahora, definimos

$$s(p,q) \equiv \begin{cases} 1, \ cuando \ m(p,q) = 0 \\ m(p,q), \ cuando \ m(p,q) \neq 0 \end{cases} \tag{4.1.19}$$

Para $\gamma = 1, ..., E$, definimos las matrices

$$\underline{\underline{M}}^{\gamma} \equiv \left(M_{pq}^{\gamma}\right) \ con \ M_{pq}^{\gamma} \equiv \frac{M_{pq}}{s(p,q)} \delta_{pq}^{\gamma} \tag{4.1.20}$$

Después, definimos las matrices:

$$\underline{\underline{A}}^{\gamma} \equiv \left(A_{(p,\alpha)(q,\beta)}^{\gamma}\right) \ con \ A_{(p,\alpha)(q,\beta)}^{\gamma} \equiv M_{pq}^{\gamma} \delta_{(\alpha,\gamma)} \delta_{(\beta,\gamma)} \tag{4.1.21}$$

y

$$\underline{\underline{A}}^{t} \equiv \sum_{\gamma=1}^{E} \underline{\underline{A}}^{\gamma} \tag{4.1.22}$$

Entonces, definimos

$$\underline{\underline{A}} \equiv \underline{\underline{a}}' \underline{\underline{A}}^{t} \underline{\underline{a}}' \tag{4.1.23}$$

El siguiente resultado fue presentado en trabajos previos [12]:

Teorema 1 *Sean $\underline{U} \in \widehat{W}$ y $\underline{u} \in W$ relacionados por $\underline{u} = R\underline{U}$, mientras $\underline{f} \in W_{12}$ está definida como*

$$\underline{f} \equiv R\left(\underline{\underline{\widehat{m}}}^{-1} \underline{F}\right) \tag{4.1.24}$$

Aquí, $\underline{\underline{\widehat{m}}}$ es una matriz diagonal que transforma $\widehat{W}$ en sí mismo, cuyos valores diagonales son $m(p)$, mientras aquí su inversa se denota como $\underline{\underline{\widehat{m}}}^{-1}$, entonces, la versión discretizada de la elasticidad estática de la ec. (4.1.48):

$$\underline{\underline{M}} \underline{U} = \underline{F} \tag{4.1.25}$$

se satisface, si y sólo si

$$\underline{\underline{aA}} \underline{u} = \underline{f} \ y \ \underline{\underline{j}} \underline{u} = 0 \tag{4.1.26}$$

Prueba.- Ver para un ejemplo [12].

4.1.2. Los Algoritmos DVS Precondicionados con restricciones

Hay cuatro algoritmos [12], y dos de ellos son los DVS-BDDC y el DVS-FETI-DP. Estas son las versiones DVS de los bien conocidos BDDC [5]-[7]] y FETI-DP [8]-[11]. Mientras que para los otros dos, nada similar ha sido reportado en la literatura previo a la publicación de los algoritmos DVS. Por ahora, es bien conocido que están muy estrechamente relacionados y lo mismo se puede decir de todo el grupo de los cuatro algoritmos DVS.
El complemento de Schur para DVS esta definido como

$$\underline{\underline{S}} \equiv \underline{\underline{A}}_{\Delta\Delta} - \underline{\underline{A}}_{\Delta\Pi} \left(\underline{\underline{A}}_{\Pi\Pi} \right)^{\sim 1} \underline{\underline{A}}_{\Pi\Delta} \tag{4.1.27}$$

También define

$$\underline{\bar{f}}_{\Delta} \equiv \underline{f}_{\Delta} - \underline{\underline{A}}_{\Delta\Pi} \left(\underline{\underline{A}}_{\Pi\Pi} \right)^{\sim 1} \underline{f}_{\Pi} \tag{4.1.28}$$

Entonces, escribiendo $\underline{u} \equiv \underline{u}_{\Pi} + \underline{u}_{\Delta}$ha sido mostrado [12] que la ec. (4.1.27) se satisface si y solo si

$$\underline{\underline{aS}}\underline{u}_{\Delta} = \underline{\bar{f}}_{\Delta}, \quad \underline{\underline{j}}\underline{u}_{\Delta} = 0 \tag{4.1.29}$$

y

$$\underline{u}_{\Pi} = \left(\underline{\underline{A}}_{\Pi\Pi} \right)^{\sim 1} \left(\underline{f}_{\Pi} - \underline{\underline{A}}_{\Pi\Delta} \underline{u}_{\Delta} \right) \tag{4.1.30}$$

La estrategia seguida en la aproximación del DVS, es encontrar $\underline{u}_{\Delta} \in W(\Delta)$primero y después aplica ec. (4.1.29) para obtener la parte remanente, $\underline{u}_{\Pi} \in W(\Pi)$, de $\underline{u}$.
Para que esta estrategia sea efectiva es esencial que la aplicación de $\left(\underline{\underline{A}}_{\Pi\Pi} \right)^{\sim 1}$sea computacionalmente económica. Diferentes algoritmos se derivan al buscar diferentes partes de información tal que $\underline{u}_{\Delta} \in W(\Delta)$ pueda ser derivada de ella de una forma computacionalmente económica.
En particular, los cuatro algoritmos DVS mencionados hacen la búsqueda para: $\underline{u}_{\Delta}$, $\underline{\underline{j}}\underline{\underline{S}}\underline{u}_{\Delta}$, $\underline{\underline{S}}^{\sim 1}\underline{\underline{j}}\underline{\underline{S}}\underline{u}_{\Delta}$ y $\underline{\underline{S}}\underline{u}_{\Delta}$, respectivamente. Descritos en [12], son listados a continuación.

4.1.3. El Algoritmo DVS-BDDC

Este algoritmo busca para $\underline{u}_{\Delta}$. Esto es

$$\underline{\underline{aS}}^{-1}\underline{\underline{aS}}\underline{u}_{\Delta} = \underline{\underline{aS}}^{-1}\underline{f}_{\Delta} \;\; and \; \underline{\underline{j}}\underline{u}_{\Delta} = 0 \tag{4.1.31}$$

4.1.4. El Algoritmo DVS-Primal

Establecemos $\underline{v}_\Delta \equiv \underline{\underline{S}}^{\sim 1}\underline{\underline{j}}\underline{\underline{S}}\underline{u}_\Delta$y el algoritmo consiste en la búsqueda para una función $\underline{v}_\Delta \in W_\Delta$, la cual satisface

$$\underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{v}_\Delta = \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \;\; y \; \underline{\underline{a}}\underline{\underline{S}}\underline{v}_\Delta = 0 \tag{4.1.32}$$

Una vez que $\underline{v}_\Delta \in W\left(\Delta\right)$ha sido obtenida, entonces

$$\underline{u}_\Delta = \underline{\underline{a}}\left(\underline{\underline{S}}^{-1}\underline{f}_\Delta + \underline{v}_\Delta\right) \tag{4.1.33}$$

4.1.5. El Algoritmo DVS-FETI-DP

Este algoritmo busca a $\underline{\lambda} \equiv \underline{\underline{j}}\underline{\underline{S}}\underline{u}_\Delta$. Por lo tanto, el algoritmo es "Dado $\underline{f}_\Delta \in \underline{\underline{a}}W_\Delta$, encontrar $\underline{\lambda} \in W_\Delta$tal que

$\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{\sim 1}\underline{\lambda} = -\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{\sim 1}\underline{f}_\Delta \;\; y \; \underline{\underline{a}}\underline{\lambda} = 0$ "

Una vez que $\underline{\lambda} \in W_\Delta$ha sido obtenida, $\underline{u}_\Delta \in \underline{\underline{a}}W_\Delta$ está dada como

$$\underline{u}_\Delta = \underline{\underline{a}}\underline{\underline{S}}^{\sim 1}\left(\underline{f}_\Delta + \underline{\lambda}\right) \tag{4.1.34}$$

4.1.6. El Algoritmo DVS-Dual

En este caso uno busca $\underline{\mu} \equiv \underline{\underline{S}}\underline{u}_\Delta$ usando la relación:

$$\underline{\underline{S}}\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\mu} = \underline{\underline{S}}\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \;\; y \; \underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\mu} = 0 \tag{4.1.35}$$

Una vez que $\underline{\mu} \in W\left(\Delta\right)$ ha sido obtenida, $\underline{u}_\Delta \in W\left(\Delta\right)$ está dada como:

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

4.1.7. Algoritmos DVS aplicados a elasticidad

Para implementar estos métodos en el sistema lineal algebraico que proviene de un sistema de ecuaciones diferenciales parciales, en este caso de elasticidad se requiere extender algunos conceptos y adaptarlos a funciones valuadas vectorialmente. Esos conceptos son básicamente: el producto interior así como el de los operadores de salto y de promedio para matrices. También es necesario extender el tamaño de los vectores pues el número de componentes se incrementa en un factor de 3.

Al igual que en el caso de una sola ecuación diferencial los conjuntos de nodos se definen de la misma forma dado que el dominio no se modifica en los sistemas de EDP's. Donde ocurre un cambio es en los espacios vectoriales definidos sobre esos conjuntos de nodos, que aunque están definidos de la misma forma que para el caso anterior ahora se incrementa el número de componentes en las funciones de esos espacios. Los espacios vectoriales de dimensión finita W y $\widehat{W}$ se definen

a continuación; sus miembros son las funciones en tres dimensiones valuadas vectorialmente definidas en Ωy X, respectivamente. Claramente, la dimensión de $\widehat{W}$ es $3N$, mientras que de W es generalmente mayor que $3N$.
Cualquier función $\widehat{\underline{u}} \in W$ y $\underline{u} \in W$ cuando es evaluada en los nodos $p \in \Omega$ y $(p,\alpha) \in \bar{\Omega}$, respectivamente, resultan en vectores de tres dimensiones a los que se les denota como $\widehat{\underline{u}}(p)$ y $\underline{u}(p,\alpha)$, respectivamente; además, usaremos la notación $\widehat{u}(p,i)$ y $u(p,i,\alpha)$ para el $i-$ésimo componente de dichos vectores, respectivamente. Una función $\underline{u} \in W$ es *continua*, cuando $\underline{u}(p,\alpha)$ es independiente de α. El subespacio $W_{12}\left(\bar{\Omega}\right) \subset W$ esta conformado de las *funciones continuas* de W, mientras que la biyección $R : \widehat{W} \to W$ es la *imersión natural* de $\widehat{W}$ en W; que esta definida como:

$$\left(R\widehat{\underline{u}}\right)(p,\alpha) \equiv \widehat{\underline{u}}(p),\ \forall (p,\alpha) \in X \tag{4.1.36}$$

El producto interior Euclideano en W se define como:

$$\widehat{\underline{u}} \bullet \widehat{\underline{w}} \equiv \sum_{p=1}^{N}\sum_{i=1}^{3} \widehat{u}(p,i)\,\widehat{w}(p,i),\ \forall \widehat{\underline{u}}, \widehat{\underline{w}} \in W \tag{4.1.37}$$

Mientras que el producto interior Euclideano para funciones valuadas vectorialmente en W es:

$$\underline{u} \bullet \underline{w} \equiv \sum_{\alpha \in Z(p)}\sum_{p=1}^{N}\sum_{i=1}^{3} u(p,i,\alpha)\,w(p,i,\alpha),\ \forall \underline{u}, \underline{w} \in W \tag{4.1.38}$$

La *matriz de promedios*, $\underline{\underline{a}}$, es la proyección ortogonal,con respecto al producto interior Euclideano, en los subespacios W_{12} de vectores continuos. Su expresión explícita es

$$a_{(p,i,\alpha)(q,j,\beta)} = \frac{1}{m(p)}\delta_{pq}\delta_{ij} \tag{4.1.39}$$

mientras que la *matriz de salto*, es $\underline{\underline{j}} \equiv \underline{\underline{I}} - \underline{\underline{a}}$. Aquí, $\underline{\underline{I}}$ es la matriz identidad.

La matriz original $\underline{\underline{A}}$ se escribe como

$$\widehat{\underline{\underline{A}}} = \left(\underset{pqij}{\widehat{A}}\right) \tag{4.1.40}$$

para cada pareja (p,q) tal que $p \in \widehat{X}$ and $q \in \hat{X}$ se define

$$\delta_{pq}^{\alpha} \equiv \begin{pmatrix} 1 & \text{if} \quad p,q \in X_\alpha \\ 0 & \text{if} \quad p \notin X_\alpha \text{o}\ p \notin X_\alpha \end{pmatrix} \quad \alpha = 1,\cdots,E \tag{4.1.41}$$

Mientras que la multiplicidad de la pareja (p,q) se escribe como $m(p,q)$ y está definida como

$$m(p,q) \equiv \sum_{\alpha=1}^{E} \delta_{pq}^{\alpha} \tag{4.1.42}$$

Con lo que se define la matriz en cada bloque γ

$$A^{\gamma}_{(q,j,\beta)(p,i,\alpha)} \equiv \delta_{\alpha\gamma}\delta_{\beta\gamma}\int_{\substack{\Omega\\ \gamma}}\left\{(\lambda+\mu)\frac{\partial \ell_p}{\partial x_i}\frac{\partial \ell_q}{\partial x_j}+\mu\sum_{r=1}^{3}\frac{\partial \ell_p}{\partial x_r}\frac{\partial \ell_q}{\partial x_r}\delta_{ij}\right\}d\underline{x} \quad (4.1.43)$$

Además, $\underline{\underline{A}}^t : W \to W$ esta definida como $\underline{\underline{A}}^t \equiv \sum_{\gamma=1}^{E}\underline{\underline{A}}^{\gamma}$.
Considerando la siguiente transformación

$$W^r \equiv \underline{\underline{a}}' W \quad (4.1.44)$$

Se va a construir la matriz

$$\underline{\underline{A}} : W^r \to W^r \quad (4.1.45)$$

Entonces, la matriz $\underline{\underline{A}}$ esta definida por $\underline{\underline{A}} \equiv \underline{\underline{a}}'\,\underline{\underline{A}}\,\underline{\underline{a}}'$.
Siguiendo los pasos descritos sucintamente en la sección (4.1.1), se construyó un software que constituye una herramienta para aplicar efectivamente hardware masivamente paralelo a problemas de solidos elásticos isotrópicos en equilibrio. El software que hemos desarrollado trata en paralelo el sistema discreto de ecuaciones lineales que se obtuvo cuando el *método de discretización estándar* usado para obtener la *discretización original* del problema con condiciones de frontera (BVP) Dirichlet definido por las ecuaciones y (4.1.47) es el de Elemento Finito (FEM). En particular, este fue obtenido aplicando el bien conocido principio variacional:

$$\int_{\Omega}\left\{(\lambda+\mu)\left(\nabla\bullet\underline{u}\right)\left(\nabla\bullet\underline{w}\right)+\mu\nabla\underline{u}:\nabla\underline{w}\right\}dx=\int_{\Omega}\underline{f}_{\Omega}\bullet\underline{w}dx \quad (4.1.46)$$

con funciones lineales.
Sujeto a condiciones de frontera tipo Dirichlet

$$\underline{u}=0,\ \text{on}\ \partial\Omega \quad (4.1.47)$$

Aunque con unas simples modificaciones al código se pueden establecer también otras condiciones de frontera.
Dicho sistema de ecuaciones puede ser escrito como

$$\underline{\underline{M}}\,\underline{U}=\underline{F} \quad (4.1.48)$$

Aquí, se entienden los vectores $\underline{U}$ y $\underline{F}$, como funciones definidas en todo el conjunto de *nodos originales* de la malla usada en la discretización de FEM, cuyos valores en cada nodo son vectores $3-D$. Estos pueden ser escritos como $\underline{U}\equiv\left(\underline{U}_p\right)\equiv(U_{pi})$ y $\underline{F}\equiv\left(\underline{F}_p\right)\equiv(F_{pi})$. Como para la matriz $\underline{\underline{M}}$, la notación

$$\underline{\underline{M}}\equiv\left(\underline{\underline{M}}_{pq}\right)\equiv(M_{piqj}) \quad (4.1.49)$$

es adoptada. Arriba, el rango de p *y* q es todo el conjunto de *nodos originales*, mientras i y j pueden tomar cualquiera de los valores $1,2,3$.

Capítulo 5

Visión General del Software DVS

5.1. Las piezas elementales del software DVS

Todos los algoritmos DVS son algoritmos iterativos y pueden ser implementados con el recurso del Método del Gradiente Conjugado (CGM por sus siglas en inglés), cuando la matriz es definida y simétrica, como lo es en el caso de los problemas de elasticidad aquí considerados, o con algún otro procedimiento iterativo como sería el GMRES, cuando ese no es el caso. En cada paso de la iteración, dependiendo del algoritmo DVS que sea aplicado, uno tiene que computar la acción sobre un vector derivado arbitrario de uno de las siguientes matrices: $\underline{\underline{aS}}^{\sim 1}\underline{\underline{aS}}$, $\underline{\underline{jSj}}\underline{\underline{S}}^{\sim 1}$, $\underline{\underline{S}}^{\sim 1}\underline{\underline{jSj}}$ o $\underline{\underline{SaS}}^{\sim 1}\underline{\underline{a}}$. De esta manera, dichas matrices son permutaciones distintas de $\underline{\underline{S}}$, $\underline{\underline{S}}^{\sim 1}$, $\underline{\underline{a}}$ y $\underline{\underline{j}}$. Por lo tanto, un código que implemente cualquiera de los algoritmos DVS puede ser fácilmente desarrollado cuando los códigos llevan a cabo la acción de cada una de las matrices que están ya disponibles.

Producir tales códigos será el objetivo de la siguiente sección, mientras que el resto de esta misma estará dedicada a obtener algunos resultados auxiliares que serán usados allí y fueron previamente presentados en [12]. El primero de dichos resultados es:

$$\underline{\underline{S}} = \underline{\underline{A}}^t_{\Delta\Delta} - \underline{\underline{A}}^t_{\Delta\Pi}\left(\underline{\underline{a}}'\underline{\underline{A}}^t_{\Pi\Pi}\underline{\underline{a}}'\right)^{\sim 1}\underline{\underline{a}}'\underline{\underline{A}}^t_{\Pi\Delta} \tag{5.1.1}$$

El segundo es: Cuando $\underline{w} \in W$, la siguiente identidad se mantiene

$$\underline{\underline{S}}^{\sim 1}\underline{w} = \left(\underline{\underline{A}}^{\sim 1}\underline{w}_\Delta\right)_\Delta \tag{5.1.2}$$

Aquí la notación $\left(\underline{\underline{A}}^{\sim 1}\underline{w}_\Delta\right)_\Delta$se mantiene para los componentes en $W(\Delta)$de $\underline{\underline{A}}^{\sim 1}\underline{w}_\Delta$.

El tercero y cuarto resultados requirieron referirse a las pseudo inversas que aparecen en las ecuaciones (5.1.1) y (5.1.2). Estos son

1. Sea $\underline{w} \in W^r(\Pi)$ y $\underline{v} \equiv \left(\underline{\underline{A}}_{\Pi\Pi}\right)^{\sim 1} \underline{w}$, entonces

$$\underline{\underline{a}}' \left(\underline{\underline{A}}^t_{\pi\pi} - \underline{\underline{A}}^t_{\pi \mathrm{I}} \left(\underline{\underline{A}}^t_{\mathrm{II}}\right)^{\sim 1} \underline{\underline{A}}^t_{\mathrm{I}\pi}\right) \underline{v}_\pi = \underline{w}_\pi - \underline{\underline{A}}^t_{\pi \mathrm{I}} \left(\underline{\underline{A}}^t_{\mathrm{II}}\right)^{\sim 1} \underline{w}_{\mathrm{I}},\ y\ \underline{\underline{j}}\underline{v}_\pi = 0 \tag{5.1.3}$$

junto con

$$\underline{v}_{\mathrm{I}} = \left(\underline{\underline{A}}^t_{\mathrm{II}}\right)^{\sim 1} \left(\underline{w}_{\mathrm{I}} - \underline{\underline{A}}^t_{\mathrm{I}\pi} \underline{v}_\pi\right) \tag{5.1.4}$$

1. Sea $\underline{w} \in W^r$ y $\underline{v} \equiv \underline{\underline{A}}^{\sim 1} \underline{w}$, entonces

$$\underline{\underline{a}}' \left(\underline{\underline{A}}^t_{\pi\pi} - \underline{\underline{A}}^t_{\pi\Sigma} \left(\underline{\underline{A}}^t_{\Sigma\Sigma}\right)^{\sim 1} \underline{\underline{A}}^t_{\Sigma\pi}\right) \underline{v}_\pi = \underline{w}_\pi - \underline{\underline{A}}^t_{\pi\Sigma} \left(\underline{\underline{A}}^t_{\Sigma\Sigma}\right)^{\sim 1} \underline{w}_\Sigma,\ and\ \underline{\underline{j}}\underline{v}_\pi = 0 \tag{5.1.5}$$

junto con

$$\underline{v}_\Sigma = \left(\underline{\underline{A}}^t_{\Sigma\Sigma}\right)^{\sim 1} \left(\underline{w}_\Sigma - \underline{\underline{A}}^t_{\Sigma\pi} \underline{v}_\pi\right) \tag{5.1.6}$$

Estos resultados permiten aplicar algoritmos iterativos, en los cuales el CGM es usado, cuando las aplicaciones de $\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{\sim 1}$ y $\underline{\underline{A}}^{\sim 1}$, respectivamente son computadas.

5.2. Construcción del software DVS

Todos los algoritmos DVS presentados en la sección 6 son iterativos, como es el caso con la mayoría de los algoritmos DDM, y para implementarlos es necesario solamente desarrollar códigos paralelizados capaces de calcular la acción de cada una de las matrices $\underline{\underline{S}}$, $\underline{\underline{S}}^{\sim 1}$, $\underline{\underline{a}}$ o $\underline{\underline{j}}$ a un vector derivado arbitrario, como fue previsto en [12].
En el código aquí reportado, todas las soluciones de los sistemas de ecuaciones que no eran locales fueron obtenidos con la ayuda del algoritmo del CGM. Debido a esta circunstancia, de hecho los siguientes subprogramas fueron requeridos: $\underline{\underline{S}}$, $\underline{\underline{S}}^{\sim 1}$ y $\left(\underline{\underline{a}}\underline{\underline{S}}^{\sim 1}\underline{\underline{a}}\underline{\underline{S}}\right)^{-1}$. Además de la aplicación de

$$\underline{\underline{S}} = \underline{\underline{A}}^t_{\Delta\Delta} - \underline{\underline{A}}^t_{\Delta\Pi} \left(\underline{\underline{a}}' \underline{\underline{A}}^t_{\Pi\Pi} \underline{\underline{a}}'\right)^{\sim 1} \underline{\underline{a}}' \underline{\underline{A}}^t_{\Pi\Delta} \tag{5.2.1}$$

Requiere el cómputo de la acción de $\left(\underline{\underline{a}}' \underline{\underline{A}}^t_{\Pi\Pi} \underline{\underline{a}}'\right)^{-1}$ la cual no es local. Por lo tanto, un subprograma eficiente para llevar a cabo esta operación eficientemente en paralelo fue requerido y fue desarrollado. Las comunicaciones requeridas por los

algoritmos DVS son muy fáciles de analizar. Realmente, cuando un procesador diferente es apartado para cada uno de los subdominios de la malla gruesa (esto es, para los subconjuntos de *nodos derivados*, X^α, $\alpha = 1, ..., E$, de la *partición sin traslape de* X) –como fue hecho en el trabajo aquí reportado– la transmisión de información entre diferentes procesadores ocurre solamente cuando el *producto interior global Euclideano* - es computado, ya sea que la matriz $\underline{\underline{a}}$ o la matriz $\underline{\underline{a}}'$ sea aplicada. Además, en esas operaciones la cantidad de información transmitida es muy pequeña.
En una primera versión tentativa del software, también se usó un procesador maestro. Sin embargo, usando tal procesador como el centro de las comunicaciones resultó más costoso en tiempo de procesamiento que cuando el procesador maestro no fue usado como centro de comunicaciones, el trabajo hecho por este es tan pequeño que puede ser eliminado fácilmente. Cuando esto se hace, el desempeño del algoritmo DVS se vuelve extremadamente bueno como se explica y discute en la sección 10.
Solamente el algoritmo DVS-BDDC fue implementado para este trabajo. A pesar de ello la implementación de los otros tres algoritmos es muy similar, y la expectativa de su eficiencia en paralelo también, su implementación habría tomado tiempo adicional y esfuerzo que preferimos guardar para trabajos en el futuro.

5.3. Construcción del software local del DVS

Una propiedad fundamental de $\underline{\underline{A}}^t$, como se definió en la ec. (4.1.22) es que es diagonal por bloques, en la cual cada uno de los bloques es $\underline{\underline{A}}^\alpha$, para cada $\alpha = 1, ..., E$, es una transformación lineal de W^αen sí mismo. Esta propiedad simplifica mucho la paralelización de los códigos para la implementación de los algoritmos DVS.
Para este fin, cada uno de los subconjuntos X^α de la *decomposición sin traslape de* X, es asignada a un procesador diferente y el conjunto de procesadores es numerado de acuerdo con ello. El hecho de que cada vector $\underline{w} \in W$ pueda ser escrito de manera única como

$$\underline{w} = \sum_{\alpha=1}^{E} \underline{w}_\alpha, \ \textit{con } \underline{w}_\alpha \in W^\alpha \tag{5.3.1}$$

se usa para este propósito. El procesador γ maneja solamente la componente $\underline{w}_\gamma$ de cada vector $\underline{w} \in W$. Entonces, todas las operaciones del procesador γ transforman $\underline{w}_\gamma$ en un vector que también pertenece a W^γ; aún los operadores $\underline{\underline{a}}$ y $\underline{\underline{a}}'$ transforman $\underline{w}_\gamma$ en un vector de W^γ, excepto que $\underline{\underline{a}}$ y $\underline{\underline{a}}'$ requieren información desde unos pocos procesadores vecinos. Sin embargo, es importante asegurar que dicha información sea actualizada en el momento y de manera conjunta.
Cuando se evalúa la acción sobre un vector de cualquiera de las matrices consideradas, el procesador γ será el responsable de la construcción de la componente γ de dicho vector; en particular, $\left(\underline{\underline{S}}\underline{w}\right)_\gamma$, $\left(\underline{\underline{S}}^{\sim 1}\underline{w}\right)_\gamma$, $\left(\underline{\underline{a}}\underline{w}\right)_\gamma$ o $\left(\underline{\underline{j}}\underline{w}\right)_\gamma$, dependiendo

de la matriz que este siendo aplicada. En lo que sigue se asume que, desde el inicio, los nodos del conjunto X^{γ} han sido clasificados en I: *interiores,* π: *primales,* y Δ :*duales*. Otras *clases de nodos* de X^{γ} que serán considerados son: Π: *primales extendidos,* y Σ :*duales extendidos*. Sin ningún otro aspecto a hacer notar, la siguiente relación también será usada:

$$W^{\gamma} \equiv W^{\gamma}(\mathrm{I}) \oplus W^{\gamma}(\pi) \oplus W^{\gamma}(\Delta) = W^{\gamma}(\Pi) \oplus W^{\gamma}(\Delta) = W^{\gamma}(\Sigma) \oplus W^{\gamma}(\pi) \tag{5.3.2}$$

5.3.1. La aplicación de $\underline{\underline{a}}$

, $\underline{\underline{a}}'$ y $\underline{\underline{j}}$

Para comenzar, evaluamos $\left(\underline{\underline{a}}\underline{w}\right)_{\gamma}$ cuando $\underline{w} \in W^{\gamma}$. Como se verá, la aplicación de $\underline{\underline{a}}$ a cualquier vector de W^{γ}requiere intercambio de información entre el procesador γ y los otros procesadores.
Esto es, recordando ec.

$$\left(\underline{\underline{a}}\underline{w}\right)_{\gamma} = \underline{\underline{a}}\underline{w}\,(p,\gamma\,) = \frac{1}{m\,(p)} \sum_{(p,\beta)\in \mathrm{Z}(p)} \underline{w}\,(p,\beta\,) \tag{5.3.3}$$

Por lo tanto, esta operación requiere información desde los procesadores que poseen nodos derivados que pertenecen a $\mathrm{Z}\,(p)$; por lo tanto, este cómputo involucra comunicaciones entre diferentes procesadores, lo cual puede hacer lento el procesamiento.
En vista de la ec.(5.3.3), es claro que exceptuando este intercambio de información, la evaluación de $\left(\underline{\underline{a}}\underline{w}\right)_{\gamma}$ es muy sencilla. Una vez que $\underline{\underline{a}}\underline{w}$ ha sido obtenido, la relación $\underline{\underline{j}}\underline{w} = \underline{w} - \underline{\underline{a}}\underline{w}$ puede ser usada para computar la acción de $\underline{\underline{j}}$. Como para la acción de $\underline{\underline{a}}'$, recordemos que $\underline{\underline{a}}'$ es obtenida cuando la aplicación de $\underline{\underline{a}}$ esta restringida a los *nodos primales*.
Antes de seguir adelante, son necesarios algunos comentarios finales. La aplicación de $\underline{\underline{a}}$, y por lo tanto la de $\underline{\underline{a}}'$, también requieren transmisión de información entre los procesadores. Por lo tanto, para un mejoramiento de la eficiencia de los códigos es esencial que la aplicación de los procedimientos sean diseñados con mucho cuidado. Como se ha visto, con unas pocas excepciones, todos los intercambios de información requerida, cuando los algoritmos DVS son implementados, es cuando las transformaciones $\underline{\underline{a}}$ y $\underline{\underline{a}}'$ son aplicadas.

5.3.2. El software DVS para $\underline{\underline{S}}$ y $\underline{\underline{S}}^{\sim \mathbf{1}}$

Debe observarse que en vista de la definición de la matriz y las submatrices que aparecen en la siguiente descomposición:

$$\underline{\underline{A}}^{\gamma} = \begin{pmatrix} \underline{\underline{A}}^{\gamma}_{\mathrm{II}} & \underline{\underline{A}}^{\gamma}_{\mathrm{I}\pi} & \underline{\underline{A}}^{\gamma}_{\mathrm{I}\Delta} \\ \underline{\underline{A}}^{\gamma}_{\pi\mathrm{I}} & \underline{\underline{A}}^{\gamma}_{\pi\pi} & \underline{\underline{A}}^{\gamma}_{\pi\Delta} \\ \underline{\underline{A}}^{\gamma}_{\Delta\mathrm{I}} & \underline{\underline{A}}^{\gamma}_{\Delta\pi} & \underline{\underline{A}}^{\gamma}_{\Delta\Delta} \end{pmatrix} \tag{5.3.4}$$

para cualquier γ esta transforma vectores de $W\left(\mathrm{X}^{\gamma}\right)$en vectores de $W\left(\mathrm{X}^{\gamma}\right)$. Por lo tanto, la *matriz local* está definida como

$$\underline{\underline{Q}} \equiv \underline{\underline{A}}^{\gamma} \tag{5.3.5}$$

donde $\underline{\underline{A}}^{\gamma}$es la matriz definida en la sección 5, por la ecuación (4.1.21). Es esta ecuación el índice γ se omite en la definición de $\underline{\underline{Q}}$, porque γ se mantiene fijo. Debido a los comentarios que ya han sido hechos, es claro que $\underline{\underline{Q}}$ es una transformación lineal bien definida de en sí misma. En particular, cuando $\underline{w}^{\gamma} \in W^{\gamma}$, la aplicación de $\left(\underline{\underline{Q}}\underline{w}^{\gamma}\right)_{\gamma}$
puede ser llevada a cabo de manera autónoma , en el procesador γ, sin intercambio de información con los otros procesadores. Esta es una diferencia fundamental con $\underline{a}$, $\underline{a}'$ y $\underline{j}$e implica que en cada procesadores la matriz sea construida o que un software interno capaz de evaluar su acción sobre cualquier vector de W^{γ}este disponible.
En vista de la ec.(5.3.4), la matriz $\underline{\underline{Q}}$ será escrita de dos maneras

$$\underline{\underline{Q}} = \begin{pmatrix} \begin{pmatrix} \underline{\underline{Q}}_{\mathrm{II}} & \underline{\underline{Q}}_{\mathrm{I}\pi} \\ \underline{\underline{Q}}_{\pi\mathrm{I}} & \underline{\underline{Q}}_{\pi\pi} \end{pmatrix} \begin{pmatrix} \underline{\underline{Q}}_{\mathrm{I}\Delta} \\ \underline{\underline{Q}}_{\pi\Delta} \end{pmatrix} \\ \begin{pmatrix} \underline{\underline{Q}}_{\Delta\mathrm{I}} & \underline{\underline{Q}}_{\Delta\pi} \end{pmatrix} \begin{pmatrix} \underline{\underline{Q}}_{\Delta\Delta} \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} \underline{\underline{Q}}_{\mathrm{II}} & \underline{\underline{Q}}_{\mathrm{I}\Delta} \\ \underline{\underline{Q}}_{\Delta\mathrm{I}} & \underline{\underline{Q}}_{\Delta\Delta} \end{pmatrix} \begin{pmatrix} \underline{\underline{Q}}_{\mathrm{I}\pi} \\ \underline{\underline{Q}}_{\Delta\pi} \end{pmatrix} \\ \begin{pmatrix} \underline{\underline{Q}}_{\pi\mathrm{I}} & \underline{\underline{Q}}_{\pi\Delta} \end{pmatrix} \begin{pmatrix} \underline{\underline{Q}}_{\pi\pi} \end{pmatrix} \end{pmatrix} \tag{5.3.6}$$

Las siguientes expresiones, las cuales son claras en vista de Ec.(5.3.6), será usado en la secuela

$$\underline{\underline{Q}} = \begin{pmatrix} \underline{\underline{Q}}_{\Pi\Pi} & \underline{\underline{Q}}_{\Pi\Delta} \\ \underline{\underline{Q}}_{\Delta\Pi} & \underline{\underline{Q}}_{\Delta\Delta} \end{pmatrix} = \begin{pmatrix} \underline{\underline{Q}}_{\Sigma\Sigma} & \underline{\underline{Q}}_{\Sigma\pi} \\ \underline{\underline{Q}}_{\pi\Sigma} & \underline{\underline{Q}}_{\pi\pi} \end{pmatrix} \tag{5.3.7}$$

Aquí:

$$\begin{array}{l} \underline{\underline{Q}}_{\Pi\Pi} \equiv \begin{pmatrix} \underline{\underline{Q}}_{\mathrm{II}} & \underline{\underline{Q}}_{\mathrm{I}\pi} \\ \underline{\underline{Q}}_{\pi\mathrm{I}} & \underline{\underline{Q}}_{\pi\pi} \end{pmatrix}, \ \underline{\underline{Q}}_{\Pi\Delta} \equiv \begin{pmatrix} \underline{\underline{Q}}_{\mathrm{I}\Delta} \\ \underline{\underline{Q}}_{\pi\Delta} \end{pmatrix} \\ \underline{\underline{Q}}_{\Delta\Pi} \equiv \begin{pmatrix} \underline{\underline{Q}}_{\Delta\mathrm{I}} & \underline{\underline{Q}}_{\Delta\pi} \end{pmatrix}, \quad \underline{\underline{Q}}_{\Delta\Delta} \equiv \begin{pmatrix} \underline{\underline{Q}}_{\Delta\Delta} \end{pmatrix} \end{array} \tag{5.3.8}$$

y

$$\begin{array}{l} \underline{\underline{Q}}_{\Sigma\Sigma} \equiv \begin{pmatrix} \underline{\underline{Q}}_{\mathrm{II}} & \underline{\underline{Q}}_{\mathrm{I}\Delta} \\ \underline{\underline{Q}}_{\Delta\mathrm{I}} & \underline{\underline{Q}}_{\Delta\Delta} \end{pmatrix}, \ \underline{\underline{Q}}_{\Sigma\pi} \equiv \begin{pmatrix} \underline{\underline{Q}}_{\mathrm{I}\pi} \\ \underline{\underline{Q}}_{\Delta\pi} \end{pmatrix} \\ \underline{\underline{Q}}_{\pi\Sigma} \equiv \begin{pmatrix} \underline{\underline{Q}}_{\pi\mathrm{I}} & \underline{\underline{Q}}_{\pi\Delta} \end{pmatrix}, \quad \underline{\underline{Q}}_{\pi\pi} \equiv \begin{pmatrix} \underline{\underline{Q}}_{\pi\pi} \end{pmatrix} \end{array} \tag{5.3.9}$$

5.3.3. El software DVS local para $\underline{\underline{S}}$

Sea $\underline{w}_{\Delta} \in W$, y recordando la Ec.(5.1.1); entonces:

$$\left(\underline{\underline{S}}\underline{w}\right)_{\gamma} = \underline{\underline{Q}}_{\Delta\Delta}\underline{w}_{\Delta} - \underline{\underline{Q}}_{\Delta\Pi}\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{\sim 1}\underline{a}'\underline{\underline{Q}}_{\Pi\Delta}\underline{w}_{\Delta} \tag{5.3.10}$$

En esta ecuación el significado de los términos $\underline{\underline{Q}}_{\Delta\Delta}\underline{w}_{\Delta}$ y $\underline{\underline{Q}}_{\Pi\Delta}\underline{w}_{\Delta}$ son claros dado que ambos $\underline{\underline{Q}}_{\Delta\Delta}$ y $\underline{\underline{Q}}_{\Pi\Delta}$ son transformaciones lineales bien definidas de W^{γ} en sí mismo. Algo similar ocurre cuando el operador $\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{\sim 1}$ es aplicado a $\underline{\underline{a}}'\underline{\underline{Q}}_{\Pi\Delta}\underline{w}_{\Delta}$, dado que esta es también una transformación lineal global. También tiene que ser entendido que, cuando es aplicado, el vector local $\left(\underline{\underline{a}}'\underline{\underline{Q}}_{\Pi\Delta}\underline{w}_{\Delta}\right)_{\gamma}$ ya ha sido almacenado en el procesador γ y en cada uno de los otros procesadores. Debido al carácter global del operador $\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{\sim 1}$ un software especial fue desarrollado para ello.

5.3.4. El software DVS local para $\left(\underline{\underline{A}}_{\Pi\Pi}\right)^{\sim 1}$

El software local que fue desarrollado está basado en la siguiente fórmula:
"Sea $\underline{w}_{\Pi} \in W^{r}(\Pi)$ *y* $\underline{v}_{\Pi} = \left(\underline{\underline{A}}_{\Pi\Pi}\right)^{\sim 1}\underline{w}_{\Pi}$, *entonces*

$$\underline{\underline{a}}'\left(\underline{\underline{A}}^{t}_{\pi\pi} - \underline{\underline{A}}^{t}_{\pi\mathrm{I}}\left(\underline{\underline{A}}^{t}_{\mathrm{II}}\right)^{\sim 1}\underline{\underline{A}}^{t}_{\mathrm{I}\pi}\right)\underline{v}_{\pi} = \underline{w}_{\pi} - \underline{\underline{A}}^{t}_{\pi\mathrm{I}}\left(\underline{\underline{A}}^{t}_{\mathrm{II}}\right)^{\sim 1}\underline{w}_{\mathrm{I}},\ y\ \underline{\underline{j}}\underline{v}_{\pi} = 0 \quad (5.3.11)$$

junto con

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

Para aplicar esta fórmula iterativamente, en cada procesador γ fue necesario desarrollar software local capaz de llevar a cabo las siguientes operaciones:

$$\underline{\underline{a}}'\left\{\underline{\underline{Q}}_{\pi\pi} - \underline{\underline{Q}}_{\pi\mathrm{I}}\left(\underline{\underline{Q}}_{\mathrm{II}}\right)^{\sim 1}\underline{\underline{Q}}_{\mathrm{I}\pi}\right\}\underline{v}_{\pi} = \underline{w}_{\pi} - \underline{\underline{a}}'\underline{\underline{Q}}_{\pi\mathrm{I}}\left(\underline{\underline{Q}}_{\mathrm{II}}\right)^{\sim 1}\underline{w}_{\mathrm{I}} \quad (5.3.12)$$

y, una vez que hay convergencia, la siguiente operación autónoma es llevada a cabo:

$$\underline{v}_{\mathrm{I}} = \left(\underline{\underline{Q}}_{\mathrm{II}}\right)^{\sim 1}\left(\underline{w}_{\mathrm{I}} - \underline{\underline{Q}}_{\mathrm{I}\pi}\underline{v}_{\pi}\right) \quad (5.3.13)$$

Aquí vemos que exceptuando todas las transformaciones lineales involucradas son autónomas y pueden ser expresadas por medio de matrices locales definidas en cada procesador. En el software del DVS que es tema de este trabajo, dichas matrices no fueron construidas pero reconocemos que en algunos problemas dicha opción podría ser más competitiva,

5.3.5. El software DVS local para $\underline{\underline{S}}^{\sim 1}$

El software local que fue desarrollado está basado en la siguiente fórmula: *"Cuando* $\underline{w}_{\Delta} \in W(\Delta)$*, entonces*

$\underline{\underline{S}}^{\sim 1}\underline{w} = \left(\underline{\underline{A}}^{\sim 1}\underline{w}_{\Delta}\right)_{\Delta}$ "

Por lo tanto, si $\underline{v} \in W^r$ está definido por la condición $\underline{\underline{A}}\underline{v} = \underline{w}_{\Delta}$ y que está escrito en la forma $\underline{v} = \underline{v}_{\mathrm{I}} + \underline{v}_{\Delta} + \underline{v}_{\pi}$, entonces $\underline{\underline{S}}^{\sim 1}\underline{w} = \underline{v}_{\Delta}$. Una forma más explícita de la condición $\underline{v} \in W^r$ es $\underline{\underline{j}}\underline{v}_{\pi} = 0$. Esta última condición junto con la ecuación $\underline{\underline{A}}\underline{v} = \underline{w}_{\Delta}$ da una visión del problema global cuya solución, en el software en paralelo hemos desarrollado, estuvo basada en el esquema iterativo: *"Sea* $\underline{w}_{\Delta} \in W(\Delta)$ *y* $\underline{v}_{\Delta} \equiv \underline{\underline{S}}^{\sim 1}\underline{w}_{\Delta} = \left(\underline{\underline{A}}^{\sim 1}\underline{w}_{\Delta}\right)_{\Delta}$*, entonces en el procesador* γ:

$$\underline{\underline{a}}'\left(\underline{\underline{Q}}_{\pi\pi} - \underline{\underline{Q}}_{\pi\Sigma}\left(\underline{\underline{Q}}_{\Sigma\Sigma}\right)^{\sim 1}\underline{\underline{Q}}_{\Sigma\pi}\right)\underline{v}_{\pi} = -\underline{\underline{a}}'\underline{\underline{Q}}_{\pi\Sigma}\left(\underline{\underline{Q}}_{\Sigma\Sigma}\right)^{\sim 1}\underline{w}_{\Delta},\ y\ \underline{\underline{j}}\underline{v}_{\pi} = 0 \tag{5.3.14}$$

Una vez que $\underline{v}_{\pi}$ *ha sido obtenida,* $\underline{v}_{\Delta}$ *está dada por*

$$\underline{\underline{S}}^{\sim 1}\underline{w}_{\Delta} = \underline{v}_{\Delta} = \left(\left(\underline{\underline{Q}}_{\Sigma\Sigma}\right)^{\sim 1}\left(\underline{w}_{\Sigma} - \underline{\underline{Q}}_{\Sigma\pi}\underline{v}_{\pi}\right)\right)_{\Delta}$$ "

En el procesador γ que este siendo considerado, las ecs.(5.3.14) y (5.3.5) son:

$$\underline{\underline{a}}'\left(\underline{\underline{Q}}_{\pi\pi} - \underline{\underline{Q}}_{\pi\Sigma}\left(\underline{\underline{Q}}_{\Sigma\Sigma}\right)^{\sim 1}\underline{\underline{Q}}_{\Sigma\pi}\right)\underline{v}_{\pi} = -\underline{\underline{a}}'\underline{\underline{Q}}_{\pi\Sigma}\left(\underline{\underline{Q}}_{\Sigma\Sigma}\right)^{\sim 1}\underline{w}_{\Delta},\ y\ \underline{\underline{j}}\underline{v}_{\pi} = 0 \tag{5.3.15}$$

y

$$\underline{v}_{\Delta} = \left(\left(\underline{\underline{Q}}_{\Sigma\Sigma}\right)^{\sim 1}\left(\underline{w}_{\Sigma} - \underline{\underline{Q}}_{\Sigma\pi}\underline{v}_{\pi}\right)\right)_{\Delta}$$ "

respectivamente.

5.3.6. Aplicaciones del Método de Gradiente Conjugado (CGM)

Hay tres instancias en las cuales el CGM es aplicado: *i*) para invertir $\underline{\underline{A}}_{\mathrm{III}}$; *ii*) para invertir $\underline{\underline{A}}$; *iii*) para resolver iterativamente la *ecuación global* –tal ecuación podría ser: cualquiera de las ecuaciones Ec.(4.1.31), Eq.(4.1.32), Eq.(4.1.5) o Eq.(4.1.35), dependiendo del *algoritmo DVS* -que sea aplicado-. Además, debe mencionarse que las inversas de las *matrices locales*: $\underline{\underline{Q}}_{\mathrm{II}}$ y $\underline{\underline{Q}}_{\Sigma\Sigma}$ pueden ser obtenidas ya sea por un método directo o por uno iterativo; en el *software del DVS* aquí reportado, está última opción fue la que se eligió y el CGM también fue aplicado a ese nivel.

Capítulo 6

DDM con DVS y su implementación computacional.

Uno de los objetivos de este trabajo doctoral es generar una solución de software que implemente los algoritmos DVS en equipos de cómputo de alto rendimiento. De manera precisa, se presenta una descripción de la implementación computacional que se hizo de los algoritmos para la solución numérica de la ecuación de elasticidad. Se presentan las características técnicas consideradas para la implementación, es decir de cómo se aprovecha el cómputo en paralelo para llevar a cabo los cálculos numéricos.

Para ilustrar el análisis y el diseño del software se describe la métodología aplicada y se presentan los diagramas que describen el diseño del software.

Recordando el paradigma orientado a objetos tenemos que los procedimientos están asociados a los datos dando lugar a tipos de datos abstractos denominados clases. Entonces cada entidad matemática, ya sea una expresión o un algoritmo puede ser representado como una de estas entidades. A su vez las relaciones entre clases están dadas como diagramas ya sea de Estado, de Secuencia o de Colaboración.

Los métodos de Descomposición de Dominio están representados por los algoritmos que se desarrollaron en el capítulo 4, específicamente las ecuaciones (4.1.31) ,(4.1.32), (4.1.5) y (4.1.35).

Dado que la descripción detallada de todo el sistema esta fuera del alcance de esta tesis, aquí solo se describen los procedimientos generales en los que están involucradas las clases más sobresalientes y se describirá el funcionamiento general que da como resultado la solución numérica de una Ecuación Diferencial

Parcial.

La ecuación (3.2.20) genera un sistema de ecuaciones lineales que al resolverlo proporciona una solución numérica de la EDP. Ahora bien, como ya hemos visto, es la aplicación de algún método de discretización el que genera dicho sistema.

Una vez obtenido el sistema de ecuaciones equivalente en el DVS entonces podemos aplicar algún método de solución iterativo. Lo que se modela son las entidades matemáticas que permiten la aplicación del método iterativo así como las operaciones del álgebra lineal numérica y la aplicación de los operadores que implican intercambio de información entre subdominios.

De forma general, el sistema de cómputo efectúa las operaciones y procedimientos para obtener la solución numérica de la ecuación de elasticidad apoyado en el diseño orientado a objetos y utilizando como herramienta de modelado UML. Con ello se presenta el diagrama de casos de uso del sistema computacional, se muestran las actividades principales que efectuará el sistema así como las entidades que interactúan con el propio sistema.
Para cada entidad matemática que procesa los datos se genera un caso de uso el cual posteriormente será la base para una clase o un paquete (conjunto de clases) que interactúan entre ellos para generar la solución numérica.
Un caso de uso es una unidad de funcionalidad del sistema la cual, a su vez, es susceptible de aplicarle la misma metodología. Esto se hace en las secciones siguientes donde se muestran y describen los componentes principales que modelan dicho caso de uso.

6.1. Análisis del sistema general

Para resolver el problema con condiciones de frontera (BVP) se genera un algoritmo que se sigue para obtener una solución numérica. Se modela en software este algoritmo, siendo este último parametrizable por lo que permite resolver las ecuaciones con distintos datos del problema. El algoritmo también requiere una serie de métodos numéricos y una serie de operaciones matemáticas que igualmente se modelan como componentes de software.

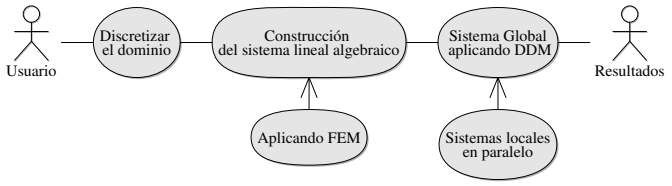

Figura 6.1: Diseño General

El procedimiento general a seguir cuando se considera paralelizar el problema aunque no se dan mayores detalles y que se muestra en la Figura (6.1) es el siguiente:

1. Para comenzar se toma el dominio del problema que en principio es un espacio infinito de puntos con una frontera bien definida y se genera un espacio finito de puntos sobre los que se construirá la solución numérica.

2. El siguiente paso es construir un sistema de ecuaciones lineales lo cual se hará al aplicar algún método de discretización al sistema de EDP's. En este caso puede ser cualquier método pero para el problema que estamos abordando en este trabajo se decidió que sería el Método de Elemento Finito.

3. Una vez que se tiene el sistema de ecuaciones lineales este se resolverá usando algún método numérico como sería CGM si la matriz del sistema es simétrica positiva definida o algún método de solución para matrices que no cumplan esa condición.

4. El procesamiento de los cálculos numéricos se organiza de tal forma que pueda repartirse entre varios procesadores para concluir con la tarea en el menor tiempo posible y que sea proporcional al número de procesadores involucrados en el cálculo.

Es en el punto 3 donde se requiere que se paralelize el proceso el cual es muy conveniente que se haga con algún método de descomposición de dominio (DDM). En este caso el DDM será el DVS.

En las siguientes secciones se hace una breve descripción en términos de casos de uso de cada uno de los componentes de software que se diseñaron para modelar los procesos y métodos numéricos mencionados justo arriba. Además de que en algunos casos se muestran las clases que los componen así como las relaciones entre ellas.

De manera general en el análisis de requerimientos de cada módulo se tendrían que especificar los datos que tienen que ser procesados y describir brevemente el proceso matemático utilizado para ese fin, en particular, los procesos computacionales de los cuales posteriormente surgirá la arquitectura de la aplicación así como el código.

Si a los casos de uso les aplicamos nuevamente el análisis generamos una nueva serie de casos de uso, los correspondientes al manejo de la geometría del dominio así como de la organización de los nodos de la discretización fina del dominio queda en un grupo de casos de uso que modelan los procesos de uno de los subdominios. Esos subdominios se obtienen cuando se aplica la descomposición del dominio.

El método de discretización numérico es otro caso de uso al que posteriormente se le aplica el análisis y genera un sistema de software que puede funcionar de manera independiente lo cual permite que ese proceso también se pueda llevar a cabo en paralelo.

Los algoritmos del DVS y todo lo referente a la paralelización de los cálculos numéricos se presentan en otro grupo de casos de uso al igual que los métodos numéricos y las operaciones del álgebra lineal numérica que se muestran como

casos de uso y todo lo anterior se representa en el siguiente diagrama de casos de uso.

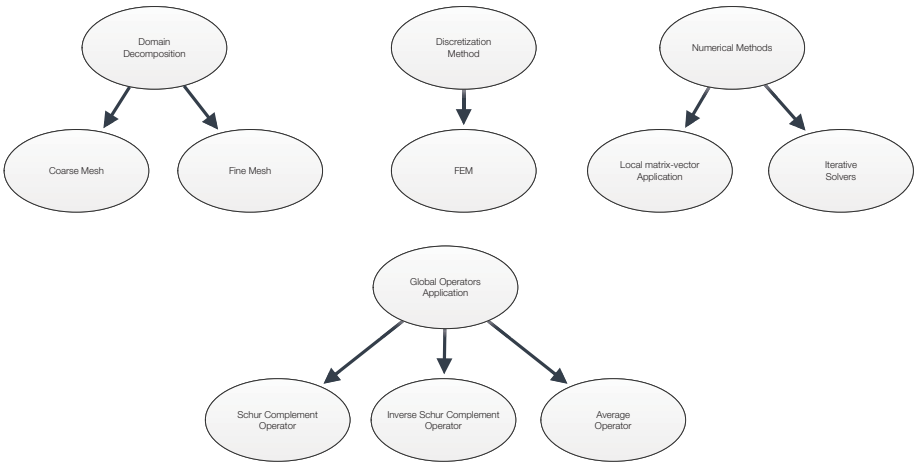

Figura 6.2: Casos de Uso Generales

6.2. Paralelización

En el área de las matemáticas computacionales la cantidad de operaciones aritméticas crece proporcionalmente con el número de puntos en la discretización del problema. Algunas metodologías intentan resolver los problemas matemáticos algebraicamente es decir, obtener una solución analítica que después sea expresada numéricamente al ser evaluada. En los métodos numéricos se aproxima a priori en un conjunto de puntos del dominio y posteriormente se intenta obtener el valor de la función. Es por ello que se requiere un número cada vez mayor de operaciones para tener valores mejor aproximados. El cómputo serial no es suficiente para resolver estos problemas debido a que el tiempo de cálculo es muy largo. Es por ello que se han buscado alternativas y una de ellas es el cómputo en paralelo el cual consiste en separar el problema para que las operaciones requeridas no sean dependientes y puedan ser ejecutadas por un conjunto de procesadores, posteriormente se recolecta la información y se arma la solución global.

Dentro de las ventajas de hacer el cálculo en paralelo están

1. La reducción del tiempo de procesamiento. Dado que las operaciones se ejecutan al mismo tiempo se reduce el tiempo de procesamiento.
2. Escalamiento. Esto significa que el problema puede ser adaptado de acuerdo con su tamaño, es decir se pueden resolver sistemas de alta dimensionalidad.
3. Gran capacidad de memoria. Esto significa que la discretización puede ser cada vez mayor.

Algunas desventajas

1. Programación más complicada. Esto se debe a que hay que tener en cuenta muchas variables y eventos que en cómputo secuencial no se presentan.
2. Incremento en el uso de memoria. Pues existirán algunos datos que estén duplicados en cada procesador involucrado en el proceso.
3. Incremento en el orden de complejidad. Esto significa que dado que estamos trabajando con un mayor número de problemas entonces el número de algunas operaciones se incrementa.

Otro aspecto fundamental del cómputo en paralelo es el intercambio de información entre los procesadores lo que requiere de un buen performance en las comunicaciones. Es por ello que requerimos un bus de datos con capacidad mayor a la habitual y sincronizado de manera adecuada, esto porque los retrasos en la transmisión de los resultados intermedios puede provocar errores en la solución buscada y hasta generar un dead lock en el sistema, esto como resultado de que el valor del cual depende el número de iteraciones no se alcance en un momento determinado.

6.3. Geometría del problema

La discretización del dominio es el primer paso de la propia discretización del problema ya que esta determina de manera única el sistema de ecuaciones lineales. Las instrucciones dadas en este documento consideran que estos procesos se ejecutan de manera local y no hay intercambio de información entre procesadores.

1. Se comienza con la construcción del sistema de nodos original que usa la discretización fina y luego, como cualquier método de descomposición de dominio, introduce una malla gruesa.
2. Se generan las listas de los nodos en el espacio DVS para tener un sistema de nodos completamente sin traslape. Esto implica también el etiquetado de los nodos para su identificación en alguno de los subconjuntos en los que se separa el DVS.

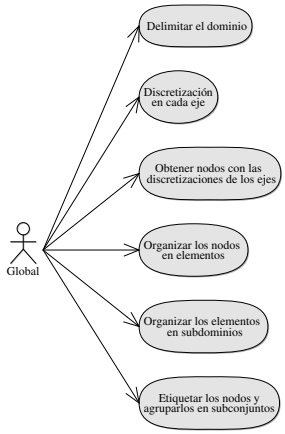

Figura 6.3: Casos de Uso Geometría

En la solución de software no se construye el sistema original sino que el procedimiento actúa de otra manera de tal forma que se aprovechan ciertas características de la geometría y de la simetría del operador diferencial para hacer el procesamiento en paralelo desde este punto. Después aplicar algún método de discretización para generar una matriz simétrica y bandada sobre la cual se aplica el siguiente paso de la metodología DVS.

La clase que se obtiene de los casos de uso mostrados en el diagrama anterior es la que representa la malla conformada por los nodos de la discretización.

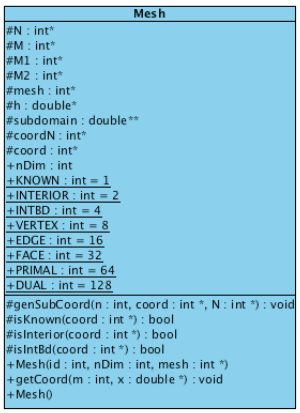

Figura 6.4: Geometría representada por la clase Mesh

6.4. Método de discretización numérico (FEM)

Para generar el sistema lineal algebraico para el operador de elasticidad,como ya se mencionó, se utiliza FEM. Siguiendo el procedimiento en el capítulo correspondiente se modela el proceso en un diagrama de actividades. De ese mismo procedimiento podemos obtener los casos de uso para construir los componentes de software. La discretización del dominio y la construcción de la geometría ya fue descrito previamente. Sobre el conjunto finito de puntos que es la discretización del espacio,se construye una base de funciones que para este trabajo son funciones lineales.

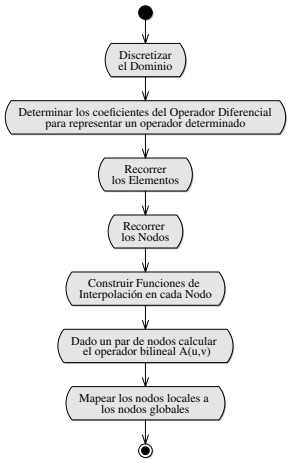

Figura 6.5: Procedimiento de FEM

En la construcción del sistema de ecuaciones lineales se obtienen las submatrices de cada subdominio mientras que las relacionadas con los subconjuntos de nodos (primales, duales e interiores) en realidad no las construimos sino que se generan listas con los nodos que componen dichos subconjuntos del DVS. Con esas listas se lleva a cabo la operación matriz vector señalando cuáles son las columnas y cuáles son los renglones que se tienen que operar.

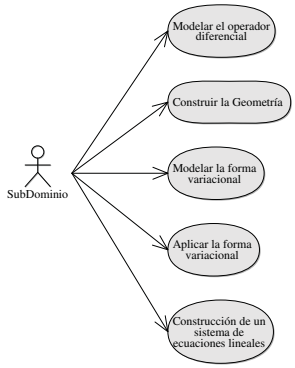

Figura 6.6: Casos de Uso de FEM

El operador diferencial puede tener, entre otros, operadores de gradiente y de divergencia los cuales quedan representados en la formulación variacional de manera básica como derivadas que serán funciones analíticas. En el caso de las integrales que aparecen en la misma formulación variacional estás serán calculadas con un procedimiento numérico que es , en este caso, la integración gaussiana.

Para construir el sistema de ecuaciones lineales hay que recorrer la geometría de una manera específica y calcular las integrales usando las funciones lineales, que ya mencionamos, sobre los elementos y almacenarla en la matriz del sistema lineal algebraico. Para calcular el lado derecho de ese mismo sistema también se calcula una integral utilizando el lado derecho de la EDP junto con las funciones lineales. Las clases que modelan lo anterior están dadas en los siguentes diagramas de clase que muestran únicamente su comportamiento estático.

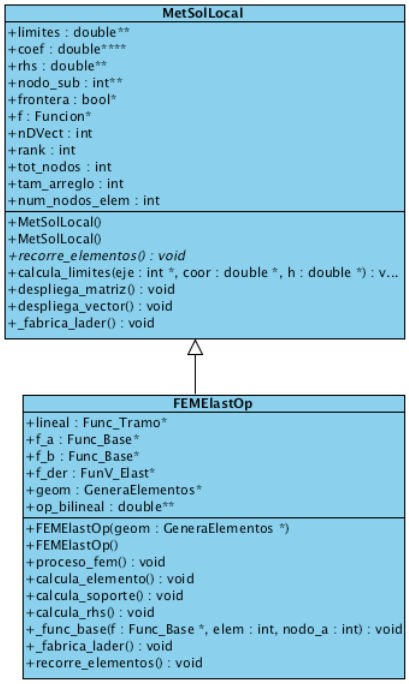

Figura 6.7: Clases de FEM

En esta clase se modela la solución numérica se genera al construir un sistema lineal algebraico obtenido de la formulación variacional, la cual también es parte de esta clase. Si seguimos el algoritmo vemos que se construyen un conjunto de combinaciones lineales de funciones polinomiales mostradas en la siguiente jerarquía de clase. Las clases que se muestran a continuación son las que construyen la soluciones locales y la que modela el operador variacional proveniente del operador de elasticidad.

En la clase *FEMElastOp* se generan los coeficientes de la matriz, es decir se aplican las funciones base en el operador variacional y se integra numéricamente.

6.5. Diseño del DVS

La implementación de software requiere de una forma sistemática para diseñarlo y desarrollarlo. Así que comenzamos el modelado del sistema que usa el marco del DVS para resolver sistemas de ecuaciones siguiendo una métodología orientada a objetos y además nos apoyaremos en la notación con UML. La forma de proceder esta documentada en este capítulo y se presenta a continuación.

El algoritmo DVS implementado en una solución de software considera que ya se cuenta con las submatrices locales y con la discretización del dominio bien

establecida, esto es, que se tienen identificados los nodos de los subconjuntosΠ, Δ, π, I, Σ y por ende ya conoce los nodos que se usan para intercambiar información. En esta implementación se aprovecha el cómputo en paralelo desde que se aplica la discretización del dominio así como en el cálculo de las submatrices al aplicar el método de discretización del operador diferencial.

Una vez que se han establecido los componentes de software mínimos que provienen del análisis de los procesos locales y del intercambio de información necesario durante el procesamiento en paralelo se lleva a cabo el análisis de los algoritmos DVS para implementarlos primero en un modelo orientado a objetos y después es una solución de software.

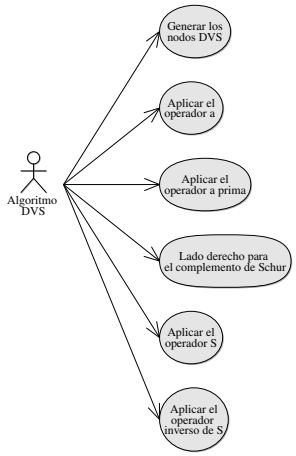

Figura 6.8: Procedimiento DVS

Se decide qué algoritmo DVS se va utilizar para obtener la solución numérica. Este proceso involucra la aplicación de un método iterativo sobre la fórmula del método DVS correspondiente. Como ya se mencionó cualquiera de los métodos DVS involucra la aplicación de los operadores S, S^{-1} , a y a'. La aplicación de los operadores S y S^{-1} son en realidad la aplicación de ciertas submatrices que están definidas en cada subdominio, recordemos que a cada subdominio en un momento se le asignará un procesador independiente. Mientras que la aplicación de los operadores a y a' indican intercambio de información entre subdominios.

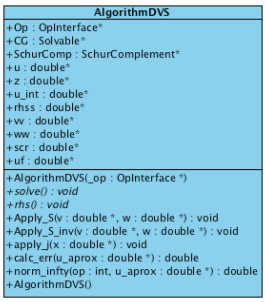

Figura 6.9: Procedimiento DVS

La aplicación de algunas submatrices requieren la solución de sistemas de ecuaciones ya sea globales o locales. En el segundo caso, el cual es más sencillo, utiliza cualquier método de solución numérico ya sea directo o iterativo. El caso de los sistemas globales requiere un método iterativo porque necesita del intercambio de información entre subdominios (procesadores).

Se resalta el momento en que se intercambia información y se modelan los procesos locales los cuales en este punto serán aplicaciones matriz vector ,los cálculos de los productos interiores locales y de los promedios. El proceso de intercambio de información entre procesadores esta modelado en una clase.

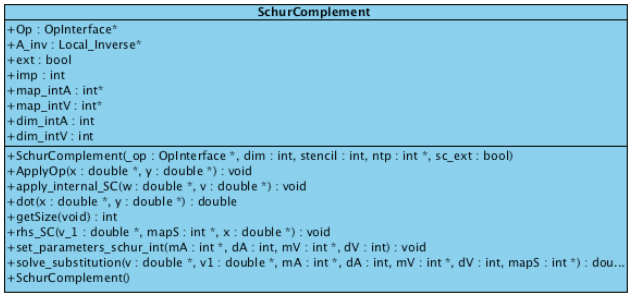

Figura 6.10: Complemento de Schur

donde se implementa la aplicación local de los principales operadores que son, como ya lo dijimos, S, S^{-1} , a y a'.

6.6. Arquitectura del software

La arquitectura del software es la estructura y la organización de los componentes de software. En este análisis tenemos dos tipos principales de componentes, los que se ejecutan en un solo procesador de manera completamente

independiente y los que operan de manera coordinada e intercambian información entre procesadores.

Esta clasificación se representa en el siguiente diagrama donde se transforman los casos de uso en procesos colaborativos que ejecutan tareas específicas, ya sea de manera local o en paralelo. También se describen de manera general las tareas que llevan a cabo en el algoritmo de solución numérica.

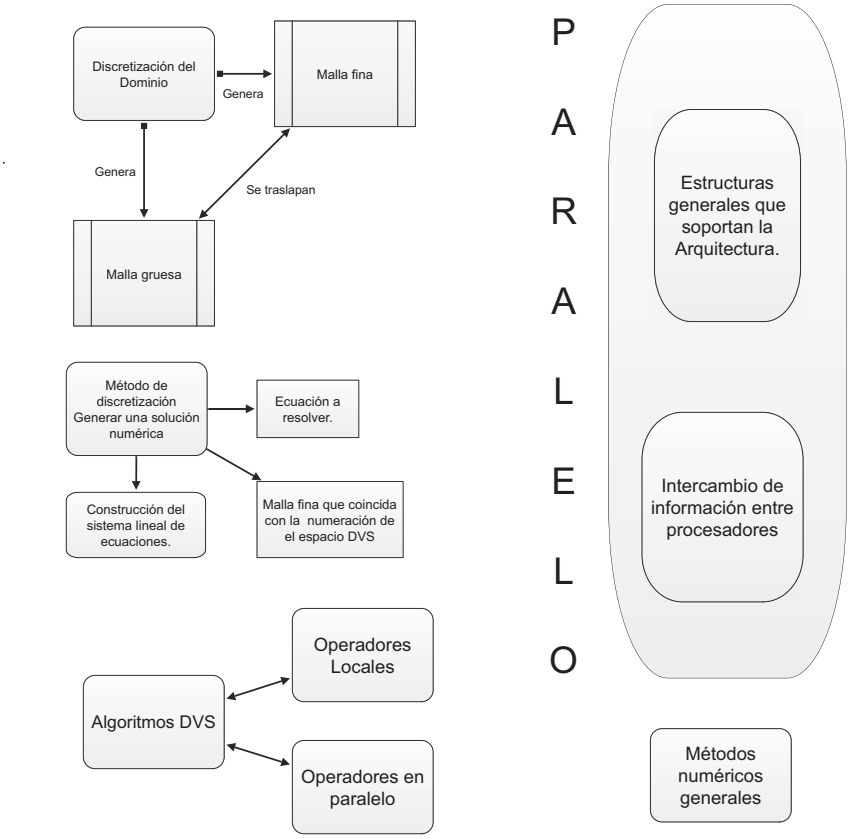

Figura 6.11: Arquitectura de Software

En el diagrama aparecen procesos que se pueden agrupar en :

- Los que modelan la geometría, tanto de la discretización del dominio en una malla fina como el de la descomposición de dominio.
- Los que llevan a cabo el método de discretización, que en este caso es FEM y que construye las estructuras del álgebra lineal numérica con las que se calcula la solución numérica de la EDP.

- Los que modelan los métodos numéricos de solución y los cálculos lógicos aritméticos que se usan en los otros procesos.
- Los que modelan la arquitectura en paralelo y que intercambian información entre procesos que se ejecutan de manera local.

6.7. Descripción del software

El manejo y la sincronización entre los procesadores se efectúa utilizando MPI, que es un mecanismo de paso de mensajes que trabaja con C y C++. Las clases que realizan los cálculos de manera local instancian una clase que cual contiene los métodos para sincronizar las operaciones del calculo a nivel local con los métodos que realizan el ajuste a nivel global. A diferencia del procedimiento teórico, la solución de software inicia considerando una partición gruesa la cual será controlada por el MPI. Esto significa que se tiene una topología virtual la cual es proporcionada por MPI y la cual controla esa malla gruesa que esta conformada por una malla ortogonal.

En el caso de estudio cuando se resuelve un sistema lineal de ecuaciones que proviene de la discretización de una ecuación diferencial parcial utilizando el algoritmo del gradiente conjugado (CGM) puede calcularse la solución en paralelo dado que los componentes del vector que estamos calculando dependen de operaciones con submatrices que están desacopladas.

El esquema sin maestro consiste en tener una tarea global que será resuelta por un conjunto de procesadores coordinados únicamente por la secuencia de operaciones que realizan. Este esquema también tiene la característica de no concentrar los resultados de todos los subprocesos y de no requerir una solución global. En el caso de los DDM hay una serie de operaciones que son las que controlan el método iterativo pero únicamente en el sentido que no y almacena la información de los nodos en la frontera interior, mientras que los esclavos resuelven los problemas locales. Este esquema fue propuesto por Herrera & de la Cruz [32].

Se conocen las coordenadas y las dimensiones del dominio así como el número de particiones por eje. Esto esta especificado en los atributos.

Al tener un conjunto de subdominios completamente independiente, pero que podemos controlar los nodos derivados en cada uno de ellos de tal forma que conozcamos a sus ancestros y podamos asociarlos con sus respectivos nodos derivados en otros subdominios, entonces se genera una discretización de cada uno de los subdominios y del problema en cada uno de ellos.

La manera en la que se ajustan los nodos al momento de hacer el intercambio de información es directa, nodo a nodo y esto es debido a que las caras de los subdominios son conformables y ajustan de manera precisa entre ellas. Este procedimiento genera una malla fina en cada uno de los subdominios. Los nodos en la frontera interior de cada subdominio estaran asociados con los nodos de frontera interior de los subdominios vecinos a través de su relación, como se mencionó anteriormente, con un nodo ancestro.

El procedimiento que inicia y finaliza la topología virtual utilizando DVS se considera un caso de uso . Una vez iniciada la topología virtual se genera la malla fina en cada subdominio. Dado que estamos considerando un procesador de la topología como un subdominio de la malla se necesita modelar el problema de manera local.

Se conocen las coordenadas y las dimensiones del dominio así como el número de particiones por eje. Esto esta especificado en los atributos. Al tener un conjuno de subdominios completamente independiente, pero que podemos controlar los nodos derivados en cada uno de ellos de tal forma que conozcamos a sus ancestros y podamos asociarlos con sus respectivos nodos derivados en otros subdominios, entonces se genera una discretización de cada uno de los subdominios y del problema en cada uno de ellos.

La manera en la que se ajustan los nodos al momento de hacer el intercambio de información es directa, nodo a nodo y esto es debido a que las caras de los subdominios son conformables y ajustan de manera precisa entre ellas. Este procedimiento genera una malla fina en cada uno de los subdominios. Los nodos en la frontera interior de cada subdominio estaran asociados con los nodos de frontera interior de los subdominios vecinos a través de su relación, como se mencionó anteriormente, con un nodo ancestro.

Se considera un caso de uso el procedimiento que inicia y finaliza la topología virtual utilizando DVS.

Una vez iniciada la topología virtual se genera la malla fina en cada subdominio. Dado que estamos considerando un procesador de la topología como un subdominio de la malla se necesita modelar el problema de manera local. Eso significa que se tienen que ejecutar y controlar los siguientes procedimientos.

6.7.1. Descripción del comportamiento dinámico del software.

El esquema sin maestro para el DVS fue propuesto en [40] y se implementó en este programa de cómputo considerando las estrategias de software ahí descritas. Este esquema tiene dos consideraciones : 1) que no hay un procesador que centralice las operaciones, que coordine la ejecución del programa y 2) que hay operaciones que se hacen con los datos intercambiados exclusivamente entre vecinos de tal forma que se propague el resultado a todos los procesadores.

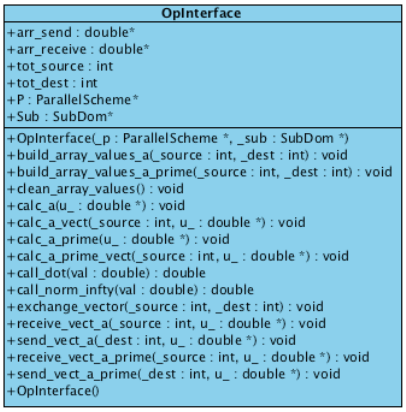

Figura 6.12: Clase que realiza los intercambios de información entre procesadores

Una clase que recibe los parámetros desde la linea de comandos y determina principalmente las partciones de la malla fina y de la malla gruesa.

La clase ParallelScheme es la que se encarga de la topología virtual en la que está la malla gruesa. Esta herramienta facilita la interacción entre los subdominios así como de la aplicación de los operadores que actúan sobre los nodos de la interface.Activa y desactiva al propio MPI.

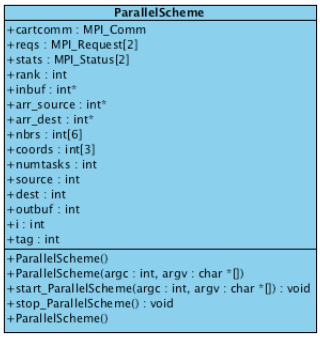

Figura 6.13: Clase que controla a MPI

Esta segunda consideranción garantiza que la sincronización de la ejecución entre los procesadores se da por el intercambio de información, lo cual indica que todos los procesadores están ejecutando la misma instrucción.

El código se ejecuta en cada procesador y quien se encarga del paso de información entre procesadores es el propio MPI así como de las operaciones que requieran centralizar un cáculo, que en este caso es sólo uno: el producto interior

global. Entonces las operaciones globales que involucren datos de todos los subdominios se llevan a cabo por medio de instructivas de MPI y de las topologías virtuales. Mientras que, como ya lo mencionamos, las operaciones entre procesadores sólo involucran a los vecinos de un subdominio. Para el intercambio de información entre procesadores también se encarga MPI por medio de la topología que puede ser parametrizada para trabajar en distintas dimensiones, en el caso de elasticidad se requiere que sea en 3D con más de un grado de libertad por nodo de discretización del Dominio.

También se considera que el método de discretización se ejecuta después de haber realizado la descomposición del Dominio. Con ello se se aprovechan aún más los recursos del cómputo en paralelo.

El main del programa esta en la clase ParallelDVS que hace las instancias de las clases que involucran todo el proceso.

La clase SubDom se encarga de la malla fina y de la aplicación del método de discretización. Algunos de sus atributos son los que contienen las submatrices, las listas de nodos pertenecientes a los distintos subconjuntos. También controla los valores en los nodos de interface y construye las listas de valores que se han de intercambiar entre subdominios.

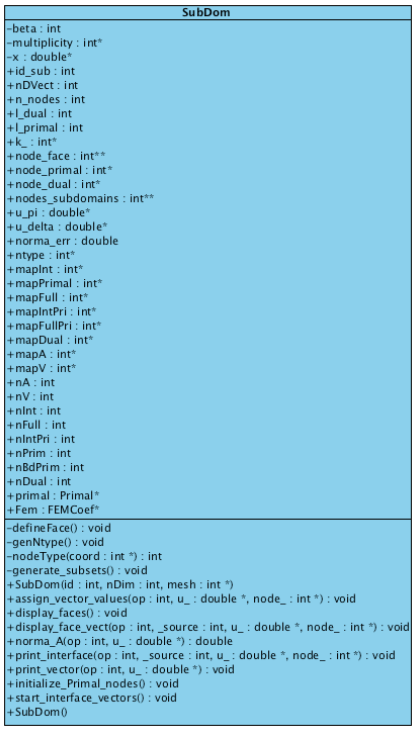

Figura 6.14: Clase que realiza las operaciones a nivel local

La clase OpInterface se encarga de intercambiar los valores entre los nodos de frontera, es una clase compuesta de las dos anteriores para que colaboren en la aplicación de los operadores globales. En esta clase es importante describir el comportamiento de los métodos encargados de los operadores a y j así como de la aplicación del producto interior.

Capítulo 7

Experimentos Numéricos

En los experimentos numéricos que fueron llevados a cabo para probar la metodología y la correspondiente implementación en software usamos la ecuación de elasticidad con la discretización que se presenta en el capítulo tres. Por el momento sólo se ha implementado el algoritmo DVS-BDDC. Este problema con condiciones de frontera lo parametrízamos en los coeficientes de la ecuación asumiendo que es un problema homogéneo cuyos coeficientes de Lamè están dados como:

$$\begin{array}{l} \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} = 29{,}6412 \times 10^9 \frac{\mathrm{N}}{m^2} \\ \mu = \frac{E}{2(1+\nu)} = 27{,}3611 \times 10^9 \frac{\mathrm{N}}{m^2} \end{array} \tag{7.0.1}$$

Estos valores corresponden a una clase de hierro [34] cuyo módulo de Young E y su coeficiente de Poisson ν son

$$E = 68{,}95 \times 10^9 \frac{N}{m^2} \text{ y } \nu = 0{,}26 \tag{7.0.2}$$

respectivamente.

El dominio el problema fue tomado como$\Omega \subset \Re^3$ con una descomposición de dominio y una discretización en cada subdominio. En cada nodo α de la malla esta definido un campo vectorial $\underline{u}_\alpha$ con cada componente identificado como $u_{\alpha i}$ para $i = 1, 2, 3$.

Se toman condiciones de frontera homogéneas tipo Dirichlet $\underline{u}_\partial = 0$, sin embargo, se asume quecualquier problema con condiciones de frontera no homogéneas es fácilmente transformado en uno con condiciones cero en la frontera.

En los experimentos al problema le hemos asignado el siguiente lado derecho

$$\begin{array}{l} f_1 = -(\mu+\lambda)(-\sin\pi x \sin\pi y \sin\pi z \\ +\cos\pi x \cos\pi y \sin\pi z \\ +\cos\pi x \sin\pi y \cos\pi z)\pi^2 \\ +(-\mu)(-\sin\pi x \sin\pi y \sin\pi z \\ -\sin\pi x \sin\pi y \sin\pi z \\ -\sin\pi x \sin\pi y \sin\pi z)\pi^2 \end{array} \tag{7.0.3}$$

$$
\begin{aligned}
f_2 = &-(\mu+\lambda)(\cos\pi x\cos\pi y\sin\pi z \\
&-\sin\pi x\sin\pi y\sin\pi z \\
&+\sin\pi x\cos\pi y\cos\pi z)\pi^2 \\
&+(-\mu)(-\sin\pi x\sin\pi y\sin\pi z \\
&-\sin\pi x\sin\pi y\sin\pi z \\
&-\sin\pi x\sin\pi y\sin\pi z)\pi^2
\end{aligned}
\tag{7.0.4}
$$

$$
\begin{aligned}
f_3 = &-(\mu+\lambda)(\cos\pi x\sin\pi y\cos\pi z \\
&+\sin\pi x\cos\pi y\cos\pi z \\
&-\sin\pi x\sin\pi y\cos\pi z)\pi^2 \\
&+(-\mu)(-\sin\pi x\sin\pi y\sin\pi z \\
&-\sin\pi x\sin\pi y\sin\pi z \\
&-\sin\pi x\sin\pi y\sin\pi z)\pi^2
\end{aligned}
\tag{7.0.5}
$$

Con una solución analítica para el problema que está represesentada como

$$
\underline{u} = (\sin\pi x\sin\pi y\sin\pi z, \sin\pi x\sin\pi y\sin\pi z, \sin\pi x\sin\pi y\sin\pi z) \tag{7.0.6}
$$

El dominio el problema fue tomado como$\Omega \subset \Re^3$ con una descomposición de dominio y una discretización en cada subdominio. En cada nodo α de la malla esta definido un campo vectorial $\underline{u}_\alpha$ con cada componente identificado como $u_{\alpha i}$ for $i = 1, 2, 3$.

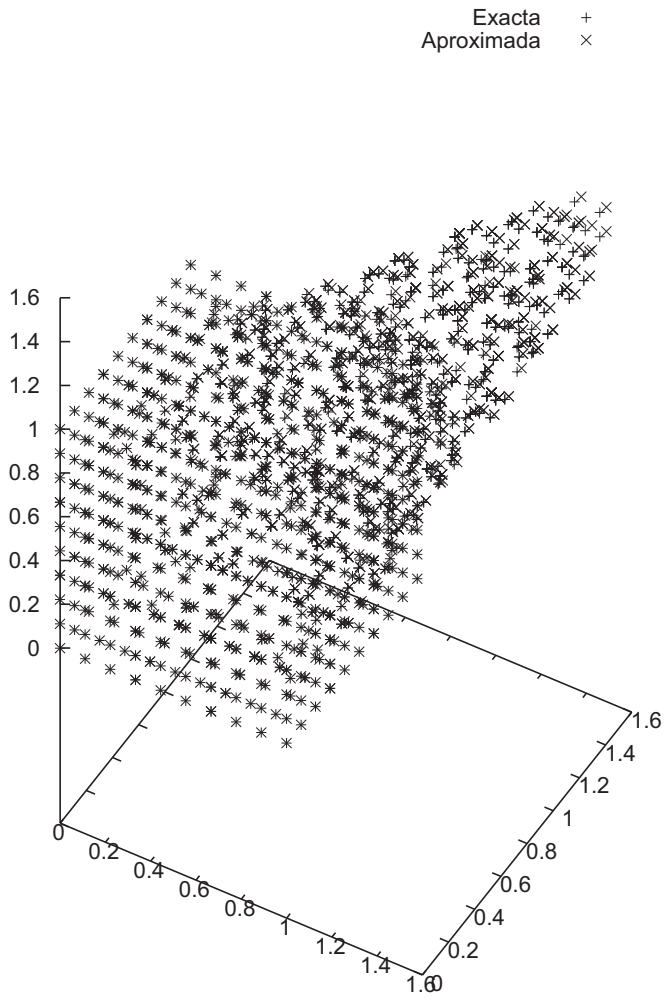

Figura 7.1: Deformación del cubo unitario

Figure 7: Deformación del cubo unitario

La familia de subdominios $\{\Omega_0, \cdots, \Omega_{E-1}\}$ se asume que la partición de $\bar{\Omega}$ que genera una frontera internaΓ. Cada uno de los subdominios esta discretizado usando FEM con funciones lineales como funciones base y que se denotan por $\underline{\psi}^{pi}$, $i = 1, 2, 3$.

La malla fina que se introdujo consiste de $193^3 = 7,189,057$cubos, lo cual da lugar a $194^3 = 7,301,384$ nodos originales. El número E de subdominios variará tomando sucesivamente los valores 8 , 27, 64 , 125 , 216 , 343 ,512 y así sucesivamente hasta 2,744 El número total de nodos derivados y su correspondiente número de grados de libertad estarán alrededor de 7×10^6 y $22{,}5 \times 10^6$, respectivamente.

Las restricciones que fueron impuestas consistieron en restricciones de continuidad en los nodos primales; en cada uno de los experimentos numéricos todos los nodos localizados en las aristas y en los vértices de la malla gruesa fueron tomados como nodos primales. De esta manera, el número total de nodos primales varió desde un mínimo de 583 a un máximo de 94,471. Cabe mencionar que estas condiciones garantizan que en cada uno de los experimentos numéricos la matriz $\underline{\underline{A}}$ será positiva definida y tendrá inversa.

Todos los códigos fueron desarrollados en C++ con MPI. Los cálculos fueron realizados en Miztli, la supercomputadora de la Universidad Nacional (UNAM), administrada por la DGTIC que es un cluster que consta de 314 nodos de procesamiento con 8 nucleos por nodo, con procesadores Intel Xeon Sandy Bridge E5-2670 a 2.6 GHz de velocidad.

Como se mostró en el análisis del sistema, la transmisión de información entre procesadores ocurre únicamente cuando se aplican los operadores de promedio a y a'.

En una primera versión del software dicho intercambio de información requería de la intervención de un maestro. Esto significa que había un procesador que recolectaba la información, la procesaba y la devolvÍa al resto de los procesadores lo cual ocasionaba un gasto de tiempo excesivo. Posteriormente se eliminó al maestro y la comunicación ahora se lleva a cabo entre subdominios vecinos.

Se presenta a continuación una tabla resumiendo los resultados numéricos.

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

Figura 7.2: Resultados Numéricos

Se hace notar que la eficiencia computacional es bastante alta, alcanzando un máximo de 96,6 %. Además, la eficiencia se incrementa conforme el número de procesadores también lo hace. Esta es una característica digna de reconocer en un software que intenta estar en la vanguardia como una herramienta para las grandes supercomputadoras que existen en la actualidad.

Capítulo 8

Conclusiones

Por medio de un método de discretización como FEM aplicado a la ecuación de Elastoestática se derivó su correspondiente sistema de ecuaciones al que se le siguió la metodología desarrollada de métodos de descomposición de dominio DVS, el cual con procedimiento sencillo permite desarrollar códigos altamente paralelizables para ese tipo de problemas. Los códigos así derivados son muy robustos. Además si se combina con técnicas de programación orientada a objetos son fácilmente construidos. En este trabajo de tesis se confirma la aplicabilidad de los algoritmos DVS, tanto de las fórmulas como de los procedimientos, a sistemas de Ecuaciones Diferenciales Parciales así como algunas otras características atractivas de dicha metodología.

De forma puntual este trabajo de tesis

- Contribuye al desarrollo de los métodos de discretización sin traslape y de la aproximación que introdujo I. Herrera y sus colaboradores [12]-[];
- Presenta un procedimiento para transformar una discretización con traslape en una sin traslape.
- Muestra que dicho método es aplicable a una sola ecuación diferencial parcial así como a sistemas de ecuaciones diferenciales parciales.
- Ilustra los procedimientos requeridos para la construcción de software basado en discretizaciones sin traslape, que en este caso particular este software esta adecuado para el tratamiento de la ecuación de elasticidad en el caso isotrópico.
- Presenta el software generado siguiendo la metodología. Este software ha sido probado numéricamente mostrando una la alta eficiencia como herramienta para la paralización.
- La conclusión principal es que la metodología DVS y las discretizaciones sin traslape son herramientas muy adecuadas para aprovechar el cómputo en paralelo para resolver las ecuaciones diferenciales parciales que aparecen en los sistemas de la ciencia y de la ingeniería.

Trabajo futuro

- Implementar el resto de los algoritmos DVS.
- Resolver el problema de elasticidad con otros coeficientes, en otras geometrías,
- Resolver otras ecuaciones con otros operadores diferenciales
- Incluir el tiempo
- Usar otros métodos de discretización

Bibliografía

[1] Herrera, I. & G.F. Pinder, "Mathematical Modelling in Science and Engineering: An axiomatic approach", John Wiley, 243p., 2012.

[2] PRESIDENT'S INFORMATION TECHNOLOGY ADVISOTY COMMITTEE: PITAC "Computational Science: Ensuring America's Competitiveness", Report to the President June 2005. 104 p. www.nitrd.gow/pitac

[3] DDM Organization, Proceedings of 22 International Conferences on Domain Decomposition Methods www.ddm.org, 1988-2014.

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

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

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

[7] Mandel J., Dohrmann C.R. and Tezaur R., *An algebraic theory for primal and dual substructuring methods by constraints*, Appl. Numer. Math., 54: 167-193, 2005.

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

[9] Mandel J. and Tezaur R. Convergence of a substructuring method with Lagrange multipliers. Numer. Math 76: 473-487, 1996.

[10] Farhat C., Lessoinne M. LeTallec P., Pierson K. and Rixen D. FETI-DP a dual-primal unified FETI method, Part I: A faster alternative to the two-level FETI method. Int. J. Numer. Methods Engrg. 50, pp 1523-1544, 2001.

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

[12] Herrera, I., de la Cruz L.M. and Rosas-Medina A. "Non Overlapping Discretization Methods for Partial, Differential Equations". NUMER METH PART D E, 30: 1427-1454, 2014, DOI 10.1002/num 21852. (Open source).

[13] Herrera, I. , Rosas-Medina A. "The Derived-Vector Space Framework and Four General Purposes Massively Parallel DDM Algorithms", EABE (Engineering Analysis with Boundary Elements), 37 pp-646-657, 2013.

[14] Carrillo-Ledesma A., Herrera I. , de la Cruz Luis Miguel. "Parallel Algorithms for Computational Models of Geophysical Systems". Geofísica Internacional, 52 (3) pp., 293-309, 2013.

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

[16] Herrera, I. "Theory of Differential Equations in Discontinuous Piecewise-Defined-Functions", NUMER METH PART D E, 23(3), pp 597-639, 2007. DOI 10.1002/num 20182.

[17] Herrera, I. "New Formulation of Iterative Substructuring Methods without Lagrange Multipliers: Neumann-Neumann and FETI", NUMER METH PART D E 24(3) pp 845-878, 2008 (Published on line Sep 17, 2007) DOI 10.1002/num 20293.

[18] Herrera, I.,Yates R. A. "The Multipliers-free Domain Decomposition Methods" NUMER. METH. PART D. E. 26: 874-905 July 2010 (DOI 10.1002/num. 20462)

[19] Herrera, I.,Yates R. A. "The Multipliers-Free Dual Primal Domain Decomposition Methods for Nonsymmetric Matrices" NUMER. METH. PART D. E. 27(5) pp. 1262-1289, 2011. DOI 10.1002/Num. 20581.

[20] Contreras I.,Herrera I. An Innovative Tool for Effectively Applying Highly Parallelized Hardware To Problems of Elasticity Geofísica Internacional, 55 (1) pp., 39-53, 2015.

[21] C. Farhat. A method of Finite Element Tearing and Interconnecting and its Parallel solution Algorithm, International Journal for Numerical Methods in Engineering, vol. 32, (1991)

[22] R. Tezaur,Analysis of Lagrange Multiplier based Domain Decomposition, Ph. D thesis, University of Colorado, (1998)

[23] A. Quarteroni, Numerical Mathematics, Springer Series in Computational Mathematics, Springer-Verlag, Berlin, (2007)

[24] G. H. Golub, Matrix Computation, Oxford : North oxford academic, (1983),473p.

[25] G. H. Golub, Scientific computing an introduction with parallel computing, Boston press, (1993),442P.

[26] B. Smith, P. Bjørstad, W. Gropp , Domain decomposition : parallel multilevel methods for elliptic partial differential equations, New York : Cambridge University Press, (1996), p224.

[27] B. Reddy, Functional analysis and boundary-value problems : An introductory treatment, New York : Longman : J. Wiley, (1986), 333 p.

[28] P. Ciarlet, Introduction to numericial algebra and optimisation, Cambridge University Press(1989), 428 p.

[29] L. Trefethen, D. Bau, Numerical Linear Algebra, Society for Industrial and Applied Mathematics, 1997, 299p.

[30] M. Gurtin, An introduction to continuum mechanics, New York Academic,1981, 129p.

[31] R. Trobec, M. Vajteršic, P. Zinterhof, Parallel Computing Numerics, Applications, and Trends, Springer, 2009.

[32] I. Herrera, L. de La Cruz. Minisympusium MS 10 "Non-overlapping Discretization Methods and How to Achieve the DDM-paradigm". Presentación oral DDM 22 Lugano, Suiza. 2013.

[33] C. Fontela, UML: Modelado de Software para profesionales. Buenos Aires Alfaomega, 2011.

[34] http://en.wikipedia.org/wiki/Poisson's_ratio

Apéndice A

Algoritmos

Hacemos notar que muchos otros algoritmos , además de los descritos hasta aquí, pueden ser definidos reemplazando las condiciones Dirichlet y/o Neumann sobre Γ por unas más generales que involucran combinaciones lineales de u y de las derivadas normales. Los métodos que a continuación se presentan pueden ser derivados sólo algebraicamente usando factorizaciones válidas del sistema, sin ninguna referencia al operador continuo subyacente, a las trazas ni a las derivadas normales. Sin embargo, empleando el marco funcional no solamente se da una interpretación a los procedimientos de iteración sino también dan una preparación para el análisis de convergencia. Los próximos algoritmos además involucran precondicionadores para resolver ambos sistemas. Aunque también se pueden resolver con el método de Richardson o con algún método de Gradiente Conjugado sin precondicionamiento.

La evaluación de Sv involucra la solución de un problema tipo Dirichlet en cada subdominio, mientras que la evaluación de Fv requiere la solución de un problema tipo Neumann en cada subdominio.

A.0.1. Algoritmo Dirichlet-Neumann

En el subdominio$\Omega_{\ 1}$ se resuelve un problema de Dirichlet con un vector inicial u_Γ^0 y con condiciones Dirichlet en $\partial\Omega_i|\Gamma$ Se obtiene un valor para u Se resuelve en dos subregiones en dos pasos por iteración, cada paso por subregión de la siguiente manera

En el problema Neumann primero se obtienen los valores de λ y después de u.

Algorithm 1 Procedimiento general

Definir condiciones de Frontera en cada uno de los subdominios
Determinar las condiciones de transmisibilidad
Dar una fórmula para representarlas y actualizarlas cuando se este aplicando un método iterativo
Despejar alguna de las condiciones de transmisibilidad
Usar algunas expresiones de los despejes (básicamente el complemento de Schur o su inversa) como precondicionadores en las ecuaciones obtenidas de los despejes de las condiciones de transmisibilidad.

Algorithm 2 Algoritmo Dirichlet-Neumann

1.Se da una suposición inicial para u_Γ^0
2.Se resuelve enΩ_1 tomando u_Γ^0 enΓ
3. Se usan los valores para un problema con condiciones Neumann-Dirichlet paraΩ_2 con condiciones Neumann sobre Γ con los valores obtenidos enΩ_1 del paso anterior y con condiciones Dirichlet en $\partial\Omega_2|\Gamma$
4. El vector obtenido u_Γ^1 será la traza de la solución enΩ_2, es decir, una combinación lineal de esa traza y u_Γ^0 tomando un valor θ de relajamiento.

Número de condicionamiento del Dirichlet-Neumann

A.1. Algoritmo Neumann-Neumann

El algoritmo básico puede ser descrito como

Algorithm 3 Algoritmo Neumann-Neumann

1.Partiendo de una suposición inicial para u_Γ^0
2.Se resuelve en la subregiónΩ_i un problema Dirichlet con los valores de u_Γ^0 enΓ.
3. Se toman las dos soluciones u_i y su diferencia se toma como la derivada normal y con esos valores se resuelve un problema tipo Neumann en cada subregiónΩ_i.
4. Los valores sobre Γ de las soluciones de esos problemas tipo Neumann se ocupan para corregir u_Γ^0 y obtener la nueva iteración u_Γ^1

A.2. Algoritmo Dirichlet-Dirichlet o FETI

Algorithm 4 Algoritmo Neumann-Neumann

1.Comenzamos con una suposición inicial en el flujo λ_Γ^0 sobreΓ
2.Resolvemos problemas Neumann en cadaΩ_i con λ_Γ^0 sobreΓ
3. La diferencia de la traza de las soluciones de Neumann en cada subregión sobreΓ
4. Los valores sobre Γ de las derivadas normales de las soluciones de esos problemas Dirichlet son empleadas para corregir λ_Γ^0 y encontrar λ_Γ^1

Apéndice B

Complemento de Schur

B.1. Sistema de ecuaciones

Sea un sistema de ecuaciones en forma matricial obtenido al discretizar una ecuación diferencial parcial

$$A = \begin{bmatrix} A_{\Pi\Pi} & A_{\Pi\Delta} \\ A_{\Delta\Pi} & A_{\Delta\Delta} \end{bmatrix} \begin{bmatrix} u_\Pi \\ u_\Delta \end{bmatrix} = \begin{bmatrix} f_\Pi \\ f_\Delta \end{bmatrix}$$

que como sistema de ecuaciones tiene la forma

$$A_{\Pi\Pi} u_\Pi + A_{\Pi\Delta} u_\Delta = f_\Pi \tag{B.1.1}$$
$$A_{\Delta\Pi} u_\Pi + A_{\Delta\Delta} u_\Delta = f_\Delta \tag{B.1.2}$$

Despejando u_Π de (B.1.1) y sustituyendo en (B.1.2)

$$u_\Pi = A_{\Pi\Pi}^{-1}(f_\Pi - A_{\Pi\Delta} u_\Delta) \tag{B.1.3}$$

obtenemos

$$(A_{\Delta\Delta} - A_{\Delta\Pi} A_{\Pi\Pi}^{-1} A_{\Pi\Delta}) u_\Delta = f_\Delta - A_{\Delta\Pi} A_{\Pi\Pi}^{-1} f_\Pi$$

donde

$$S = A_{\Delta\Delta} - A_{\Delta\Pi} A_{\Pi\Pi}^{-1} A_{\Pi\Delta}$$

es conocido como el complemento de Schur.

B.2. Formulación convencional

Dado que la matriz A se obtiene de un método de discretización podemos descomponer dicha matriz en bloques donde se identifiquen los subdominios y los nodos que estan involucrados. de esta manera el sistema de ecuaciones queda representado como

$$\begin{bmatrix} A^1_{\Pi\Pi} & & & & A^1_{\Pi\Delta} \\ & A^2_{\Pi\Pi} & & & A^2_{\Pi\Delta} \\ & & \ddots & & \vdots \\ & & & A^n_{\Pi\Pi} & \\ A^1_{\Delta\Pi} & A^2_{\Delta\Pi} & \dots & & A_{\Delta\Delta} \end{bmatrix} \begin{bmatrix} u_\Pi \\ \\ \\ \\ u_\Delta \end{bmatrix} = \begin{bmatrix} f_\Pi \\ \\ \\ \\ f_\Delta \end{bmatrix}$$

Dicho sistema se puede resolver obteniendo el complemento de Schur de forma tal que

$$(A_{\Delta\Delta} - \sum_{i=1}^{n} A^{(i)}_{\Delta\Pi} A^{(i)-1}_{\Pi\Pi} A^{(i)}_{\Pi\Delta}) u_\Delta = f_\Delta - \sum_{i=1}^{n} A^{(i)}_{\Delta\Pi} A^{(i)}_{\Pi\Pi} f^{(i)}_\Pi \quad \text{(B.2.1)}$$

Donde identificamos

$$\underline{\underline{S}} = (A_{\Delta\Delta} - \sum_{i=1}^{n} A^{(i)}_{\Delta\Pi} A^{(i)-1}_{\Pi\Pi} A^{(i)}_{\Pi\Delta})$$

como el complemento de Schur.

B.3. Formulación con funciones discontinuas

Separamos la matriz original A en dos matrices

$$\underline{\underline{A}} = \underline{\underline{L}} + \underline{\underline{R}}$$

tal que

$$\underline{\underline{L}} = \begin{bmatrix} A_{\Pi\Pi} & A_{\Pi\Delta} \\ 0 & 0 \end{bmatrix} R = \begin{bmatrix} 0 & 0 \\ A_{\Delta\Pi} & A_{\Delta\Delta} \end{bmatrix}$$

Así, dado el sistema (B.2.1) y una $v \in D$, es decir, una armónica de la ecuación diferencial entonces podemos expresar dicha ecuación como

$$\underline{\underline{A}}v = \underline{\underline{R}}v = \begin{bmatrix} 0 \\ \underline{\underline{S}}v_\Delta \end{bmatrix}$$

Sea entonces $\bar{\underline{u}} \in \bar{D}(\bar{\Omega})$ y $\underline{u}_P \in \tilde{D}(\bar{\Omega})$ tal que

$$\bar{\underline{u}} = \underline{u} + \underline{u}_P$$

y se satisfacen las siguientes condiciones

$$\underline{\underline{L}}\underline{u}_P = \underline{f}_\Pi \tag{B.3.1}$$
$$\underline{\underline{j}}\underline{u}_P = 0 \tag{B.3.2}$$

mientras que

$$\underline{\underline{L}}\underline{u} = 0 \tag{B.3.3}$$
$$[[\underline{\underline{R}}]]\underline{u} = \underline{f}_\Delta - [[\underline{\underline{R}}]]\underline{u}_P \tag{B.3.4}$$
$$\dot{\underline{\underline{R}}}^T \underline{u}_P \tag{B.3.5}$$

Lo anterior no garantiza que la $\underline{u}_P$ sea única a menos de que escojamos $(\underline{u}_P)_\Delta = 0$, en este caso $\underline{u}_P$ sí es solución única de un problema Dirichlet.

El sistema a resolver esta dado por

$$\underline{\underline{aS}}\underline{u}_\Delta = \underline{f}_\Delta - \underline{\underline{a}}A_{\Delta\Pi}A_{\Pi\Pi}^{-1}\underline{f}_\Pi \tag{B.3.6}$$

B.4. Solución del sistema en paralelo usando CGM

Dada la estructura de la matriz A se puede implementar el algoritmo del gradiente conjugado para que se resuelva de manera independiente cada uno de los bloques que conforman la matriz. Es decir, el sistema (B.2.1), o el sistema (B.3.6) para el caso con funciones discontinuas, puede ser resuelto en cada uno de sus bloques de manera independiente y después sumarse para aplicar una iteración del gradiente conjugado.

B.4.1. Formulación Convencional

Podemos definir el sistema (B.2.1) como

$$Su_\Delta = g_\Delta \tag{B.4.1}$$

donde

$$\underline{\underline{S}} = \sum_{i=0}^{n} S^{(i)} \tag{B.4.2}$$

$$\underline{\underline{S}}^{(i)} = A_{\Delta\Delta} - A^{(i)}_{\Delta\Pi} A^{(i)-1}_{\Pi\Pi} A^{(i)}_{\Pi\Delta} \tag{B.4.3}$$

$$g_\Delta = \underline{f}^{(i)}_\Delta - A^{(i)}_{\Delta\Pi} A^{(i)-1}_{\Pi\Pi} \underline{f}^{(i)}_\Pi \tag{B.4.4}$$

Por la estructura de las matrices es posible ejecutar operaciones de forma independiente, es por eso que se puede asignar cada cálculo de $S^{(i)}$ a un procedimiento ajeno a los resultados de los otros bloques, concentrandose la información en un procesador que calcula u_Δ con base en la información proporcionada por todos los procedimientos ejecutándose en paralelo. Es de esta manera que no es necesario construir S de forma explícita ya que las operaciones por bloque, $S^{(i)}$, son suficientes para calcular la parte correspondiente del vector u_Δ el cual se arma, de la misma forma que las matrices, por bloque $u^{(i)}_\Delta$.

Se hace notar de manera particular, que la aplicación de la inversa de $A^{(i)-1}_{\Pi\Pi}$ a un vector corresponde a la solución de un problema de Dirichlet en $Omega_i$ o a un problema con condiciones de frontera Dirichlet en $\partial\Omega_i \cap \Gamma$ y con datos Neumann homogeneos en $\partial\Omega_i \cap \partial\Omega_N$

De la misma forma no es necesario calcular $A^{(i)-1}_{\Pi\Pi}$ ya que solo requerimos el cálculo

$$A^{(i)-1}_{\Pi\Pi} A^{(i)}_{\Pi\Delta} u_\Delta \tag{B.4.5}$$

esto es, si definimos

$$x_\Delta = A^{(i)-1}_{\Pi\Pi} x_\Pi \tag{B.4.6}$$

donde

$$x_\Pi = A^{(i)}_{\Pi\Delta} u_\Delta$$

Algorithm 5 Gradiente Conjugado

$u_\Delta = v^0$
$b = f_\Delta - \sum_{i=1}^{n} A_{\Delta\Pi}^{(i)} A_{\Pi\Pi}^{(i)} f_\Pi^{(i)}$
$Au_\Delta = (A_{\Delta\Delta} - \sum_{i=1}^{n} A_{\Delta\Pi}^{(i)} A_{\Pi\Pi}^{(i)-1} A_{\Pi\Delta}^{(i)}) u_\Delta$
$r = b - Au_\Delta$
$p = r$
$\mu = r \cdot r$
while $\mu \geq \epsilon$ **do**
  $u_\Delta = (A_{\Delta\Delta} - \sum_{i=1}^{n} A_{\Delta\Pi}^{(i)} A_{\Pi\Pi}^{(i)-1} A_{\Pi\Delta}^{(i)}) p^{(i)}$
  $\lambda = p \cdot u_\Delta$
  $\alpha = \frac{\mu}{\lambda}$
  $u_\Delta = u_\Delta + \alpha p$
  $r = r - \alpha v$
  $\mu' = r \cdot r$
  $\beta = \frac{\mu'}{\mu}$
  $p = r + \beta p$
  $\mu = \mu'$
end while

entonces podemos calcular

$$A_{\Pi\Pi}^{(i)} x_\Delta = A_{\Pi\Pi}^{(i)} A_{\Pi\Pi}^{(i)-1} x_\Pi$$

es decir, bastará resolver el sistema

$$A_{\Pi\Pi}^{(i)} x_\Delta = x_\Pi$$

y posteriormente hacer las sustituciones correspondientes para obtener (B.4.14).

Una vez resuelto el sistema (B.2.1) sólo basta obtener el valor de u_Π, para ello se sustituye en (B.1.3) para obtener los correspondientes $u_\Pi^{(i)}$. Dado que en esta ecuación también interviene $A_{\Pi\Pi}^{(i)-1}$ procedemos de una manera semejante a el procedimiento para (B.4.14). Esto es

$$x_\delta = f_\Pi - A_{\Pi\Delta} u_\Delta \tag{B.4.7}$$
$$u_\Pi^{(i)} = A_{\Pi\Pi}^{(i)-1} x_\delta \tag{B.4.8}$$

entonces sólo requerimos solucionar

$$A_{\Pi\Pi}^{(i)} u^{(i)\Pi} = x_d elta \tag{B.4.9}$$

para posteriormente sustituir y obtener los valores de u_Π que son los que buscamos.

B.4.2. Formulación con funciones discontinuas

La forma de proceder es semejante sólo se sustituye (B.3.6) como

$$\underline{\underline{a}} S u_\Delta = g_\Delta \tag{B.4.10}$$

donde

$$\underline{\underline{S}} = \sum_{i=0}^{n} S^{(i)} \tag{B.4.11}$$

$$\underline{\underline{S}}^{(i)} = A_{\Delta\Delta} - A^{(i)}_{\Delta\Pi} A^{(i)-1}_{\Pi\Pi} A^{(i)}_{\Pi\Delta} \tag{B.4.12}$$

$$g_\Delta = \underline{f}^{(i)}_\Delta - \underline{\underline{a}} A^{(i)}_{\Delta\Pi} A^{(i)-1}_{\Pi\Pi} \underline{f}^{(i)}_\Pi \tag{B.4.13}$$

Algorithm 6 Gradiente Conjugado

$u_\Delta = v^0$
$b = f_\Delta - \sum_{i=1}^{n} \underline{\underline{a}} A^{(i)}_{\Delta\Pi} A^{(i)}_{\Pi\Pi} f^{(i)}_\Pi$
$Au_\Delta = (A_{\Delta\Delta} - \sum_{i=1}^{n} \underline{\underline{a}} A^{(i)}_{\Delta\Pi} A^{(i)-1}_{\Pi\Pi} A^{(i)}_{\Pi\Delta}) u_\Delta$
$r = b - Au_\Delta$
$p = r$
$\mu = r \cdot r$
while $\mu \geq \epsilon$ **do**
  $u_\Delta = (A_{\Delta\Delta} - \sum_{i=1}^{n} \underline{\underline{a}} A^{(i)}_{\Delta\Pi} A^{(i)-1}_{\Pi\Pi} A^{(i)}_{\Pi\Delta}) p^{(i)}$
  $\lambda = p \cdot u_\Delta$
  $\alpha = \frac{\mu}{\lambda}$
  $u_\Delta = u_\Delta + \alpha p$
  $r = r - \alpha v$
  $\mu' = r \cdot r$
  $\beta = \frac{\mu'}{\mu}$
  $p = r + \beta p$
  $\mu = \mu'$
end while

De la misma forma no es necesario calcular $A^{(i)-1}_{\Pi\Pi}$ ya que solo requerimos el cálculo

$$A^{(i)-1}_{\Pi\Pi} A^{(i)}_{\Pi\Delta} u_\Delta \tag{B.4.14}$$

esto es, si definimos

$$x_\Delta = A_{\Pi\Pi}^{(i)-1} x_\Pi \tag{B.4.15}$$

donde

$$x_\Pi = A_{\Pi\Delta}^{(i)} u_\Delta$$

entonces podemos calcular

$$A_{\Pi\Pi}^{(i)} x_\Delta = A_{\Pi\Pi}^{(i)} A_{\Pi\Pi}^{(i)-1} x_\Pi$$

es decir, bastará resolver el sistema

$$A_{\Pi\Pi}^{(i)} x_\Delta = x_\Pi$$

y posteriormente hacer las sustituciones correspondientes para obtener (B.4.14).

Una vez resuelto el sistema (B.2.1) sólo basta obtener el valor de u_Π, para ello se sustituye en (B.1.3) para obtener los correspondientes $u_\Pi^{(i)}$. Dado que en esta ecuación también interviene $A_{\Pi\Pi}^{(i)-1}$ procedemos de una manera semejante a el procedimiento para (B.4.14). Esto es

$$x_\delta = f_\Pi - A_{\Pi\Delta} u_\Delta \tag{B.4.16}$$

$$u_\Pi^{(i)} = A_{\Pi\Pi}^{(i)-1} x_\delta \tag{B.4.17}$$

entonces sólo requerimos solucionar

$$A_{\Pi\Pi}^{(i)} u^{(i)\Pi} = x_\delta \tag{B.4.18}$$

para posteriormente sustituir y obtener los valores de u_Π que son los que buscamos.