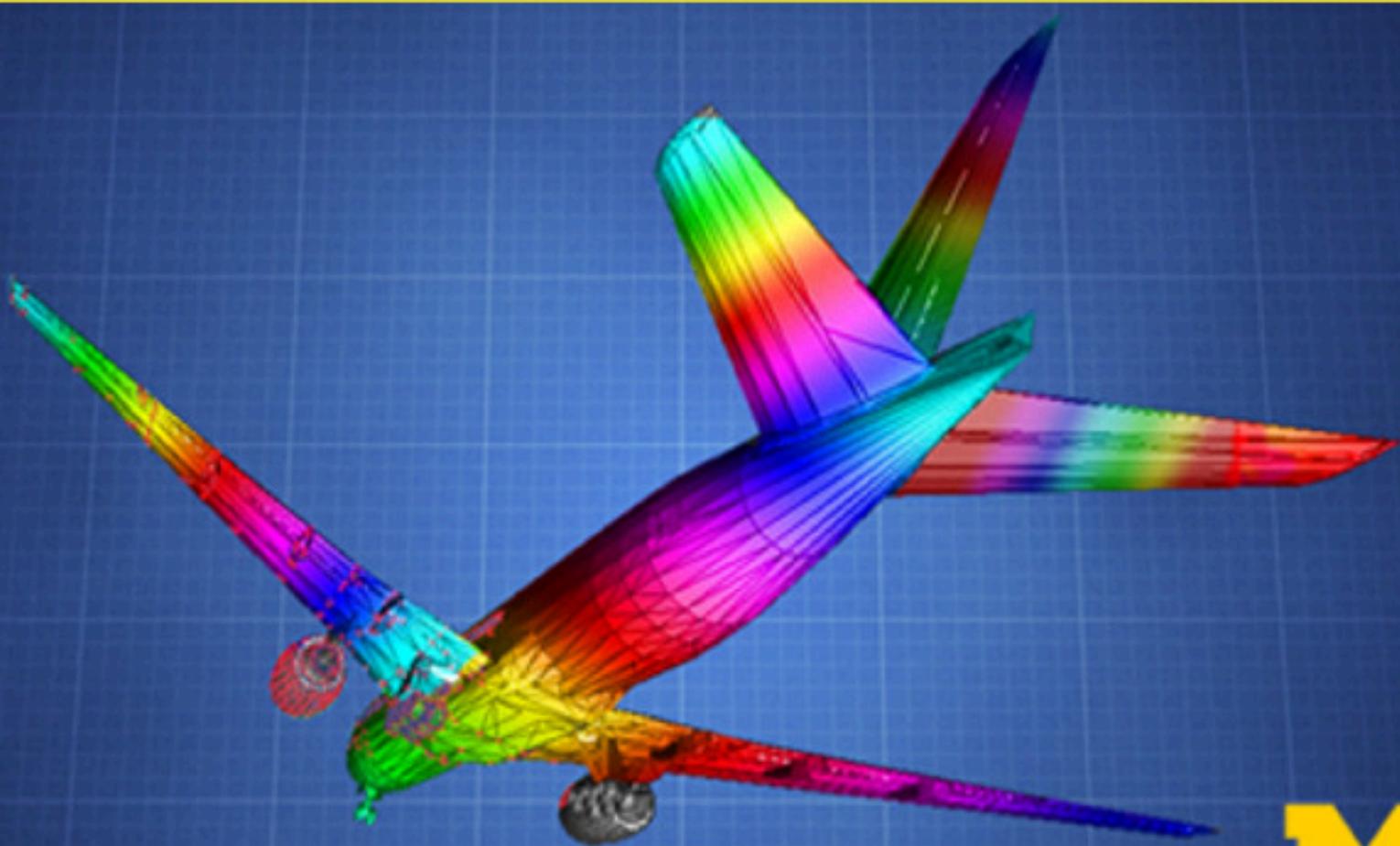


INTRODUCCIÓN A LOS MÉTODOS DE DESCOMPOSICIÓN DE DOMINIO



M

Antonio Carrillo Sedesma

Introducción a los Métodos de Descomposición de Dominio

Antonio Carrillo Ledesma
Facultad de Ciencias, UNAM

<http://academicos.fciencias.unam.mx/antoniocarrillo>

La última versión de este trabajo se puede descargar de la página:

<https://sites.google.com/ciencias.unam.mx/acl/en-desarrollo>

<http://132.248.181.216/acl/EnDesarrollo.html>

2025, Versión 1.0 α ¹

¹El presente trabajo está licenciado bajo un esquema Creative Commons Atribución CompartirIgual (CC-BY-SA) 4.0 Internacional. Los textos que componen el presente trabajo se publican bajo formas de licenciamiento que permiten la copia, la redistribución y la realización de obras derivadas siempre y cuando éstas se distribuyan bajo las mismas licencias libres y se cite la fuente. ¡Copia este libro! ... Compartir no es delito.

Índice

1	Introducción	8
1.1	Antecedentes	8
1.2	Métodos de Descomposición de Dominio	10
1.3	El Método de Descomposición de Dominio en el Espacio de Vectores Derivados	13
1.4	Objetivos del trabajo	16
1.5	Infraestructura Computacional Usada	17
1.6	Organización del Trabajo	19
1.7	Agradecimientos	20
2	Sistemas Continuos y sus Modelos	22
2.1	Los Modelos	22
2.1.1	Física Microscópica y Física Macroscópica	23
2.2	Cinemática de los Modelos de Sistemas Continuos	23
2.2.1	Propiedades Intensivas y sus Representaciones	25
2.2.2	Propiedades Extensivas	28
2.2.3	Balance de Propiedades Extensivas e Intensivas	29
2.3	Ejemplos de Modelos	33
3	Ecuaciones Diferenciales Parciales	37
3.1	Clasificación	37
3.2	Condiciones Iniciales y de Frontera	41
3.3	Modelos Completos	42
4	Análisis Funcional y Problemas Variacionales	44
4.1	Operador Lineal Elíptico	44
4.2	Espacios de Sobolev	46
4.2.1	Trazas de una Función en $H^m(\Omega)$	49
4.2.2	Espacios $H_0^m(\Omega)$	50
4.2.3	Espacios $H(\text{div}, \Omega)$	52
4.3	Fórmulas de Green y Problemas Adjuntos	54
4.4	Adjuntos Formales para Sistemas de Ecuaciones	63
4.5	Problemas Variacionales con Valor en la Frontera	68

5	Métodos de Solución Aproximada para EDP	73
5.1	Método Galerkin	74
5.1.1	El Método de Residuos Pesados	77
5.1.2	Método de Elemento Finito	78
5.2	Método de Penalización Interior	82
5.3	Método Galerkin Discontinuo	83
5.3.1	Generalización del Método Galerkin Discontinuo	84
5.3.2	Flujos Numéricos Independientes de ∇u_h	87
5.3.3	Flujos Numéricos Independientes de σ_h	90
5.3.4	Distintos tipos de Métodos Galerkin Discontinuo	92
5.4	Método Discontinuo Enriquecido	93
5.4.1	Formulación Variacional Híbrida con Continuidad Débil	96
5.4.2	Formulación Débil	97
5.4.3	Aproximación Galerkin	98
5.4.4	Condensación Estática	99
5.4.5	Aproximación de los Multiplicadores de Lagrange	101
5.4.6	Condiciones de Frontera Neumann y Robin	102
5.5	Métodos de Elementos Finitos Mixtos e Híbridos	104
5.5.1	Métodos Mixtos	105
5.5.2	Métodos Híbridos	114
6	Método FETI	120
6.1	Conceptos Básicos	120
6.1.1	Una Ecuación para el Flujo Usando el Complemento de Schur	124
6.1.2	Extensión Armónica Discreta	126
6.2	One-Level FETI	126
6.2.1	Algoritmos en Dos Subdominios	126
6.2.2	Algoritmos en Múltiples Subdominios	130
6.2.3	El Algoritmo One-Level FETI Simplificado	138
6.3	Dual-Primal FETI	141
6.4	Variantes para la Implementación Numérica	146
6.4.1	Implementación de la Matriz \underline{J}	147
6.4.2	Cálculo de la Matriz \underline{S}	150
6.4.3	Cálculo de la Matriz \underline{S}^{-1}	152
6.4.4	Cálculo de los Nodos Interiores	152
6.5	Implementación Computacional	158

7	Funciones Definidas por Tramos	169
7.1	Espacios de Sobolev de Funciones Definidas por Tramos	173
7.2	Fórmulas Green-Herrera	175
7.3	Formulaciones Variacionales con Valor en la Frontera con Saltos Prescritos	180
8	Método de Trefftz	183
8.1	Conceptos Básicos	185
8.1.1	Condiciones de Poincaré-Steklov	187
8.2	Método Indirecto de Trefftz-Herrera	189
8.3	Método directo de Steklov-Poincaré	193
9	Métodos de Funciones Discontinuas Definidas por Tramos	198
9.1	Algoritmos a Nivel Continuo	199
9.1.1	Algoritmo Neumann-Neumann	199
9.1.2	Algoritmo Dirichlet-Dirichlet	201
9.2	Discretización Axiomática	203
9.3	Esquema General	206
9.4	Procedimiento para Evaluar la Transformación de Componentes	213
9.5	Métodos Dual-Primal	217
10	Unificación y Simplificación de los Métodos Dual-Primal de Descomposición de Dominio	219
10.1	Espacio Dual-Primal	219
10.2	Espacio de Vectores	221
10.3	El Producto Interior Euclidiano	222
10.4	Subespacios de Vectores: Las Matrices de Promedio y de Salto	224
10.5	El Subespacio Dual-Primal	226
10.6	La Formulaci3n Matricial Discontinua Libre de Multiplicadores de Lagrange	228
10.7	Construcci3n de la Matriz \underline{A}	231
11	Formulaciones Dirichlet-Dirichlet y Neumann-Neumann a Nivel Continuo	233
11.1	El Problema no Precondicionado Dirichlet-Dirichlet	234
11.2	El Problema no Precondicionado Neumann-Neumann	236

12 Marco Teórico del Espacio de Vectores Derivados	238
12.1 Discretización del Dominio para los Métodos Duales y Primal	239
12.2 Una Verdadera Descomposición de Dominio sin Traslapes . . .	241
12.3 El Problema Original	245
12.4 El Espacio de Vectores Derivado	248
12.5 Discretización Partiendo de la Construcción de la Matriz \underline{A}^t .	251
12.6 El Problema General con Restricciones	256
13 Formulaciones Dirichlet-Dirichlet y Neumann-Neumann en el Marco del Espacio de Vectores Derivados	258
13.1 Algoritmos no Precondicionados	259
13.1.1 Algoritmo del Complemento de Schur	259
13.1.2 Formulación Dual del Problema Neumann-Neumann .	260
13.1.3 Formulación Primal del Problema Neumann-Neumann	261
13.1.4 Segunda Formulación Dual del Problema Neumann-Neumann	262
13.2 Algoritmos Precondicionados	263
13.2.1 Versión DVS del Algoritmo BDDC	263
13.2.2 Versión DVS del Algoritmo FETI-DP	265
13.2.3 Formulación Primal Precondicionada del Problema Neumann-Neumann	266
13.2.4 Segunda Formulación Dual Precondicionada del Problema Neumann-Neumann	268
13.3 El Operador de Steklov-Poincaré	270
14 Los Métodos FETI-DP y BDDC en el Marco de DVS	276
14.1 El Método FETI-DP en el Marco de DVS	277
14.2 El Método BDDC en el Marco de DVS	280
14.3 Comparaciones	283
15 Formulación Numérica de los Métodos DVS	285
15.1 Discretización de los Métodos Partiendo de la Formulación Local	286
15.2 Formulación Operacional de los Métodos DVS	287
15.3 Implementación Numérica de DVS	291
15.3.1 Implementación para Matrices Simétricas	292
15.3.2 Implementación para Matrices no Simétricas e Indefinidas	293
15.4 Evaluación de los Operadores Virtuales \underline{S} y \underline{S}^{-1}	294

16 Implementación Computacional de DVS	296
16.1 Esquema Maestro-Esclavo como una Forma de Implementación	296
16.2 Análisis, Diseño y Programación Orientada a Objetos	297
16.2.1 Implementación Secuencial en C++	299
16.2.2 Implementación Paralela en C++ Usando MPI	300
16.3 Alcances y Limitaciones del Esquema Maestro-Esclavo	302
16.4 Afectación del Rendimiento al Refinar la Descomposición	308
16.5 Opciones para Soportar una Descomposición Fina del Dominio	311
16.6 Otras Opciones de Paralelización	313
17 Análisis y Discusión de Resultados	316
17.1 Análisis de Rendimiento para Problemas Simétricos y no Simétricos	317
17.2 Análisis de Rendimiento para Problemas Indefinidos	321
17.3 Análisis de Rendimiento para Problemas de Advección-Difusión	322
17.4 Análisis de Rendimiento para Sistemas de Ecuaciones	328
17.5 Análisis de Rendimiento en Equipos Paralelos	329
17.5.1 Selección Óptima de una Descomposición del Dominio	329
17.5.2 Análisis de Rendimiento Usando Métricas	334
17.5.3 Escalabilidad del Esquema DVS	338
17.6 Criterios Integrales para Evaluar el Esquema DVS	339
18 Conclusiones y Trabajo Futuro	344
18.1 Conclusiones	345
18.2 Trabajo Futuro	348
19 Apéndice A: Consideraciones Sobre la Formulación Numérica y su Implementación Computacional	350
19.1 Matrices Virtuales y Susceptibles de Construir	350
19.2 Evaluación de la Matriz $\underline{\underline{S}}$ con Nodos Primitives Definidos	351
19.3 Evaluación de la Matriz $\underline{\underline{S}}^{-1}$ con Nodos Primitives Definidos	354
19.4 Cálculo de los Nodos Interiores	355
19.5 Descomposición de Schur sin Nodos Primitives	355
19.6 DVS para Ecuaciones Escalares y Vectoriales	356
19.7 Método de Descomposición de Dominio de Subestructuración	361

20 Apéndice B: Estructura Óptima de las Matrices en su Implementación Computacional	372
20.1 Almacenamiento en la Memoria RAM	373
20.2 Matrices Bandadas	377
20.3 Matrices Dispersas	379
20.4 Multiplicación Matriz-Vector	381
21 Apéndice C: Solución de Grandes Sistemas de Ecuaciones Lineales	383
21.1 Métodos Directos	384
21.1.1 Eliminación Gaussiana	384
21.1.2 Factorización LU	385
21.1.3 Factorización Cholesky	386
21.2 Métodos Iterativos	387
21.2.1 Jacobi	389
21.2.2 Gauss-Seidel	390
21.2.3 Richardson	390
21.2.4 Relajación Sucesiva	391
21.2.5 Método de Gradiente Conjugado	392
21.2.6 Gradiente Conjugado Precondicionado	394
21.2.7 Método Residual Mínimo Generalizado	397
21.3 Algunos Ejemplos	399
21.3.1 Usando Python	401
21.3.2 Usando C++	404
21.4 Cómputo Paralelo	405
21.5 Nuestra Implementación	409
22 Apéndice D: El Cómputo en Paralelo	412
22.1 Computadoras Actuales	416
22.2 ¿Que es Computación Paralela?	454
22.3 Clasificación Clásica de Flynn	461
22.4 Categorías de Computadoras Paralelas	464
22.4.1 Equipo Paralelo de Memoria Compartida	464
22.4.2 Equipo Paralelo de Memoria Distribuida	468
22.4.3 Equipo Paralelo de Memoria Compartida-Distribuida	469
22.4.4 Cómputo Paralelo en Multihilos	475
22.4.5 Cómputo Paralelo en CUDA	476
22.5 Métricas de Desempeño	481

22.6 Programación de Cómputo de Alto Rendimiento	486
22.6.1 Programando con OpenMP para Memoria Compartida	488
22.6.2 Programando con MPI para Memoria Distribuida . . .	491
22.6.3 Esquema de Paralelización Principal-Subordinados . .	496
22.6.4 Opciones de Paralelización Híbridas	498
22.7 Programando Desde la Nube	500
23 Apéndice E: Aritmética de Punto Flotante	502
23.1 Errores de Redondeo y de Aritmética	515
23.2 Trabajando con Punto Flotante	518
23.3 Aritmética de Baja Precisión	536
24 Apéndice F: Método de Elementos Finitos	540
24.1 Triangulación	541
24.2 Generar Mallas con GMSH	542
24.3 Interpolación para el Método de Elementos Finitos	549
24.4 Discretización en 2D Usando Rectángulos	549
24.5 Discretización en 2D Usando Triángulos	555
25 Apéndice G: Nociones de Álgebra Lineal	561
25.1 Nociones de Álgebra Lineal	561
25.2 σ -Álgebra y Espacios Medibles	562
25.3 Espacios L^p	564
25.4 Distribuciones	565
26 Bibliografía	570

1 Introducción

Los sistemas continuos -sistemas físicos macroscópicos (véase [1])- , tales como los yacimientos petroleros, la atmósfera, los campos electromagnéticos, los océanos, el aparato circulatorio de los seres humanos, la corteza terrestre y muchos otros sistemas de interés en Ciencia y en Ingeniería, al modelarse, estos contienen un gran número de grados de libertad¹.

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

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

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

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

1.1 Antecedentes

La solución numérica de ecuaciones diferenciales parciales por los esquemas tradicionales -tipo Diferencias Finitas, Volumen Finito y Elemento Finito-

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

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

El desarrollo de métodos numéricos para sistemas algebraicos grandes, es central en el desarrollo de códigos eficientes para la solución de problemas en Ciencia e Ingeniería, actualmente cuando se necesita implementar una solución computacional, se dispone de bibliotecas optimizadas² para la solución de sistemas lineales que pueden correr en ambientes secuenciales y/o paralelos. Estas bibliotecas implementan métodos algebraicos robustos para muchos problemas prácticos, pero sus discretizaciones no pueden ser construidas por sólo técnicas algebraicas simples, tales como aproximaciones a la inversa o factorización incompleta.

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

Para hacer eficiente la solución de sistemas de ecuaciones diferenciales parciales, se introdujeron los métodos de descomposición de dominio que toman en cuenta la ecuación diferencial parcial y su discretización, permitiendo una alta eficiencia computacional en diversas arquitecturas paralelas (véase [6] y [8]). La idea básica detrás de los métodos de descomposición de dominio es que en lugar de resolver un enorme problema sobre un dominio, puede ser conveniente -o necesario- resolver múltiples problemas de tamaño menor so-

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

bre un solo subdominio un cierto número de veces. Mucho del trabajo en la descomposición de dominio se relaciona con la selección de subproblemas que aseguren que la razón de convergencia del nuevo método iterativo sea rápida. En otras palabras, los métodos de descomposición de dominio proveen preconditionadores a priori que puedan acelerarse por métodos en el espacio de Krylov (véase [19]).

1.2 Métodos de Descomposición de Dominio

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

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

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

Desde hace ya algún tiempo, la comunidad internacional³ inicio el estudio

³Ello se refleja en las más de 19 conferencias internacionales de Métodos Descomposición

intensivo de los métodos de descomposición de dominio, la atención se ha desplazado (véase [9]) de los métodos con traslape en el dominio (véase [73]) a los métodos sin traslape en el dominio (véase [56], [57], [67], [68] y [69]), ya que estos últimos son más efectivos para una gran variedad de problemas de la Ciencia y la Ingeniería.

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

Tomando en cuenta que los métodos de descomposición de dominio sin traslape son fácilmente implementados para su uso en computadoras paralelas mediante técnicas de programación orientada a objetos -porqué el algoritmo del método es paralelo-, además, con los continuos avances en cómputo, en particular, en la computación en paralelo mediante equipos de cómputo de alto desempeño y/o Clusters parecen ser el mecanismo más efectivo para incrementar la capacidad y velocidad de resolución de varios tipos de problemas de interés en Ciencias e Ingenierías usando métodos de descomposición de dominio (véase [57], [67], [68], [70] y [72]).

La implementación de los métodos de descomposición de dominio permite utilizar de forma eficiente, las crecientes capacidades del cómputo en paralelo (véase [18] y [19]) -Grids⁴ de decenas de Clusters, cada uno con cientos o miles de procesadores interconectados por red, con un creciente poder de cómputo medible en Peta Flops-, así como el uso de una amplia memoria -ya sea distribuida y/o compartida del orden de Tera Bytes-, permitiendo atacar

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

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

una gran variedad de problemas que sin estas técnicas es imposible hacerlo de manera flexible y eficiente. Pero hay que notar que existe una amplia gama de problemas que se necesitan resolver, los cuales superan la capacidad de cómputo actual, ya sea por el tiempo requerido para su solución, por el consumo excesivo de memoria o ambos.

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

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

Dos de los esquemas más comúnmente usados (véase [18], [42], [43], [57], [67], [68], [70], [72] y [30]) en los métodos de descomposición de dominio sin traslapes son Finite Element Tearing and Interconnect Dual-Primal (FETI-DP) y Balancing Domain Decomposition by Constraints (BDDC). Aquí, FETI es sinónimo de Finite Element Tearing and Interconnect de Farhat (véase [20] y [21]); y FETI-DP es la versión Dual-Primal de FETI (véase [22] a [24]). BDD es el método de Balancing Domain Decomposition de Mandel (véase [25] y [26]), mientras que BDDC es BDD con restricciones (véase [27] al [29]). Ambos: FETI-DP y BDDC inician de la ecuación diferencial parcial y en ellos los grados de libertad están asociados con las funciones base usadas.

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

Idealmente, los métodos de descomposición buscan satisfacer lo que lla-

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

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

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

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

En el presente trabajo se da una perspectiva general de algunas de las más importantes formulaciones algebraicas de métodos de descomposición de dominio sin traslape. Dos de los esquemas más comúnmente usados son (véase [18], [42], [43], [57], [67], [68], [70] y [72]): BBDC y FETI-DP. Los cuales fueron puestos en el marco primal -véase secciones (14.1) y (14.2)- del espacio de vectores derivados -Derived Vectors Space (DVS)-. El cual permite

una efectiva y sintética presentación de ambos métodos (véase [36] y [38]): Formulaciones primal y dual.

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

El espacio de vectores derivados constituye un espacio de Hilbert con respecto al adecuado producto interior -el producto interior Euclidiano- y mediante la utilización de la formulación DVS, se saca provecho de la estructura del espacio de Hilbert, obteniendo de esta manera una gran simplicidad para el algoritmo en el espacio de funciones definidas por tramos y se usa para establecer una clara correspondencia entre los problemas a nivel continuo y aquellos obtenidos después de la discretización.

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

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

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

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

Idealmente, los métodos DDM intentan producir algoritmos tal que “la

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

Algunas características importantes de la metodología desarrollada son:

1. Las formulaciones Dual y Primal son de dos de los métodos comúnmente usados -BDDC y FETI-DP- que han sido derivados de una manera unificada.
2. El esquema DVS incluye formulaciones algebraicas para matrices simétricas, no simétricas e indefinidas -i.e. no positivas y no negativas definidas-. Además se detallan las condiciones que tales matrices deben de satisfacer para que los algoritmos generales sean aplicables.
3. El esquema DVS permite aplicar técnicas de descomposición de dominio directamente a la matriz que es obtenida después de que la ecuación diferencial -o sistema de tales ecuaciones- ha sido discretizada. La aplicación de tales procedimientos no requiere del conocimiento acerca de la ecuación diferencial que originó la matriz.
4. El esquema DVS puede ser aplicado independientemente del procedimiento de discretización usado para obtenerlo; ya que la matriz que es obtenida inmediatamente después de la discretización no está definida en el espacio de vectores derivados. La teoría provee una fórmula para derivar otra matriz en términos del problema formulado en el espacio de vectores derivados (véase [34]).
5. Como es común en los métodos de descomposición de dominio, la matriz global nunca se construye; a este respecto, una manera simple de definir el espacio vectorial derivado es presentada en este trabajo.

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

1.4 Objetivos del trabajo

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

Objetivos Generales Los objetivos generales del presente trabajo son:

- Presentar el esquema del método de descomposición de dominio en el espacio de vectores derivados (véase [36], [39] y [38]), el cual permite aplicar técnicas de descomposición de dominio directamente al sistema de matrices que son obtenidas después de que la ecuación diferencial o sistema de tales ecuaciones han sido discretizadas. La aplicación de tales procedimientos no requiere del conocimiento acerca de la ecuación diferencial que originó las matrices.
- Mostrar cómo la teoría provee una fórmula para derivar la matriz en términos del problema formulado en el espacio de vectores derivados, ya que la matriz que es obtenida inmediatamente después de la discretización no está definida en el espacio de vectores derivados. Como es común en los métodos de descomposición de dominio, tal matriz nunca se construye. A este respecto, una manera simple de definir el espacio de vectores derivados es presentada en este trabajo.
- Mostrar cómo el esquema DVS es igualmente aplicable a matrices simétricas, no simétricas e indefinidas.
- Mostrar la eficiencia computacional de los algoritmos desarrollados en problemas que generan matrices simétricas y no simétricas, además de mostrar que la implementación en paralelo es escalable.

Objetivos Particulares Los objetivos particulares de este trabajo son:

- Mostrar los ocho algoritmos iterativos básicos de descomposición de dominio sin traslape que se han obtenido, de los cuales cuatro son formulaciones primales y los otros cuatro son formulaciones duales.
- Mostrar como dos de las fórmulas preconditionadas corresponden a los algoritmos FETI-DP y BDDC.

- Mostrar en el caso de matrices simétricas y no simétricas (véase [34] y [35]), en particular para problemas de Advección-Difusión (véase [29] y [52]) que la eficiencia numérica de los algoritmos preconditionados están en el mismo orden que los reportados en la literatura.

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

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

1.5 Infraestructura Computacional Usada

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

Hay que notar que, el paradigma de programación orientada a objetos sacrifica algo de eficiencia computacional por requerir mayor manejo de recursos computacionales al momento de la ejecución. Pero en contraste, permite mayor flexibilidad a la hora de adaptar los códigos a nuevas especificaciones. Adicionalmente, disminuye notoriamente el tiempo invertido en el mantenimiento y búsqueda de errores dentro del código. Esto tiene especial interés

cuando se piensa en la cantidad de meses invertidos en la programación comparada con los segundos consumidos en la ejecución.

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

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

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

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

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

- Cluster homogéneo IO de 22 Cores Intel Xeon a 2.8 GHz de 32 bits, cada 2 Cores con 1 GB de RAM interconectados con un switch de 100 Mbps, a cargo de la Dra. Alejandra Arciniega Ceballos del Departamento de Vulcanología del Instituto de Geofísica de la UNAM.
- PC de alto rendimiento Antipolis de 8 Cores Intel Xeon a 2.33 GHz de 64 bits y 32 GB de RAM, a cargo del Dr. Víctor Cruz Atienza del Departamento de Sismología del Instituto de Geofísica de la UNAM.

1.6 Organización del Trabajo

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

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

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

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

En el capítulo ocho, se realiza el análisis de rendimiento para mostrar la

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

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

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

En el apéndice B, se muestra como hacer una eficiente implementación para la solución de grandes sistemas de ecuaciones lineales por medio de los métodos directos e iterativos, así como las estructuras óptimas de las matrices al codificarse en el lenguaje de programación C++ tanto para su implementación secuencial como en paralelo.

1.7 Agradecimientos

Este texto es una recopilación de múltiples fuentes, nuestra aportación -si es que podemos llamarla así- es plasmarlo en este documento, en el que tratamos de dar coherencia a nuestra visión de los temas desarrollados.

En la realización de este texto se han revisado -en la mayoría de los casos indicamos la referencia, pero pudimos omitir varias de ellas, por lo cual pedimos una disculpa- múltiples páginas Web, artículos técnicos, libros, entre otros materiales bibliográficos, los más representativos y de libre acceso los ponemos a su disposición en la siguiente liga:

Herramientas
<http://132.248.181.216/Herramientas/>

Además, la documentación y los diferentes ejemplos que se presentan en este trabajo, se encuentran disponibles en dicha liga, para que puedan ser copiados desde el navegador y ser usados. En aras de que el interesado pueda correr dichos ejemplos y afianzar sus conocimientos, además de que puedan ser usados en diferentes ámbitos a los presentados aquí.

Agradezco la revisión y aportaciones del Dr. Robert Yates (Alternativas en Computación, S.A de C.V), del Dr. Ismael Herrera Revilla (Instituto de Geofísica, UNAM) y del Dr. Martín Alberto Díaz Viera (Instituto Mexicano del Petróleo, IMP).

2 Sistemas Continuos y sus Modelos

Los fundamentos de la física macroscópica nos proporciona la ‘teoría de los medios continuos’. En este capítulo, en base a ella se introduce una formulación clara, general y sencilla de los modelos matemáticos de los sistemas continuos. Esta formulación es tan sencilla y tan general, que los modelos básicos de sistemas tan complicados y diversos como la atmósfera, los océanos, los yacimientos petroleros, o los geotérmicos, se derivan por medio de la aplicación repetida de una sola ecuación diferencial: ‘la ecuación diferencial de balance’.

Dicha formulación también es muy clara, pues en el modelo general no hay ninguna ambigüedad; en particular, todas las variables y parámetros que intervienen en él, están definidos de manera unívoca. En realidad, este modelo general de los sistemas continuos constituye una realización extraordinaria de los paradigmas del pensamiento matemático. El descubrimiento del hecho de que los modelos matemáticos de los sistemas continuos, independientemente de su naturaleza y propiedades intrínsecas, pueden formularse por medio de balances, cuya idea básica no difiere mucho de los balances de la contabilidad financiera, fue el resultado de un largo proceso de perfeccionamiento en el que concurrieron una multitud de mentes brillantes.

2.1 Los Modelos

Un modelo de un sistema es un sustituto de cuyo comportamiento es posible derivar el correspondiente al sistema original. Los modelos matemáticos, en la actualidad, son los utilizados con mayor frecuencia y también los más versátiles. En las aplicaciones específicas están constituidos por programas de cómputo cuya aplicación y adaptación a cambios de las propiedades de los sistemas es relativamente fácil. También, sus bases y las metodologías que utilizan son de gran generalidad, por lo que es posible construirlos para situaciones y sistemas muy diversos.

Los modelos matemáticos son entes en los que se integran los conocimientos científicos y tecnológicos, con los que se construyen programas de cómputo que se implementan con medios computacionales. En la actualidad, la simulación numérica permite estudiar sistemas complejos y fenómenos naturales que sería muy costoso, peligroso o incluso imposible de estudiar por experimentación directa. En esta perspectiva la significación de los modelos matemáticos en ciencias e ingeniería es clara, porque la modelación

matemática constituye el método más efectivo de predecir el comportamiento de los diversos sistemas de interés. En nuestro país, ellos son usados ampliamente en la industria petrolera, en las ciencias y la ingeniería del agua y en muchas otras.

2.1.1 Física Microscópica y Física Macroscópica

La materia, cuando se le observa en el ámbito ultramicroscópico, está formada por moléculas y átomos. Estos a su vez, por partículas aún más pequeñas como los protones, neutrones y electrones. La predicción del comportamiento de estas partículas es el objeto de estudio de la mecánica cuántica y la física nuclear. Sin embargo, cuando deseamos predecir el comportamiento de sistemas tan grandes como la atmósfera o un yacimiento petrolero, los cuales están formados por un número extraordinariamente grande de moléculas y átomos, su estudio resulta inaccesible con esos métodos y en cambio el enfoque macroscópico es apropiado.

Por eso en lo que sigue distinguiremos dos enfoques para el estudio de la materia y su movimiento. El primero -el de las moléculas, los átomos y las partículas elementales- es el enfoque microscópico y el segundo es el enfoque macroscópico. Al estudio de la materia con el enfoque macroscópico, se le llama física macroscópica y sus bases teóricas las proporciona la mecánica de los medios continuos.

Cuando se estudia la materia con este último enfoque, se considera que los cuerpos llenan el espacio que ocupan, es decir que no tienen huecos, que es la forma en que los vemos sin el auxilio de un microscopio. Por ejemplo, el agua llena todo el espacio del recipiente donde está contenida. Este enfoque macroscópico está presente en la física clásica. La ciencia ha avanzado y ahora sabemos que la materia está llena de huecos, que nuestros sentidos no perciben y que la energía también está cuantizada. A pesar de que estos dos enfoques para el análisis de los sistemas físicos, el microscópico y el macroscópico, parecen a primera vista conceptualmente contradictorios, ambos son compatibles, y complementarios, y es posible establecer la relación entre ellos utilizando la mecánica estadística.

2.2 Cinemática de los Modelos de Sistemas Continuos

En la teoría de los sistemas continuos, los cuerpos llenan todo el espacio que ocupan. Y en cada punto del espacio físico hay una y solamente una partícula.

Así, definimos como sistema continuo a un conjunto de partículas. Aún más, dicho conjunto es un subconjunto del espacio Euclidiano tridimensional. Un cuerpo es un subconjunto de partículas que en cualquier instante dado ocupa un dominio, en el sentido matemático, del espacio físico; es decir, del espacio Euclidiano tridimensional. Denotaremos por $B(t)$ a la región ocupada por el cuerpo \mathcal{B} , en el tiempo t , donde t puede ser cualquier número real.

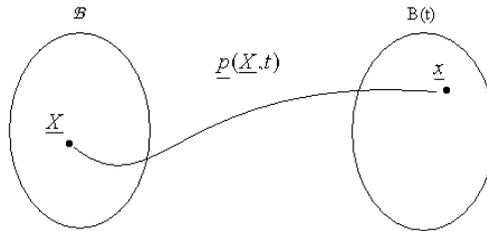


Figura 1: Representación del movimiento de partículas de un cuerpo \mathcal{B} , para un tiempo dado.

Frecuentemente, sin embargo, nuestro interés de estudio se limitará a un intervalo finito de tiempo. Dado un cuerpo \mathcal{B} , todo subdominio $\tilde{\mathcal{B}} \subset \mathcal{B}$, constituye a su vez otro cuerpo; en tal caso, se dice que $\tilde{\mathcal{B}} \subset \mathcal{B}$ es un subcuerpo de \mathcal{B} . De acuerdo con lo mencionado antes, una hipótesis básica de la teoría de los sistemas continuos es que en cualquier tiempo $t \in (-\infty, \infty)$ y en cada punto $x \in \mathcal{B}$ de la región ocupada por el cuerpo, hay una y sólo una partícula del cuerpo. Como en nuestra revisión se incluye no solamente la estática (es decir, los cuerpos en reposo), sino también la dinámica (es decir, los cuerpos en movimiento), un primer problema de la cinemática de los sistemas continuos consiste en establecer un procedimiento para identificar a las partículas cuando están en movimiento en el espacio físico.

Sea $\underline{X} \in \mathcal{B}$, una partícula y $p(\underline{X}, t)$ el vector de la posición que ocupa, en el espacio físico, dicha partícula en el instante t . Una forma, pero no la única, de identificar la partícula \underline{X} es asociándole la posición que ocupa en un instante determinado. Tomaremos en particular el tiempo $t = 0$, en tal caso $p(\underline{X}, 0) \equiv \underline{X}$.

A las coordenadas del vector $\underline{X} \equiv (x_1, x_2, x_3)$, se les llama las coordenadas materiales de la partícula. En este caso, las coordenadas materiales de una partícula son las coordenadas del punto del espacio físico que ocupaba la partícula en el tiempo inicial, $t = 0$. Desde luego, el tiempo inicial puede ser

cualquier otro, si así se desea. Sea \mathcal{B} el dominio ocupado por un cuerpo en el tiempo inicial, entonces $\underline{X} \in \mathcal{B}$ si y solamente si la partícula \underline{X} es del cuerpo. Es decir, \mathcal{B} caracteriza al cuerpo. Sin embargo, debido al movimiento, la región ocupada por el mismo cambia con el tiempo y será denotada por $\mathcal{B}(t)$.

Formalmente, para cualquier $t \in (-\infty, \infty)$, $\mathcal{B}(t)$ se define por

$$\mathcal{B}(t) \equiv \{ \underline{x} \in \mathbb{R}^3 \mid \exists \underline{X} \in \mathcal{B} \text{ tal que } \underline{x} = p(\underline{X}, t) \} \quad (2.1)$$

el vector posición $p(\underline{X}, t)$ es función del vector tridimensional \underline{X} y del tiempo. Si fijamos el tiempo t , $p(\underline{X}, t)$ define una transformación del espacio Euclideo \mathbb{R}^3 en sí mismo y la Ec. (2.1) es equivalente a $\mathcal{B}(t) = \underline{p}(\mathcal{B}, t)$. Una notación utilizada para representar esta familia de funciones es $\underline{p}(\cdot, t)$. De acuerdo a la hipótesis de los sistemas continuos: En cualquier tiempo $t \in (-\infty, \infty)$ y en cada punto $\underline{x} \in \mathcal{B}$ de la región ocupada por el cuerpo hay una y sólo una partícula del cuerpo \mathcal{B} para cada t fijo. Es decir, $\underline{p}(\cdot, t)$ es una función biunívoca, por lo que existe la función inversa $\underline{p}^{-1}(\cdot, t)$.

Si se fija la partícula \underline{X} en la función $\underline{p}(\underline{X}, t)$ y se varía el tiempo t , se obtiene su trayectoria. Esto permite obtener la velocidad de cualquier partícula, la cual es un concepto central en la descripción del movimiento. Ella se define como la derivada con respecto al tiempo de la posición cuando la partícula se mantiene fija. Es decir, es la derivada parcial con respecto al tiempo de la función de posición $\underline{p}(\underline{X}, t)$. Por lo mismo, la velocidad como función de las coordenadas materiales de las partículas, está dada por

$$\underline{V}(\underline{X}, t) \equiv \frac{\partial p}{\partial t}(\underline{X}, t). \quad (2.2)$$

2.2.1 Propiedades Intensivas y sus Representaciones

En lo que sigue consideraremos funciones definidas para cada tiempo, en cada una de las partículas de un sistema continuo. A tales funciones se les llama ‘propiedades intensivas’. Las propiedades intensivas pueden ser funciones escalares o funciones vectoriales. Por ejemplo, la velocidad, definida por la Ec. (2.2), es una función vectorial que depende de la partícula \underline{X} y del tiempo t .

Una propiedad intensiva con valores vectoriales es equivalente a tres escalares, correspondientes a cada una de sus tres componentes. Hay dos formas de representar a las propiedades intensivas: la representación Euleriana y la representación Lagrangiana. Los nombres son en honor a los matemáticos

Leonard Euler (1707-1783) y Joseph Louis Lagrange (1736-1813), respectivamente. Frecuentemente, el punto de vista Lagrangiano es utilizado en el estudio de los sólidos, mientras que el Euleriano se usa más en el estudio de los fluidos.

Considere una propiedad intensiva escalar, la cual en el tiempo t toma el valor $\phi(\underline{X}, t)$ en la partícula \underline{X} . Entonces, de esta manera se define una función $\phi : \mathcal{B} \rightarrow \mathbb{R}^1$, para cada $t \in (-\infty, \infty)$ a la que se denomina representación Lagrangiana de la propiedad intensiva considerada. Ahora, sea $\psi(\underline{x}, t)$ el valor que toma esa propiedad en la partícula que ocupa la posición \underline{x} , en el tiempo t . En este caso, para cada $t \in (-\infty, \infty)$ se define una función $\psi : \mathcal{B}(t) \rightarrow \mathbb{R}^1$ a la cual se denomina representación Euleriana de la función considerada. Estas dos representaciones de una misma propiedad están relacionadas por la siguiente identidad

$$\phi(\underline{X}, t) \equiv \psi(\underline{p}(\underline{X}, t), t). \quad (2.3)$$

Nótese que, aunque ambas representaciones satisfacen la Ec. (2.3), las funciones $\phi(\underline{X}, t)$ y $\psi(\underline{x}, t)$ no son idénticas. Sus argumentos \underline{X} y \underline{x} son vectores tridimensionales (es decir, puntos de \mathbb{R}^3); sin embargo, si tomamos $\underline{X} = \underline{x}$, en general

$$\phi(\underline{X}, t) \neq \psi(\underline{X}, t). \quad (2.4)$$

La expresión de la velocidad de una partícula dada por la Ec. (2.2), define a su representación Lagrangiana, por lo que utilizando la Ec. (2.3) es claro que

$$\frac{\partial p}{\partial t}(\underline{X}, t) = \underline{V}(\underline{X}, t) \equiv \underline{v}(\underline{p}(\underline{X}, t), t) \quad (2.5)$$

donde $\underline{v}(\underline{x}, t)$ es la representación Euleriana de la velocidad. Por lo mismo

$$\underline{v}(\underline{x}, t) \equiv \underline{V}(\underline{p}^{-1}(\underline{x}, t), t). \quad (2.6)$$

Esta ecuación tiene la interpretación de que la velocidad en el punto \underline{x} del espacio físico, es igual a la velocidad de la partícula que pasa por dicho punto en el instante t . La Ec. (2.6) es un caso particular de la relación

$$\psi(\underline{x}, t) \equiv \phi(\underline{p}^{-1}(\underline{x}, t), t)$$

de validez general, la cual es otra forma de expresar la relación de la Ec. (2.3) que existe entre las dos representaciones de una misma propiedad intensiva.

La derivada parcial con respecto al tiempo de la representación Lagrangiana $\phi(\underline{X}, t)$ de una propiedad intensiva, de acuerdo a la definición de la derivada parcial de una función, es la tasa de cambio con respecto al tiempo que ocurre en una partícula fija. Es decir, si nos montamos en una partícula y medimos a la propiedad intensiva y luego los valores así obtenidos los derivamos con respecto al tiempo, el resultado final es $\frac{\partial\phi(\underline{X}, t)}{\partial t}$. En cambio, si $\psi(\underline{x}, t)$ es la representación Euleriana de esa misma propiedad, entonces $\frac{\partial\psi(\underline{x}, t)}{\partial t}$ es simplemente la tasa de cambio con respecto al tiempo que ocurre en un punto fijo en el espacio. Tiene interés evaluar la tasa de cambio con respecto al tiempo que ocurre en una partícula fija, cuando se usa la representación Euleriana. Derivando con respecto al tiempo a la identidad de la Ec. (2.3) y la regla de la cadena, se obtiene

$$\frac{\partial\phi(\underline{X}, t)}{\partial t} = \frac{\partial\psi}{\partial t}(\underline{p}(\underline{X}, t), t) + \sum_{i=1}^3 \frac{\partial\psi}{\partial x_i}(\underline{p}(\underline{X}, t), t) \frac{\partial p_i}{\partial t}(\underline{X}, t). \quad (2.7)$$

Se acostumbra definir el símbolo $\frac{D\psi}{Dt}$ por

$$\frac{D\psi}{Dt} = \frac{\partial\psi}{\partial t} + \sum_{i=1}^3 v_i \frac{\partial\psi}{\partial x_i} \quad (2.8)$$

o, más brevemente,

$$\frac{D\psi}{Dt} = \frac{\partial\psi}{\partial t} + \underline{v} \cdot \nabla\psi \quad (2.9)$$

utilizando esta notación, se puede escribir

$$\frac{\partial\phi(\underline{X}, t)}{\partial t} = \frac{D\psi}{Dt}(\underline{p}(\underline{X}, t)) \equiv \left(\frac{\partial\psi}{\partial t} + \underline{v} \cdot \nabla\psi \right) (\underline{p}(\underline{X}, t), t). \quad (2.10)$$

Por ejemplo, la aceleración de una partícula se define como la derivada de la velocidad cuando se mantiene a la partícula fija. Aplicando la Ec. (2.9) se tiene

$$\frac{D\underline{v}}{Dt} = \frac{\partial\underline{v}}{\partial t} + \underline{v} \cdot \nabla\underline{v} \quad (2.11)$$

una expresión más transparente se obtiene aplicando la Ec. (2.9) a cada una de las componentes de la velocidad. Así, se obtiene

$$\frac{Dv_i}{Dt} = \frac{\partial v_i}{\partial t} + \underline{v} \cdot \nabla v_i. \quad (2.12)$$

Desde luego, la aceleración, en representación Lagrangiana es simplemente

$$\frac{\partial}{\partial t} \underline{V}(\underline{X}, t) = \frac{\partial^2}{\partial t^2} p(\underline{X}, t). \quad (2.13)$$

2.2.2 Propiedades Extensivas

En la sección anterior se consideraron funciones definidas en las partículas de un cuerpo, más precisamente, funciones que hacen corresponder a cada partícula y cada tiempo un número real, o un vector del espacio Euclidiano tridimensional \mathbb{R}^3 . En esta, en cambio, empezaremos por considerar funciones que a cada cuerpo \mathcal{B} de un sistema continuo, y a cada tiempo t le asocia un número real o un vector de \mathbb{R}^3 . A una función de este tipo $\mathbb{E}(\mathcal{B}, t)$ se le llama ‘propiedad extensiva’ cuando está dada por una integral

$$\mathbb{E}(\mathcal{B}, t) \equiv \int_{\mathcal{B}(t)} \psi(\underline{x}, t) d\underline{x}. \quad (2.14)$$

Observe que, en tal caso, el integrando define una función $\psi(\underline{x}, t)$ y por lo mismo, una propiedad intensiva. En particular, la función $\psi(\underline{x}, t)$ es la representación Euleriana de esa propiedad intensiva. Además, la Ec. (2.14) establece una correspondencia biunívoca entre las propiedades extensivas y las intensivas, porque dada la representación Euleriana $\psi(\underline{x}, t)$ de cualquier propiedad intensiva, su integral sobre el dominio ocupado por cualquier cuerpo, define una propiedad extensiva. Finalmente, la notación empleada en la Ec. (2.14) es muy explícita, pues ahí se ha escrito $\mathbb{E}(\mathcal{B}, t)$ para enfatizar que el valor de la propiedad extensiva corresponde al cuerpo \mathcal{B} . Sin embargo, en lo que sucesivo, se simplificará la notación omitiendo el símbolo \mathcal{B} es decir, se escribirá $\mathbb{E}(t)$ en vez de $\mathbb{E}(\mathcal{B}, t)$.

Hay diferentes formas de definir a las propiedades intensivas. Como aquí lo hemos hecho, es por unidad de volumen. Sin embargo, es frecuente que se le defina por unidad de masa véase [10]. Es fácil ver que la propiedad intensiva por unidad de volumen es igual a la propiedad intensiva por unidad de masa multiplicada por la densidad de masa (es decir, masa por unidad de volumen), por lo que es fácil pasar de un concepto al otro, utilizando la densidad de masa.

Sin embargo, una ventaja de utilizar a las propiedades intensivas por unidad de volumen, en lugar de las propiedades intensivas por unidad de masa, es que la correspondencia entre las propiedades extensivas y las intensivas es más directa: dada una propiedad extensiva, la propiedad intensiva

que le corresponde es la función que aparece como integrando, cuando aquélla se expresa como una integral de volumen. Además, del cálculo se sabe que

$$\psi(\underline{x}, t) \equiv \lim_{Vol \rightarrow 0} \frac{\mathbb{E}(t)}{Vol} = \lim_{Vol \rightarrow 0} \frac{\int_{\mathcal{B}(t)} \psi(\underline{\xi}, t) d\underline{\xi}}{Vol}. \quad (2.15)$$

La Ec. (2.15) proporciona un procedimiento efectivo para determinar las propiedades extensivas experimentalmente: se mide la propiedad extensiva en un volumen pequeño del sistema continuo de que se trate, se le divide entre el volumen y el cociente que se obtiene es una buena aproximación de la propiedad intensiva.

El uso que haremos del concepto de propiedad extensiva es, desde luego, lógicamente consistente. En particular, cualquier propiedad que satisface las condiciones de la definición de propiedad extensiva establecidas antes es, por ese hecho, una propiedad extensiva. Sin embargo, no todas las propiedades extensivas que se pueden obtener de esta manera son de interés en la mecánica de los medios continuos. Una razón básica por la que ellas son importantes es porqué el modelo general de los sistemas continuos se fórmula en términos de ecuaciones de balance de propiedades extensivas, como se verá más adelante.

2.2.3 Balance de Propiedades Extensivas e Intensivas

Los modelos matemáticos de los sistemas continuos están constituidos por balances de propiedades extensivas. Por ejemplo, los modelos de transporte de solutos (los contaminantes transportados por corrientes superficiales o subterráneas, son un caso particular de estos procesos de transporte) se construyen haciendo el balance de la masa de soluto que hay en cualquier dominio del espacio físico. Aquí, el término balance se usa, esencialmente, en un sentido contable. En la contabilidad que se realiza para fines financieros o fiscales, la diferencia de las entradas menos las salidas nos da el aumento, o cambio, de capital. En forma similar, en la mecánica de los medios continuos se realiza, en cada cuerpo del sistema continuo, un balance de las propiedades extensivas en que se basa el modelo.

Ecuación de Balance Global Para realizar tales balances es necesario, en primer lugar, identificar las causas por las que las propiedades extensivas pueden cambiar. Tomemos como ejemplo de propiedad extensiva a las existencias de maíz que hay en el país. La primera pregunta es: ¿qué causas

pueden motivar su variación, o cambio, de esas existencias?. Un análisis sencillo nos muestra que dicha variación puede ser debida a que se produzca o se consuma. También a que se importe o se exporte por los límites del país (fronteras o litorales). Y con esto se agotan las causas posibles; es decir, esta lista es exhaustiva. Producción y consumo son términos similares, pero sus efectos tienen signos opuestos, que fácilmente se engloban en uno solo de esos conceptos. De hecho, si convenimos en que la producción puede ser negativa, entonces el consumo es una producción negativa.

Una vez adoptada esta convención, ya no es necesario ocuparnos separadamente del consumo. En forma similar, la exportación es una importación negativa. Entonces, el incremento en las existencias $\Delta\mathbb{E}$ en un período Δt queda dado por la ecuación

$$\Delta\mathbb{E} = \mathbb{P} + \mathbb{I} \quad (2.16)$$

donde a la producción y a la importación, ambas con signo, se les ha representado por \mathbb{P} y \mathbb{I} respectivamente.

Similarmente, en la mecánica de los medios continuos, la lista exhaustiva de las causas por las que una propiedad extensiva de cualquier cuerpo puede cambiar, contiene solamente dos motivos:

- i) Por producción en el interior del cuerpo; y
- ii) Por importación (es decir, transporte) a través de la frontera.

Esto conduce a la siguiente ecuación de “balance global”, de gran generalidad, para las propiedades extensivas

$$\frac{d\mathbb{E}}{dt}(t) = \int_{\mathcal{B}(t)} g(\underline{x}, t) d\underline{x} + \int_{\partial\mathcal{B}(t)} q(\underline{x}, t) d\underline{x} + \int_{\Sigma(t)} g_{\Sigma}(\underline{x}, t) d\underline{x}. \quad (2.17)$$

Donde $g(\underline{x}, t)$ es la generación en el interior del cuerpo, con signo, de la propiedad extensiva correspondiente, por unidad de volumen, por unidad de tiempo. Además, en la Ec. (2.17) se ha tomado en cuenta la posibilidad de que haya producción concentrada en la superficie $\Sigma(t)$, la cual está dada en esa ecuación por la última integral, donde $g_{\Sigma}(\underline{x}, t)$ es la producción por unidad de área. Por otra parte $q(\underline{x}, t)$ es lo que se importa o transporta hacia el interior del cuerpo a través de la frontera del cuerpo $\partial\mathcal{B}(t)$, en otras palabras, es el flujo de la propiedad extensiva a través de la frontera del cuerpo, por unidad de área, por unidad de tiempo. Puede demostrarse, con

base en hipótesis válidas en condiciones muy generales, que para cada tiempo t existe un campo vectorial $\tau(\underline{x}, t)$ tal que

$$q(\underline{x}, t) \equiv \tau(\underline{x}, t) \cdot \underline{n}(\underline{x}, t) \quad (2.18)$$

donde $\underline{n}(\underline{x}, t)$ es normal exterior a $\partial\mathcal{B}(t)$. En vista de esta relación, la Ec. (2.17) de balance se puede escribir como

$$\frac{d\mathbb{E}}{dt}(t) = \int_{\mathcal{B}(t)} g(\underline{x}, t) d\underline{x} + \int_{\partial\mathcal{B}(t)} \tau(\underline{x}, t) \cdot \underline{n}(\underline{x}, t) d\underline{x} + \int_{\Sigma(t)} g_{\Sigma}(\underline{x}, t) d\underline{x}. \quad (2.19)$$

La relación (2.19) se le conoce con el nombre de “ecuación general de balance global” y es la ecuación básica de los balances de los sistemas continuos. A la función $g(\underline{x}, t)$ se le denomina el generación interna y al campo vectorial $\tau(\underline{x}, t)$ el campo de flujo.

Condiciones de Balance Local Los modelos de los sistemas continuos están constituidos por las ecuaciones de balance correspondientes a una colección de propiedades extensivas. Así, a cada sistema continuo le corresponde una familia de propiedades extensivas, tal que, el modelo matemático del sistema está constituido por las condiciones de balance de cada una de las propiedades extensivas de dicha familia.

Sin embargo, las propiedades extensivas mismas no se utilizan directamente en la formulación del modelo, en su lugar se usan las propiedades intensivas asociadas a cada una de ellas. Esto es posible porque las ecuaciones de balance global son equivalentes a las llamadas condiciones de balance local, las cuales se expresan en términos de las propiedades intensivas correspondientes. Las condiciones de balance local son de dos clases: ‘las ecuaciones diferenciales de balance local’ y ‘las condiciones de salto’.

Las primeras son ecuaciones diferenciales parciales, que se deben satisfacer en cada punto del espacio ocupado por el sistema continuo, y las segundas son ecuaciones algebraicas que las discontinuidades deben satisfacer donde ocurren; es decir, en cada punto de Σ . Cabe mencionar que las ecuaciones diferenciales de balance local son de uso mucho más amplio que las condiciones de salto, pues estas últimas solamente se aplican cuando y donde hay discontinuidades, mientras que las primeras en todo punto del espacio ocupado por el sistema continuo.

Una vez establecidas las ecuaciones diferenciales y de salto del balance local, e incorporada la información científica y tecnológica necesaria para

completar el modelo (la cual por cierto se introduce a través de las llamadas 'ecuaciones constitutivas'), el problema matemático de desarrollar el modelo y derivar sus predicciones se transforma en uno correspondiente a la teoría de la ecuaciones diferenciales, generalmente parciales, y sus métodos numéricos.

Las Ecuaciones de Balance Local En lo que sigue se supone que las propiedades intensivas pueden tener discontinuidades, de salto exclusivamente, a través de la superficie $\Sigma(t)$. Se entiende por 'discontinuidad de salto', una en que el límite por ambos lados de $\Sigma(t)$ existe, pero son diferentes.

Se utilizará en lo que sigue los resultados matemáticos que se dan a continuación, ver [37].

Teorema 1 Para cada $t > 0$, sea $\mathcal{B}(t) \subset \mathbb{R}^3$ el dominio ocupado por un cuerpo. Suponga que la 'propiedad intensiva' $\psi(\underline{x}, t)$ es de clase C^1 , excepto a través de la superficie $\Sigma(t)$. Además, sean las funciones $\underline{v}(\underline{x}, t)$ y $\underline{v}_\Sigma(\underline{x}, t)$ esta última definida para $\underline{x} \in \Sigma(t)$ solamente, las velocidades de las partículas y la de $\Sigma(t)$, respectivamente. Entonces

$$\frac{d}{dt} \int_{\mathcal{B}(t)} \psi d\underline{x} \equiv \int_{\mathcal{B}(t)} \left\{ \frac{\partial \psi}{\partial t} + \nabla \cdot (\underline{v}\psi) \right\} d\underline{x} + \int_{\Sigma} [(\underline{v} - \underline{v}_\Sigma) \psi] \cdot \underline{n} d\underline{x}. \quad (2.20)$$

Teorema 2 Considere un sistema continuo, entonces, la 'ecuación de balance global' (2.19) se satisface para todo cuerpo del sistema continuo si y solamente si se cumplen las condiciones siguientes:

i) La ecuación diferencial

$$\frac{\partial \psi}{\partial t} + \nabla \cdot (\underline{v}\psi) = \nabla \cdot \underline{\tau} + g \quad (2.21)$$

vale en todo punto $\underline{x} \in \mathbb{R}^3$, de la región ocupada por el sistema.

ii) La ecuación

$$[\psi (\underline{v} - \underline{v}_\Sigma) - \underline{\tau}] \cdot \underline{n} = g_\Sigma \quad (2.22)$$

vale en todo punto $\underline{x} \in \Sigma$.

A las ecuaciones (2.21) y (2.22), se les llama 'ecuación diferencial de balance local' y 'condición de salto', respectivamente.

Desde luego, el caso más general que se estudiará se refiere a situaciones dinámicas; es decir, aquéllas en que las propiedades intensivas cambian con el tiempo. Sin embargo, los estados estacionarios de los sistemas continuos

son de sumo interés. Por estado estacionario se entiende uno en que las propiedades intensivas son independientes del tiempo. En los estados estacionarios, además, las superficies de discontinuidad $\Sigma(t)$ se mantienen fijas (no se mueven). En este caso $\frac{\partial \psi}{\partial t} = 0$ y $\underline{v}_\Sigma = 0$. Por lo mismo, para los estados estacionarios, la ecuación de balance local y la condición de salto se reducen a

$$\nabla \cdot (\underline{v}\psi) = \nabla \cdot \underline{\tau} + g \quad (2.23)$$

que vale en todo punto $\underline{x} \in \mathbb{R}^3$ y

$$[\psi \underline{v} - \underline{\tau}] \cdot \underline{n} = g_\Sigma \quad (2.24)$$

que se satisface en todo punto de la discontinuidad $\Sigma(t)$ respectivamente.

2.3 Ejemplos de Modelos

Una de las aplicaciones más sencillas de las condiciones de balance local es para formular restricciones en el movimiento. Aquí ilustramos este tipo de aplicaciones formulando condiciones que se deben cumplir localmente cuando un fluido es incompresible. La afirmación de que un fluido es incompresible significa que todo cuerpo conserva el volumen de fluido en su movimiento. Entonces, se considerarán dos casos: el de un ‘fluido libre’ y el de un ‘fluido en un medio poroso’. En el primer caso, el fluido llena completamente el espacio físico que ocupa el cuerpo, por lo que el volumen del fluido es igual al volumen del dominio que ocupa el cuerpo, así

$$V_f(t) = \int_{\mathcal{B}(t)} d\underline{x} \quad (2.25)$$

aquí, $V_f(t)$ es el volumen del fluido y $\mathcal{B}(t)$ es el dominio del espacio físico (es decir, de \mathbb{R}^3) ocupado por el cuerpo. Observe que una forma más explícita de esta ecuación es

$$V_f(t) = \int_{\mathcal{B}(t)} 1 d\underline{x} \quad (2.26)$$

porque en la integral que aparece en la Ec. (2.25) el integrando es la función idénticamente 1. Comparando esta ecuación con la Ec. (2.14), vemos que el volumen del fluido es una propiedad extensiva y que la propiedad intensiva que le corresponde es $\psi = 1$.

Además, la hipótesis de incompresibilidad implica

$$\frac{dV_f}{dt}(t) = 0 \quad (2.27)$$

está es el balance global de la Ec. (2.19), con $g = g_\Sigma = 0$ y $\tau = 0$, el cual a su vez es equivalente a las Ecs. (2.21) y (2.22). Tomando en cuenta además que $\psi = 1$, la Ec. (2.21) se reduce a

$$\nabla \cdot \underline{v} = 0. \quad (2.28)$$

Esta es la bien conocida condición de incompresibilidad para un fluido libre, además, aplicando la Ec. (2.22) donde haya discontinuidades, se obtiene $[\underline{v}] \cdot \underline{n} = 0$. Esto implica que si un fluido libre es incompresible, la velocidad de sus partículas es necesariamente continua.

El caso en que el fluido se encuentra en un ‘medio poroso’, es bastante diferente. Un medio poroso es un material sólido que tiene huecos distribuidos en toda su extensión, cuando los poros están llenos de un fluido, se dice que el medio poroso esta ‘saturado’. Esta situación es la de mayor interés en la práctica y es también la más estudiada. En muchos de los casos que ocurren en las aplicaciones el fluido es agua o petróleo. A la fracción del volumen del sistema, constituido por la ‘matriz sólida’ y los huecos, se le llama ‘porosidad’ y se le representara por ϕ , así

$$\phi(x, t) = \lim_{V \rightarrow 0} \frac{\text{Volumen de huecos}}{\text{Volumen total}} \quad (2.29)$$

aquí hemos escrito $\phi(x, t)$ para enfatizar que la porosidad generalmente es función tanto de la posición como del tiempo. Las variaciones con la posición pueden ser debidas, por ejemplo, a la heterogeneidad del medio y los cambios con el tiempo a su elasticidad; es decir, los cambios de presión del fluido originan esfuerzos en los poros que los dilatan o los encogen.

Cuando el medio está saturado, el volumen del fluido V_f es igual al volumen de los huecos del dominio del espacio físico que ocupa, así

$$V_f(t) = \int_{\mathcal{B}(t)} \phi(x, t) d\underline{x}. \quad (2.30)$$

En vista de esta ecuación, la propiedad intensiva asociada al volumen de fluido es la porosidad $\phi(x, t)$ por lo que la condición de incompresibilidad del fluido contenido en un medio poroso, está dada por la ecuación diferencial

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\underline{v}\phi) = 0. \quad (2.31)$$

Que la divergencia de la velocidad sea igual a cero en la Ec. (2.28) como condición para que un fluido en su movimiento libre conserve su volumen, es ampliamente conocida. Sin embargo, este no es el caso de la Ec. (2.31), como condición para la conservación del volumen de los cuerpos de fluido contenidos en un medio poroso. Finalmente, debe observarse que cualquier fluido incompresible satisface la Ec. (2.28) cuando se mueve en el espacio libre y la Ec. (2.31) cuando se mueve en un medio poroso.

Cuando un fluido efectúa un movimiento en el que conserva su volumen, al movimiento se le llama ‘isocórico’. Es oportuno mencionar que si bien es cierto que cuando un fluido tiene la propiedad de ser incompresible, todos sus movimientos son isocóricos, lo inverso no es cierto: un fluido compresible en ocasiones puede efectuar movimientos isocóricos.

Por otra parte, cuando un fluido conserva su volumen en su movimiento satisface las condiciones de salto de Ec. (2.22), las cuales para este caso son

$$[\phi(\underline{v} - \underline{v}_\Sigma)] \cdot \underline{n} = 0. \quad (2.32)$$

En aplicaciones a geohidrología y a ingeniería petrolera, las discontinuidades de la porosidad están asociadas a cambios en los estratos geológicos y por está razón están fijas en el espacio; así, $\underline{v}_\Sigma = 0$ y la Ec. (2.32) se reduce a

$$[\phi \underline{v}] \cdot \underline{n} = 0 \quad (2.33)$$

o, de otra manera

$$\phi_+ v_{n_+} = \phi_- v_{n_-}. \quad (2.34)$$

Aquí, la componente normal de la velocidad es $v_n \equiv \underline{v} \cdot \underline{n}$ y los subíndices más y menos se utilizan para denotar los límites por los lados más y menos de Σ , respectivamente. Al producto de la porosidad por la velocidad se le conoce con el nombre de velocidad de Darcy \underline{U} , es decir

$$\underline{U} = \phi \underline{v} \quad (2.35)$$

utilizando, las Ecs. (2.33) y (2.34) obtenemos

$$[\underline{U}] \cdot \underline{n} = 0 \quad \text{y} \quad \underline{U}_{n_+} = \underline{U}_{n_-} \quad (2.36)$$

es decir, 1.

La Ec. (2.34) es ampliamente utilizada en el estudio del agua subterránea (geohidrología). Ahí, es frecuente que la porosidad ϕ sea discontinua en la

superficie de contacto entre dos estratos geológicos diferentes, pues generalmente los valores que toma esta propiedad dependen de cada estrato. En tal caso, $\phi_+ \neq \phi_-$ por lo que $v_{n_+} \neq v_{n_-}$ necesariamente.

Para más detalles de la forma y del desarrollo de algunos modelos usados en ciencias de la tierra, véase [37], [10], [47] y [48].

3 Ecuaciones Diferenciales Parciales

Cada una de las ecuaciones de balance da lugar a una ecuación diferencial parcial u ordinaria (en el caso en que el modelo depende de una sola variable independiente), la cual se complementa con las condiciones de salto, en el caso de los modelos discontinuos. Por lo mismo, los modelos de los sistemas continuos están constituidos por sistemas de ecuaciones diferenciales cuyo número es igual al número de propiedades intensivas que intervienen en la formulación del modelo básico.

Los sistemas de ecuaciones diferenciales se clasifican en elípticas, hiperbólicas y parabólicas. Es necesario aclarar que esta clasificación no es exhaustiva; es decir, existen sistemas de ecuaciones diferenciales que no pertenecen a ninguna de estas categorías. Sin embargo, casi todos los modelos de sistemas continuos, en particular los que han recibido mayor atención hasta ahora, si están incluidos en alguna de estas categorías.

3.1 Clasificación

Es importante clasificar a las ecuaciones diferenciales parciales y a los sistemas de tales ecuaciones, por qué muchas de sus propiedades son comunes a cada una de sus clases. Así, su clasificación es un instrumento para alcanzar el objetivo de unidad conceptual. La forma más general de abordar la clasificación de tales ecuaciones, es estudiando la clasificación de sistemas de ecuaciones. Sin embargo, aquí solamente abordaremos el caso de una ecuación diferencial de segundo orden, pero utilizando un método de análisis que es adecuado para extenderse a sistemas de ecuaciones.

La forma general de un operador diferencial cuasi-lineal de segundo orden definido en $\Omega \subset \mathbb{R}^2$ es

$$\mathcal{L}u \equiv a(x, y) \frac{\partial^2 u}{\partial x^2} + b(x, y) \frac{\partial^2 u}{\partial x \partial y} + c(x, y) \frac{\partial^2 u}{\partial y^2} = F(x, y, u, u_x, u_y) \quad (3.1)$$

para una función u de variables independientes x e y . Nos restringimos al caso en que a, b y c son funciones sólo de x e y y no funciones de u .

Para la clasificación de las ecuaciones de segundo orden consideraremos una simplificación de la ecuación anterior en donde $F(x, y, u, u_x, u_y) = 0$ y los coeficientes a, b y c son funciones constantes, es decir

$$a \frac{\partial^2 u}{\partial x^2} + b \frac{\partial^2 u}{\partial x \partial y} + c \frac{\partial^2 u}{\partial y^2} = 0 \quad (3.2)$$

en la cual, examinaremos los diferentes tipos de solución que se pueden obtener para diferentes elecciones de a, b y c . Entonces iniciando con una solución de la forma

$$u(x, y) = f(mx + y) \quad (3.3)$$

para una función f de clase \mathbb{C}^2 y para una constante m , que deben ser determinadas según los requerimientos de la Ec. (3.2). Usando un apóstrofe para denotar la derivada de f con respecto de su argumento, las derivadas parciales requeridas de segundo orden de la Ec. (3.2) son

$$\frac{\partial^2 u}{\partial x^2} = m^2 f'', \quad \frac{\partial^2 u}{\partial x \partial y} = m f'', \quad \frac{\partial^2 u}{\partial y^2} = f'' \quad (3.4)$$

sustituyendo la ecuación anterior en la Ec. (3.2) obtenemos

$$(am^2 + bm + c) f'' = 0 \quad (3.5)$$

de la cual podemos concluir que $f'' = 0$ ó $am^2 + bm + c = 0$ ó ambas. En el caso de que $f'' = 0$ obtenemos la solución $f = f_0 + mx + y$, la cual es una función lineal de x e y y es expresada en términos de dos constantes arbitrarias, f_0 y m . En el otro caso obtenemos

$$am^2 + bm + c = 0 \quad (3.6)$$

resolviendo esta ecuación cuadrática para m obtenemos las dos soluciones

$$m_1 = \frac{(-b + \sqrt{b^2 - 4ac})}{2a}, \quad m_2 = \frac{(-b - \sqrt{b^2 - 4ac})}{2a} \quad (3.7)$$

de donde es evidente la importancia de los coeficientes de la Ec. (3.2), ya que el signo del discriminante ($b^2 - 4ac$) es crucial para determinar el número y tipo de soluciones de la Ec. (3.6). Así, tenemos tres casos a considerar:

Caso I. ($b^2 - 4ac$) > 0, **Ecuación Hiperbólica.**

La Ec. (3.6) tiene dos soluciones reales distintas, m_1 y m_2 . Así cualquier función de cualquiera de los dos argumentos $m_1x + y$ ó $m_2x + y$ resuelven a la Ec. (3.2). Por lo tanto la solución general de la Ec. (3.2) es

$$u(x, y) = \mathcal{F}(m_1x + y) + \mathcal{G}(m_2x + y) \quad (3.8)$$

donde \mathcal{F} y \mathcal{G} son cualquier función de clase \mathbb{C}^2 . Un ejemplo de este tipo de ecuaciones es la ecuación de onda, cuya ecuación canónica es

$$\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial t^2} = 0. \quad (3.9)$$

Caso II. $(b^2 - 4ac) = 0$, **Ecuación Parabólica.**

Asumiendo que $b \neq 0$ y $a \neq 0$ (lo cual implica que $c \neq 0$). Entonces se tiene una sola raíz degenerada de la Ec. (3.6) con el valor de $m_1 = \frac{-b}{2a}$ que resuelve a la Ec. (3.2). Por lo tanto la solución general de la Ec. (3.2) es

$$u(x, y) = \mathcal{F}(m_1x + y) + y\mathcal{G}(m_1x + y) \quad (3.10)$$

donde \mathcal{F} y \mathcal{G} son cualquier función de clase \mathbb{C}^2 . Si $b = 0$ y $a = 0$, entonces la solución general es

$$u(x, y) = \mathcal{F}(x) + y\mathcal{G}(x) \quad (3.11)$$

la cual es análoga si $b = 0$ y $c = 0$. Un ejemplo de este tipo de ecuaciones es la ecuación de difusión o calor, cuya ecuación canónica es

$$\frac{\partial^2 u}{\partial x^2} - \frac{\partial u}{\partial t} = 0. \quad (3.12)$$

Caso III. $(b^2 - 4ac) < 0$, **Ecuación Elíptica.**

La Ec. (3.6) tiene dos soluciones complejas m_1 y m_2 las cuales satisfacen que m_2 es el conjugado complejo de m_1 , es decir, $m_2 = \overline{m_1}$. La solución general puede ser escrita en la forma

$$u(x, y) = \mathcal{F}(m_1x + y) + \mathcal{G}(m_2x + y) \quad (3.13)$$

donde \mathcal{F} y \mathcal{G} son cualquier función de clase \mathbb{C}^2 . Un ejemplo de este tipo de ecuaciones es la ecuación de Laplace, cuya ecuación canónica es

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0. \quad (3.14)$$

Consideremos ahora el caso de un operador diferencial lineal de segundo orden definido en $\Omega \subset \mathbb{R}^n$ cuya forma general es

$$\mathcal{L}u = \sum_{i=1}^n \sum_{j=1}^n a_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=1}^n b_i \frac{\partial u}{\partial x_i} + cu \quad (3.15)$$

y consideremos también la ecuación homogénea asociada a este operador

$$\mathcal{L}u = 0 \quad (3.16)$$

además, sea $\underline{x} \in \Omega$ un punto del espacio Euclidiano y $V(\underline{x})$ una vecindad de ese punto. Sea una función u definida en $V(\underline{x})$ con la propiedad de que exista una variedad Σ de dimensión $n - 1$ cerrada y orientada, tal que la función u satisface la Ec. (3.16) en $V(\underline{x}) \setminus \Sigma$. Se supone además que existe un vector unitario \underline{n} que apunta en la dirección positiva (único) está definido en Σ . Además, la función u y sus derivadas de primer orden son continuas a través de Σ , mientras que los límites de las segundas derivadas de u existen por ambos lados de Σ . Sea $\underline{x} \in \Sigma$ tal que

$$\left[\frac{\partial^2 u}{\partial x_i \partial x_j}(\underline{x}) \right] \neq 0 \quad (3.17)$$

para alguna pareja $i, j = 1, \dots, n$. Entonces decimos que la función u es una solución débil de esta ecuación en \underline{x} .

Teorema 3 *Una condición necesaria para que existan soluciones débiles de la ecuación homogénea (3.16) en un punto $\underline{x} \in \Sigma$ es que*

$$\sum_{i=1}^n \sum_{j=1}^n a_{ij} n_i n_j = 0. \quad (3.18)$$

Así, si definimos a la matriz $\underline{\underline{A}} = (a_{ij})$ y observamos que

$$\underline{n} \cdot \underline{\underline{A}} \cdot \underline{n} = \sum_{i=1}^n \sum_{j=1}^n a_{ij} n_i n_j \quad (3.19)$$

entonces podemos decir que:

- I) Cuando todos los eigenvalores de la matriz $\underline{\underline{A}}$ son distintos de cero y además del mismo signo, entonces se dice que el operador es **Elíptico**.
- II) Cuando todos los eigenvalores de la matriz $\underline{\underline{A}}$ son distintos de cero y además $n - 1$ de ellos tienen el mismo signo, entonces se dice que el operador es **Hiperbólico**.
- III) Cuando uno y sólo uno de los eigenvalores de la matriz $\underline{\underline{A}}$ es igual a cero, entonces se dice que el operador es **Parabólico**.

Para el caso en que $n = 2$, esta forma de clasificación coincide con la dada anteriormente.

3.2 Condiciones Iniciales y de Frontera

Dado un problema concreto de ecuaciones en derivadas parciales sobre un dominio Ω , si la solución existe, esta no es única ya que generalmente este tiene un número infinito de soluciones. Para que el problema tenga una y sólo una solución es necesario imponer condiciones auxiliares apropiadas y estas son las condiciones iniciales y condiciones de frontera.

En esta sección sólo se enuncian de manera general las condiciones iniciales y de frontera que son esenciales para definir un problema de ecuaciones diferenciales:

A) Condiciones Iniciales

Las condiciones iniciales expresan el valor de la función al tiempo inicial $t = 0$ (t puede ser fijada en cualquier valor)

$$u(\underline{x}, \underline{y}, 0) = \gamma(\underline{x}, \underline{y}). \quad (3.20)$$

B) Condiciones de Frontera

Las condiciones de frontera especifican los valores que la función $u(\underline{x}, \underline{y}, t)$ o $\nabla u(\underline{x}, \underline{y}, t)$ tomarán en la frontera $\partial\Omega$, siendo de tres tipos posibles:

1) Condiciones tipo Dirichlet

Especifica los valores que la función $u(\underline{x}, \underline{y}, t)$ toma en la frontera $\partial\Omega$

$$u(\underline{x}, \underline{y}, t) = \gamma(\underline{x}, \underline{y}). \quad (3.21)$$

2) Condiciones tipo Neumann

Aquí se conoce el valor de la derivada de la función $u(\underline{x}, \underline{y}, t)$ con respecto a la normal \underline{n} a lo largo de la frontera $\partial\Omega$

$$\nabla u(\underline{x}, \underline{y}, t) \cdot \underline{n} = \gamma(\underline{x}, \underline{y}). \quad (3.22)$$

3) Condiciones tipo Robin

esta condición es una combinación de las dos anteriores

$$\alpha(\underline{x}, \underline{y})u(\underline{x}, \underline{y}, t) + \beta(\underline{x}, \underline{y})\nabla u(\underline{x}, \underline{y}, t) \cdot \underline{n} = g_{\partial}(\underline{x}, \underline{y}) \quad (3.23)$$

$$\forall \underline{x}, \underline{y} \in \partial\Omega.$$

En un problema dado se debe prescribir las condiciones iniciales al problema y debe existir alguno de los tipos de condiciones de frontera o combinación de ellas en $\partial\Omega$.

3.3 Modelos Completos

Los modelos de los sistemas continuos están constituidos por:

- Una colección de propiedades intensivas o lo que es lo mismo, extensivas.
- El conjunto de ecuaciones de balance local correspondientes (diferenciales y de salto).
- Suficientes relaciones que ligen a las propiedades intensivas entre sí y que definan a g , $\underline{\tau}$ y \underline{v} en términos de estas, las cuales se conocen como leyes constitutivas.

Una vez que se han planteado las ecuaciones que gobiernan al problema, las condiciones iniciales, de frontera y mencionado los procesos que intervienen de manera directa en el fenómeno estudiado, necesitamos que nuestro modelo sea *completo*. Decimos que el modelo de un sistema es *completo* si define un problema *bien planteado*. Un problema de valores iniciales y condiciones de frontera es *bien planteado* si cumple que:

- i) Existe una y sólo una solución y,
- ii) La solución depende de manera continua de las condiciones iniciales y de frontera del problema.

Es decir, un modelo completo es aquél en el cual se incorporan condiciones iniciales y de frontera que definen conjuntamente con las ecuaciones diferenciales un problema bien planteado.

A las ecuaciones diferenciales definidas en $\Omega \subset \mathbb{R}^n$

$$\begin{aligned} \Delta u &= 0 \\ \frac{\partial^2 u}{\partial t^2} - \Delta u &= 0 \\ \frac{\partial u}{\partial t} - \Delta u &= 0 \end{aligned} \tag{3.24}$$

se les conoce con los nombres de ecuación de Laplace, ecuación de onda y ecuación del calor, respectivamente. Cuando se considera la primera de estas

ecuaciones, se entiende que u es una función del vector $x \equiv (x_1, \dots, x_n)$, mientras que cuando se considera cualquiera de las otras dos, u es una función del vector $x \equiv (x_1, \dots, x_n, t)$. Así, en estos últimos casos el número de variables independientes es $n + 1$ y los conceptos relativos a la clasificación y las demás nociones discutidas con anterioridad deben aplicarse haciendo la sustitución $n \rightarrow n + 1$ e identificando $x_{n+1} = t$.

Ecuación de Laplace Para la ecuación de Laplace consideraremos condiciones del tipo Robin. En particular, condiciones de Dirichlet y condiciones de Neumann. Sin embargo, en este último caso, la solución no es única pues cualquier función constante satisface la ecuación de Laplace y también $\frac{\partial u}{\partial n} = g_\partial$ con $g_\partial = 0$.

Ecuación de Onda Un problema general importante consiste en obtener la solución de la ecuación de onda, en el dominio del espacio-tiempo $\Omega \times [0, t]$, que satisface para cada $t \in (0, t]$ una condición de frontera de Robin en $\partial\Omega$ y las condiciones iniciales

$$u(\underline{x}, 0) = u_0(\underline{x}) \quad \text{y} \quad \frac{\partial u}{\partial t}(\underline{x}, 0) = v_0(\underline{x}), \quad \forall \underline{x} \in \Omega \quad (3.25)$$

aquí $u_0(\underline{x})$ y $v_0(\underline{x})$ son dos funciones prescritas. El hecho de que para la ecuación de onda se prescriban los valores iniciales, de la función y su derivada con respecto al tiempo, es reminiscente de que en la mecánica de partículas se necesitan las posiciones y las velocidades iniciales para determinar el movimiento de un sistema de partículas.

Ecuación de Calor También para la ecuación del calor un problema general importante consiste en obtener la solución de la ecuación de onda, en el dominio del espacio-tiempo $\Omega \times [0, t]$, que satisface para cada $t \in (0, t]$ una condición de frontera de Robin y ciertas condiciones iniciales. Sin embargo, en este caso en ellas sólo se prescribe a la función

$$u(\underline{x}, 0) = u_0(\underline{x}), \quad \forall \underline{x} \in \Omega. \quad (3.26)$$

4 Análisis Funcional y Problemas Variacionales

En este capítulo se detallan los conceptos básicos de análisis funcional y problemas variacionales con énfasis en problemas elípticos de orden par $2m$, para comenzar detallaremos lo que entendemos por un operador diferencial parcial elíptico de orden par $2m$ en n variables, para después definir a los espacios de Sobolev para poder tratar problemas variacionales con valor en la frontera.

En donde, restringiéndonos a problemas elípticos, contestaremos una cuestión central en la teoría de problemas elípticos con valores en la frontera, y está se relaciona con las condiciones bajo las cuales uno puede esperar que el problema tenga solución y esta es única, así como conocer la regularidad de la solución, para mayor referencia de estos resultados ver [2], [54], [55] y [65].

4.1 Operador Lineal Elíptico

Definición 4 Entenderemos por un dominio al conjunto $\Omega \subset \mathbb{R}^n$ que sea abierto y conexo.

Para poder expresar de forma compacta derivadas parciales de orden m o menor, usaremos la definición siguiente.

Definición 5 Sea \mathbb{Z}_+^n el conjunto de todas las n -duplas de enteros no negativos, un miembro de \mathbb{Z}_+^n se denota usualmente por α ó β (por ejemplo $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$). Denotaremos por $|\alpha|$ la suma $|\alpha| = \alpha_1 + \alpha_2 + \dots + \alpha_n$ y por $D^\alpha u$ la derivada parcial

$$D^\alpha u = \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \dots \partial x_n^{\alpha_n}} \quad (4.1)$$

así, si $|\alpha| = m$, entonces $D^\alpha u$ denota la m -ésima derivada parcial de u .

Sea \mathcal{L} un operador diferencial parcial de orden par $2m$ en n variables y de la forma

$$\mathcal{L}u = \sum_{|\alpha|, |\beta| \leq m} (-1)^{|\alpha|} D^\alpha (a_{\alpha\beta}(\underline{x}) D^\beta u), \quad \underline{x} \in \Omega \subset \mathbb{R}^n \quad (4.2)$$

donde Ω es un dominio en \mathbb{R}^n . Los coeficientes $a_{\alpha\beta}$ son funciones suaves real valuadas de \underline{x} .

El operador \mathcal{L} es asumido que aparece dentro de una ecuación diferencial parcial con la forma

$$\mathcal{L}u = f, \quad (4.3)$$

donde f pertenece al rango del operador \mathcal{L} .

La clasificación del operador \mathcal{L} depende sólo de los coeficientes de la derivada más alta, esto es, de la derivada de orden $2m$, y a los términos involucrados en esa derivada son llamados la parte principal del operador \mathcal{L} denotado por \mathcal{L}_0 y para el operador (4.2) es de la forma

$$\mathcal{L}_0 = \sum_{|\alpha|, |\beta| \leq m} a_{\alpha\beta} D^{\alpha+\beta} u. \quad (4.4)$$

Teorema 6 Sea ξ un vector en \mathbb{R}^n , y sea $\xi^\alpha = \xi_1^{\alpha_1} \dots \xi_n^{\alpha_n}$, $\alpha \in \mathbb{Z}_n^+$. Entonces

i) \mathcal{L} es elíptico en $\underline{x}_o \in \Omega$, si

$$\sum_{|\alpha|, |\beta|=m} a_{\alpha\beta}(\underline{x}_o) \xi^{\alpha+\beta} \neq 0 \quad \forall \xi \neq 0; \quad (4.5)$$

ii) \mathcal{L} es elíptico, si es elíptico en todos los puntos de Ω ;

iii) \mathcal{L} es fuertemente elíptico, si existe un número $\mu > 0$ tal que

$$\left| \sum_{|\alpha|, |\beta|=m} a_{\alpha\beta}(\underline{x}_o) \xi^{\alpha+\beta} \right| \geq \mu |\xi|^{2m} \quad (4.6)$$

satisfaciéndose en todo punto $\underline{x}_o \in \Omega$, y para todo $\xi \in \mathbb{R}^n$. Aquí $|\xi| = (\xi_1^2 + \dots + \xi_n^2)^{\frac{1}{2}}$.

Para el caso en el cual \mathcal{L} es un operador de 2do orden ($m = 1$), la notación se simplifica, tomando la forma

$$\mathcal{L}u = - \sum_{i,j=1}^n \frac{\partial}{\partial x_i} \left(a_{ij}(\underline{x}) \frac{\partial u}{\partial x_j} \right) + \sum_{j=1}^n a_j \frac{\partial u}{\partial x_j} + a_0 u = f \quad (4.7)$$

en Ω .

Para coeficientes adecuados a_{ij} , a_j y a_0 la condición para conocer si el operador es elíptico, es examinado por la condición

$$\sum_{i,j=1}^n a_{ij}(\underline{x}_0) \xi_i \xi_j \neq 0 \quad \forall \xi \neq 0 \quad (4.8)$$

y para conocer si el operador es fuertemente elíptico, es examinado por la condición

$$\sum_{i,j=1}^n a_{ij}(\underline{x}_0) \xi_i \xi_j > \mu |\xi|^2. \quad (4.9)$$

4.2 Espacios de Sobolev

En esta subsección detallaremos algunos resultados de los espacios de Sobolev sobre el conjunto de números reales, en estos espacios son sobre los cuales trabajaremos tanto para plantear el problema elíptico como para encontrar la solución al problema. Primeramente definiremos lo que entendemos por un espacio L^2 .

Definición 7 Una función medible $u(\underline{x})$ definida sobre $\Omega \subset \mathbb{R}^n$ se dice que pertenece al espacio $L^2(\Omega)$ si

$$\int_{\Omega} |u(\underline{x})|^2 d\underline{x} < \infty \quad (4.10)$$

es decir, es integrable.

La definición de los espacios medibles, espacios L^p , distribuciones y derivadas de distribuciones están dados en el apéndice, estos resultados son la base para poder definir a los espacios de Sobolev.

Definición 8 El espacio de Sobolev de orden m , denotado por $H^m(\Omega)$, es definido

$$H^m(\Omega) = \{u \mid D^\alpha u \in L^2(\Omega) \quad \forall \alpha \text{ tal que } |\alpha| \leq m\}. \quad (4.11)$$

El producto escalar $\langle \cdot, \cdot \rangle$ de dos elementos u y $v \in H^m(\Omega)$ está dado por

$$\langle u, v \rangle_{H^m} = \int_{\Omega} \sum_{|\alpha| \leq m} (D^\alpha u) (D^\alpha v) d\underline{x} \text{ para } u, v \in H^m(\Omega). \quad (4.12)$$

Nota: Es común que el espacio $L^2(\Omega)$ sea denotado por $H^0(\Omega)$.

Un espacio completo con producto interior es llamado un espacio de Hilbert, un espacio normado y completo es llamado espacio de Banach. Y como todo producto interior define una norma, entonces todo espacio de Hilbert es un espacio de Banach.

Definición 9 La norma $\|\cdot\|_{H^m}$ inducida a partir del producto interior $\langle \cdot, \cdot \rangle_{H^m}$ queda definida por

$$\|u\|_{H^m}^2 = \langle u, u \rangle_{H^m} = \int_{\Omega} \sum_{|\alpha| \leq m} (D^\alpha u)^2 d\underline{x}. \quad (4.13)$$

Ahora, con norma $\|\cdot\|_{H^m}$, el espacio $H^m(\Omega)$ es un espacio de Hilbert, esto queda plasmado en el siguiente resultado.

Teorema 10 El espacio $H^m(\Omega)$ con la norma $\|\cdot\|_{H^m}$ es un espacio de Hilbert.

Ya que algunas de las propiedades de los espacios de Sobolev sólo son válidas cuando la frontera del dominio es suficientemente suave. Para describir al conjunto donde los espacios de Sobolev están definidos, es común pedirle algunas propiedades y así definimos lo siguiente.

Definición 11 Una función f definida sobre un conjunto $\Gamma \subset \mathbb{R}^n$ es llamada Lipschitz continua si existe una constante $L > 0$ tal que

$$|f(x) - f(y)| \leq L |x - y| \quad \forall x, y \in \Gamma. \quad (4.14)$$

Notemos que una función Lipschitz continua es uniformemente continua.

Sea $\Omega \subset \mathbb{R}^n$ ($n \geq 2$) un dominio con frontera $\partial\Omega$, sea $x_0 \in \partial\Omega$ y construyamos la bola abierta con centro en x_0 y radio ε , i.e. $B(x_0, \varepsilon)$, entonces definiremos el sistema coordenado (ξ_1, \dots, ξ_n) tal que el segmento $\partial\Omega \cap B(x_0, \varepsilon)$ pueda expresarse como una función

$$\xi_n = f(\xi_1, \dots, \xi_{n-1}) \quad (4.15)$$

entonces definimos.

Definición 12 La frontera $\partial\Omega$ del dominio Ω es llamada de Lipschitz si f definida como en la Ec. (4.15) es una función Lipschitz continua.

El siguiente teorema resume las propiedades más importantes de los espacios de Sobolev $H^m(\Omega)$.

Teorema 13 Sea $H^m(\Omega)$ el espacio de Sobolev de orden m y sea $\Omega \subset \mathbb{R}^n$ un dominio acotado con frontera Lipschitz. Entonces

- i) $H^r(\Omega) \subset H^m(\Omega)$ si $r \geq m$
- ii) $H^m(\Omega)$ es un espacio de Hilbert con respecto a la norma $\|\cdot\|_{H^m}$
- iii) $H^m(\Omega)$ es la cerradura con respecto a la norma $\|\cdot\|_{H^m}$ del espacio $C^\infty(\bar{\Omega})$.

De la parte *iii*) del teorema anterior, se puede hacer una importante interpretación: Para toda $u \in H^m(\Omega)$ es siempre posible encontrar una función infinitamente diferenciable f , tal que este arbitrariamente cerca de u en el sentido que

$$\|u - f\|_{H^m} < \varepsilon \quad (4.16)$$

para algún $\varepsilon > 0$ dado.

Cuando $m = 0$, se deduce la propiedad $H^0(\Omega) = L^2(\Omega)$ a partir del teorema anterior.

Corolario 14 El espacio $L^2(\Omega)$ es la cerradura, con respecto a la norma L^2 , del espacio $C^\infty(\bar{\Omega})$.

Otra propiedad, se tiene al considerar a cualquier miembro de $u \in H^m(\Omega)$, este puede ser identificado con una función en $C^m(\bar{\Omega})$, después de que posiblemente sean cambiados algunos valores sobre un conjunto de medida cero, esto queda plasmado en los dos siguientes resultados.

Teorema 15 Sean X y Y dos espacios de Banach, con $X \subset Y$. Sea $f : X \rightarrow Y$ tal que $f(u) = u$. Si el espacio X tiene definida la norma $\|\cdot\|_X$ y el espacio Y tiene definida la norma $\|\cdot\|_Y$, decimos que X está inmersa continuamente en Y si

$$\|f(u)\|_Y = \|u\|_Y \leq K \|u\|_X \quad (4.17)$$

para alguna constante $K > 0$.

Teorema 16 (Inmersión de Sobolev)

Sea $\Omega \subset \mathbb{R}^n$ un dominio acotado con frontera $\partial\Omega$ de Lipschitz. Si $(m - k) > n/2$, entonces toda función en $H^m(\Omega)$ pertenece a $C^k(\bar{\Omega})$, es decir, hay un miembro que pertenece a $C^k(\bar{\Omega})$. Además, la inmersión

$$H^m(\Omega) \subset C^k(\bar{\Omega}) \quad (4.18)$$

es continua.

4.2.1 Trazas de una Función en $H^m(\Omega)$.

Una parte fundamental en los problemas con valores en la frontera definidos sobre el dominio Ω , es definir de forma única los valores que tomará la función sobre la frontera $\partial\Omega$, en este apartado veremos bajo qué condiciones es posible tener definidos de forma única los valores en la frontera $\partial\Omega$ tal que podamos definir un operador $tr(\cdot)$ continuo que actúe en $\bar{\Omega}$ tal que $tr(u) = u|_{\partial\Omega}$.

El siguiente lema nos dice que el operador $tr(\cdot)$ es un operador lineal continuo de $C^1(\bar{\Omega})$ a $C(\partial\Omega)$, con respecto a las normas $\|\cdot\|_{H^1(\Omega)}$ y $\|\cdot\|_{L^2(\partial\Omega)}$.

Lema 17 *Sea Ω un dominio con frontera $\partial\Omega$ de Lipschitz. La estimación*

$$\|tr(u)\|_{L^2(\partial\Omega)} \leq C \|u\|_{H^1(\Omega)} \quad (4.19)$$

se satisface para toda función $u \in C^1(\bar{\Omega})$, para alguna constante $C > 0$.

Ahora, para el caso $tr(\cdot) : H^1(\Omega) \rightarrow L^2(\partial\Omega)$, se tiene el siguiente teorema.

Teorema 18 *Sea Ω un dominio acotado en \mathbb{R}^n con frontera $\partial\Omega$ de Lipschitz. Entonces:*

i) Existe un único operador lineal acotado $tr(\cdot) : H^1(\Omega) \rightarrow L^2(\partial\Omega)$, tal que

$$\|tr(u)\|_{L^2(\partial\Omega)} \leq C \|u\|_{H^1(\Omega)}, \quad (4.20)$$

con la propiedad que si $u \in C^1(\bar{\Omega})$, entonces $tr(u) = u|_{\partial\Omega}$.

ii) El rango de $tr(\cdot)$ es denso en $L^2(\partial\Omega)$.

El argumento anterior puede ser generalizado para los espacios $H^m(\Omega)$, de hecho, cuando $m > 1$, entonces para toda $u \in H^m(\Omega)$ tenemos que

$$D^\alpha u \in H^1(\Omega) \quad \text{para } |\alpha| \leq m - 1, \quad (4.21)$$

por el teorema anterior, el valor de $D^\alpha u$ sobre la frontera está bien definido y pertenece a $L^2(\Omega)$, es decir

$$tr(D^\alpha u) \in L^2(\Omega), \quad |\alpha| \leq m - 1. \quad (4.22)$$

Además, si u es m -veces continuamente diferenciable, entonces $D^\alpha u$ es al menos continuamente diferenciable para $|\alpha| \leq m - 1$ y

$$tr(D^\alpha u) = (D^\alpha u)|_{\partial\Omega}. \quad (4.23)$$

4.2.2 Espacios $H_0^m(\Omega)$.

Los espacio $H_0^m(\Omega)$ surgen comúnmente al trabajar con problemas con valor en la frontera y serán aquellos espacios que se nulifiquen en la frontera del dominio, es decir

Definición 19 *Definimos a los espacios $H_0^m(\Omega)$ como la cerradura, en la norma de Sobolev $\|\cdot\|_{H^m}$, del espacio $C_0^m(\Omega)$ de funciones con derivadas continuas del orden menor que m , todas las cuales tienen soporte compacto en Ω , es decir $H_0^m(\Omega)$ es formado al tomar la unión de $C_0^m(\Omega)$ y de todos los límites de sucesiones de Cauchy en $C_0^m(\Omega)$ que no pertenecen a $C_0^m(\Omega)$.*

Las propiedades básicas de estos espacios están contenidas en el siguiente resultado.

Teorema 20 *Sea Ω un dominio acotado en \mathbb{R}^n con frontera $\partial\Omega$ suficientemente suave y sea $H_0^m(\Omega)$ la cerradura de $C_0^\infty(\Omega)$ en la norma $\|\cdot\|_{H^m}$, entonces*

- a) $H_0^m(\Omega)$ es la cerradura de $C_0^\infty(\Omega)$ en la norma $\|\cdot\|_{H^m}$;
- b) $H_0^m(\Omega) \subset H^m(\Omega)$;
- c) Si $u \in H^m(\Omega)$ pertenece a $H_0^m(\Omega)$, entonces

$$D^\alpha u = 0, \text{ sobre } \partial\Omega, |\alpha| \leq m - 1. \quad (4.24)$$

Teorema 21 *(Desigualdad de Poincaré-Friedrichs)*

Sea Ω un dominio acotado en \mathbb{R}^n . Entonces existe una constante $C > 0$ tal que

$$\int_{\Omega} |u|^2 dx \leq C \int_{\Omega} |\nabla u|^2 dx \quad (4.25)$$

para toda $u \in H_0^1(\Omega)$.

Introduciendo ahora una familia de semi-normas sobre $H^m(\Omega)$ (una semi-norma $|\cdot|$ satisface casi todos los axiomas de una norma excepto el de positivo definido), de la siguiente forma:

Definición 22 *La semi-norma $|\cdot|_m$ sobre $H^m(\Omega)$, se define como*

$$|u|_m^2 = \sum_{|\alpha|=m} \int_{\Omega} |D^\alpha u|^2 dx. \quad (4.26)$$

Esta es una semi-norma, ya que $|u|_m = 0$ implica que $D^\alpha u = 0$ para $|\alpha| = m$, lo cual no implica que $u = 0$.

La relevancia de esta semi-norma está al aplicar la desigualdad de Poincaré-Friedrichs ya que es posible demostrar que $|\cdot|_1$ es de hecho una norma sobre $H_0^1(\Omega)$.

Corolario 23 *La semi-norma $|\cdot|_1$ es una norma sobre $H_0^1(\Omega)$, equivalente a la norma estándar $\|\cdot\|_{H^1}$.*

Es posible extender el teorema anterior y su corolario a los espacios $H_0^m(\Omega)$ para cualquier $m \geq 1$, de la siguiente forma:

Teorema 24 *Sea Ω un dominio acotado en \mathbb{R}^n . Entonces existe una constante $C > 0$ tal que*

$$\|u\|_{L^2}^2 \leq C |u|_m^2 \quad (4.27)$$

para toda $u \in H_0^m(\Omega)$, además, $|\cdot|_m$ es una norma sobre $H_0^m(\Omega)$ equivalente a la norma estándar $\|\cdot\|_{H^m}$.

Definición 25 *Sea Ω un dominio acotado en \mathbb{R}^n . Definimos por $H^{-m}(\Omega)$ al espacio de todas las funcionales lineales acotadas sobre $H_0^m(\Omega)$, es decir, $H^{-m}(\Omega)$ será el espacio dual del espacio $H_0^m(\Omega)$.*

Teorema 26 *q será una distribución de $H^{-m}(\Omega)$ si y sólo si q puede ser expresada en la forma*

$$q = \sum_{|\alpha| < m} D^\alpha q_\alpha \quad (4.28)$$

donde q_α son funcionales en $L^2(\Omega)$.

Algunos Comentarios y Precisiones Sea Ω un dominio tal que la frontera $\partial\Omega$ es suficientemente suave (considerando sólo frontera Lipschitz continua), entonces existe un operador $\gamma_0 : H^1(\Omega) \rightarrow L^2(\Omega)$ lineal y continuo, tal que $\gamma_0 v = \text{tr}(v)$ sobre $\partial\Omega$ para toda v suave (por ejemplo $v \in C^1(\overline{\Omega})$), un análisis más profundo muestra que tomando las trazas de todas las funciones de $H^1(\Omega)$ uno no obtiene el espacio completo de $L^2(\Omega)$, sólo obtiene un subespacio de este. Tenemos entonces

$$H^1(\partial\Omega) \subset \gamma_0(H^1(\Omega)) \subset L^2(\partial\Omega) \equiv H^0(\partial\Omega) \quad (4.29)$$

donde cada inclusión es estricta.

Finalmente reconocemos que el espacio $\gamma_0(H^1(\Omega))$ pertenece a la familia de espacios $H^s(\partial\Omega)$ y corresponde exactamente a el valor de $s = 1/2$. De tal forma que

$$H^{1/2}(\partial\Omega) = \gamma_0(H^1(\Omega)) \quad (4.30)$$

con

$$\|g\|_{H^{1/2}(\partial\Omega)} = \inf_{v \in H^1(\Omega) \text{ y } \gamma_0 v = g} \|v\|_{H^1(\Omega)} \quad (4.31)$$

de forma similar, se ve que las trazas de las funciones en $H^2(\Omega)$ pertenecen a el espacio $H^s(\partial\Omega)$ para $s = 3/2$, por lo tanto tenemos que

$$H^{3/2}(\partial\Omega) = \gamma_0(H^2(\Omega)) \quad (4.32)$$

$$\|g\|_{H^{3/2}(\partial\Omega)} = \inf_{v \in H^2(\Omega) \text{ y } \gamma_0 v = g} \|v\|_{H^2(\Omega)}. \quad (4.33)$$

Esto puede generalizarse a las trazas de derivadas de orden alto. Por ejemplo, si la frontera $\partial\Omega$ es suficientemente suave, podemos definir $\frac{\partial v}{\partial n}|_{\partial\Omega} \in H^{1/2}(\partial\Omega)$ para $v \in H^2(\Omega)$.

Por otro lado, para ejemplificar algunos casos de H_0^m notemos que

$$\begin{aligned} H_0^1(\Omega) &= \{v \mid v \in H^1(\Omega), v|_{\partial\Omega} = 0\} \\ H_0^2(\Omega) &= \left\{v \mid v \in H^2(\Omega), v|_{\partial\Omega} = 0 \text{ y } \frac{\partial v}{\partial n}|_{\partial\Omega} = 0\right\}. \end{aligned} \quad (4.34)$$

Además, en algunas ocasiones necesitamos considerar funciones que se nulifican en alguna parte de la frontera, supongamos que $\partial\Omega = D \cup N$, donde D es frontera tipo Dirichlet y N es frontera tipo Neumann y $D \cap N = \emptyset$, entonces podemos definir

$$H_{0,D}^1(\Omega) = \{v \mid v \in H^1(\Omega), v|_D = 0\} \quad (4.35)$$

y donde tenemos que $H_0^1(\Omega) \subset H_{0,D}^1(\Omega) \subset H^1(\Omega)$.

4.2.3 Espacios $H(\text{div}, \Omega)$

Definición 27 Sea Ω un dominio acotado en \mathbb{R}^n . Definimos a $(L^2(\Omega))^n$ al espacio

$$(L^2(\Omega))^n = \{\text{grad } H^1(\Omega)\} \oplus \{\text{rot } H_0^1(\Omega)\}. \quad (4.36)$$

Definición 28 Sea Ω un dominio acotado en \mathbb{R}^n . Definimos por $H(\text{div}, \Omega)$ al espacio

$$H(\text{div}, \Omega) = \{ \underline{q} \mid \underline{q} \in (L^2(\Omega))^n, \text{div } \underline{q} \in L^2(\Omega) \}. \quad (4.37)$$

Definición 29 La norma $\|\cdot\|_{H(\text{div}, \Omega)}^2$ de $H(\text{div}, \Omega)$, se define como

$$\|\underline{q}\|_{H(\text{div}, \Omega)}^2 = \|\underline{q}\|_{0, \Omega}^2 + \|\text{div } \underline{q}\|_{0, \Omega}^2. \quad (4.38)$$

Cuando $H(\text{div}, \Omega)$ es equipada con la norma $\|\cdot\|_{H(\text{div}, \Omega)}^2$ el correspondiente producto interior se convierte en un espacio de Hilbert.

Notemos que, si Ω es un dominio acotado en \mathbb{R}^n , con frontera suave $\partial\Omega$, si \underline{n} es un vector normal a $\partial\Omega$ y sea $\underline{q} \in H(\text{div}, \Omega)$, entonces los vectores de $H(\text{div}, \Omega)$ admiten una norma de la traza sobre $\partial\Omega$. Esta norma de la traza $\underline{q} \cdot \underline{n}$ pertenece a $H^{-1/2}(\partial\Omega)$ y esto se sigue de la fórmula de integración por partes

$$\int_{\Omega} \underline{q} \cdot \text{grad } v dx + \int_{\Omega} \text{div } \underline{q} v dx = \langle v, \underline{q} \cdot \underline{n} \rangle_{H^{1/2}(\partial\Omega) \times H^{-1/2}(\partial\Omega)} \quad (4.39)$$

para toda $\underline{q} \in H(\text{div}, \Omega)$ y cualquier $v \in H^1(\Omega)$. Pudiendo escribir formalmente $\int_{\partial\Omega} v \underline{q} \cdot \underline{n} ds$ en lugar del producto dual $\langle v, \underline{q} \cdot \underline{n} \rangle$.

Lema 30 Sea $\underline{q} \in H(\text{div}, \Omega)$, podemos definir $\underline{q} \cdot \underline{n}|_{\partial\Omega} \in H^{-1/2}(\partial\Omega)$ y por la fórmula de Green

$$\int_{\Omega} \text{div } \underline{q} v dx + \int_{\Omega} \underline{q} \cdot \text{grad } v dx = \langle v, \underline{q} \cdot \underline{n} \rangle \quad (4.40)$$

para toda $v \in H^1(\Omega)$.

Lema 31 La traza del operador $\underline{q} \in H(\text{div}, \Omega) \rightarrow \underline{q} \cdot \underline{n}|_{\partial\Omega} \in H^{-1/2}(\partial\Omega)$ es suprayectiva.

Sea Ω un dominio con frontera suave $\partial\Omega$, además supongamos que es frontera tipo Neumann $N = \partial\Omega$, entonces podemos definir

$$H_{0,N}(\text{div}, \Omega) = \{ \underline{q} \mid \underline{q} \in H(\text{div}, \Omega), \langle v, \underline{q} \cdot \underline{n} \rangle = 0, \forall v \in H_{0,D}^1(\Omega) \}. \quad (4.41)$$

este espacio contiene funciones del espacio $H(\text{div}, \Omega)$ cuyas trazas normales se nulifican en la frontera N .

Definición 32 *Un subespacio importante de $H(\operatorname{div}, \Omega)$ es $N^0(\operatorname{div}, \Omega)$, que se define como*

$$N^0(\operatorname{div}, \Omega) = \{ \underline{q} \mid \underline{q} \in H(\operatorname{div}, \Omega), \operatorname{div} \underline{q} = 0 \}. \quad (4.42)$$

Lema 33 *El operador de traza normal $\underline{q} \rightarrow \underline{q} \cdot \underline{n}|_{\partial\Omega}$ es una forma suprayectiva $N^0(\operatorname{div}, \Omega)$ sobre $\{ \mu \mid \mu \in H^{-1/2}(\partial\Omega), \langle \mu, 1 \rangle = 0 \}$.*

4.3 Fórmulas de Green y Problemas Adjuntos

Una cuestión central en la teoría de problemas elípticos con valores en la frontera se relaciona con las condiciones bajo las cuales uno puede esperar una única solución a problemas de la forma

$$\begin{aligned} \mathcal{L}u &= f_\Omega \quad \text{en } \Omega \subset \mathbb{R}^n \\ &\left. \begin{aligned} B_0 u &= g_0 \\ B_1 u &= g_1 \\ &\vdots \\ B_{m-1} u &= g_{m-1} \end{aligned} \right\} \quad \text{en } \partial\Omega \end{aligned} \quad (4.43)$$

donde \mathcal{L} es un operador elíptico de orden $2m$, de forma

$$\mathcal{L}u = \sum_{|\alpha| \leq m} (-1)^{|\alpha|} D^\alpha \left(\sum_{|\beta| \leq m} a_{\alpha\beta}(\underline{x}) D^\beta u \right), \quad \underline{x} \in \Omega \subset \mathbb{R}^n \quad (4.44)$$

donde los coeficientes $a_{\alpha\beta}$ son funciones de \underline{x} suaves y satisfacen las condiciones para que el operador sea elíptico, el conjunto B_0, B_1, \dots, B_{m-1} de operadores de frontera son de la forma

$$B_j u = \sum_{|\alpha| \leq q_j} b_\alpha^{(j)} D^\alpha u = g_j \quad (4.45)$$

y constituyen un conjunto de condiciones de frontera que cubren a \mathcal{L} . Los coeficientes $b_\alpha^{(j)}$ son asumidos como funciones suaves.

En el caso de problemas de segundo orden la Ec. (4.45) puede expresarse como una sola condición de frontera

$$Bu = \sum_{j=1}^n b_j \frac{\partial u}{\partial x_j} + cu = g \quad \text{en } \partial\Omega. \quad (4.46)$$

Antes de poder ver las condiciones bajo las cuales se garantiza la existencia y unicidad es necesario introducir el concepto de fórmula de Green asociada con el operador \mathcal{L}^* , para ello definimos:

Definición 34 Con el operador dado como en la Ec. (4.44), denotaremos por \mathcal{L}^* al operador definido por

$$\mathcal{L}^*u = \sum_{|\alpha| \leq m} (-1)^{|\alpha|} D^\alpha \left(\sum_{|\beta| \leq m} a_{\beta\alpha}(\underline{x}) D^\beta u \right) \quad (4.47)$$

y nos referiremos a \mathcal{L}^* como el adjunto formal del operador \mathcal{L} .

La importancia del adjunto formal es que si aplicamos el teorema de Green (167) a la integral

$$\int_{\Omega} v \mathcal{L}u d\underline{x} \quad (4.48)$$

obtenemos

$$\int_{\Omega} v \mathcal{L}u d\underline{x} = \int_{\Omega} u \mathcal{L}^*v d\underline{x} + \int_{\partial\Omega} F(u, v) d\underline{s} \quad (4.49)$$

en la cual $F(u, v)$ representa términos de frontera que se nulifican al aplicar el teorema ya que la función $v \in H_0^1(\Omega)$. Si $\mathcal{L} = \mathcal{L}^*$; i.e. $a_{\alpha\beta} = a_{\beta\alpha}$ el operador es llamado de manera formal el auto-adjunto.

En el caso de problemas de segundo orden, dos sucesivas aplicaciones del teorema de Green (167) y obtenemos, para i y j fijos

$$\begin{aligned} - \int_{\Omega} v \frac{\partial}{\partial x_i} \left(a_{ij} \frac{\partial u}{\partial x_j} \right) d\underline{x} &= - \int_{\partial\Omega} v a_{ij} \frac{\partial u}{\partial x_j} n_i d\underline{s} + \int_{\Omega} a_{ij} \frac{\partial u}{\partial x_j} \frac{\partial v}{\partial x_i} d\underline{x} \\ &= - \int_{\partial\Omega} \left[v a_{ij} \frac{\partial u}{\partial x_j} n_i - u a_{ij} \frac{\partial v}{\partial x_i} n_j \right] d\underline{s} \\ &\quad - \int_{\Omega} u \frac{\partial}{\partial x_j} \left(a_{ij} \frac{\partial v}{\partial x_i} \right) d\underline{x}. \end{aligned} \quad (4.50)$$

Pero sumando sobre i y j , obtenemos de la Ec. (4.49)

$$\mathcal{L}^*v = - \sum_{i,j=1}^n \frac{\partial}{\partial x_i} \left(a_{ji}(\underline{x}) \frac{\partial v}{\partial x_j} \right) \quad (4.51)$$

y

$$F(u, v) = - \sum_{i,j=1}^n a_{ij} \left(v \frac{\partial u}{\partial x_j} n_i - u \frac{\partial v}{\partial x_i} n_j \right) \quad (4.52)$$

tal que \mathcal{L} es formalmente el auto-ajunto si $a_{ji} = a_{ij}$.

Para hacer el tratamiento más simple, restringiremos nuestra atención al problema homogéneo, es decir, en el cual $g_0, g_1, \dots, g_{m-1} = 0$ (esta no es una restricción real, ya que se puede demostrar que cualquier problema no-homogéneo con condiciones de frontera puede convertirse en uno con condiciones de frontera homogéneo de una manera sistemática), asumiremos también que Ω es suave y la frontera $\partial\Omega$ de Ω es de clase C^∞ .

Así, en lo que resta de la sección, daremos los pasos necesarios para poder conocer bajo que condiciones el problema elíptico con valores en la frontera del tipo

$$\begin{aligned} \mathcal{L}u &= f_\Omega \quad \text{en } \Omega \subset \mathbb{R}^n \\ &\left. \begin{aligned} B_0 u &= 0 \\ B_1 u &= 0 \\ &\vdots \\ B_{m-1} u &= 0 \end{aligned} \right\} \quad \text{en } \partial\Omega \end{aligned} \quad (4.53)$$

donde el operador \mathcal{L} y B_j están dados como en (4.44) y (4.45), con $s \geq 2m$ tiene solución y esta es única. Para ello, necesitamos adoptar el lenguaje de la teoría de operadores lineales, algunos resultados clave de álgebra lineal están detallados en el apéndice.

Primeramente denotemos $N(B_j)$ al espacio nulo del operador de frontera $B_j : H^s(\Omega) \rightarrow L^2(\Omega)$, entonces

$$N(B_j) = \{u \in H^s(\Omega) \mid B_j u = 0 \text{ en } \partial\Omega\} \quad (4.54)$$

para $j = 0, 1, 2, \dots, m-1$.

Adicionalmente definimos al dominio del operador \mathcal{L} , como el espacio

$$\begin{aligned} D(\mathcal{L}) &= H^s(\Omega) \cap N(B_0) \cap \dots \cap N(B_{m-1}) \\ &= \{u \in H^s(\Omega) \mid B_j u = 0 \text{ en } \partial\Omega, j = 0, 1, \dots, m-1\}. \end{aligned} \quad (4.55)$$

Entonces el problema elíptico con valores en la frontera de la Ec. (4.53) con $s \geq 2m$, puede reescribirse como, dado $\mathcal{L} : D(\mathcal{L}) \rightarrow H^{s-2m}(\Omega)$, hallar u que satisfaga

$$\mathcal{L}u = f_\Omega \quad \text{en } \Omega. \quad (4.56)$$

Lo primero que hay que determinar es el conjunto de funciones f_Ω en $H^{s-2m}(\Omega)$ para las cuales la ecuación anterior se satisface, i.e. debemos identificar el rango $R(\mathcal{L})$ del operador \mathcal{L} . Pero como nos interesa conocer bajo qué condiciones la solución u es única, entonces podemos definir el núcleo $N(\mathcal{L})$ del operador \mathcal{L} como sigue

$$\begin{aligned} N(\mathcal{L}) &= \{u \in D(\mathcal{L}) \mid \mathcal{L}u = 0\} \\ &= \{u \in H^s(\Omega) \mid \mathcal{L}u = 0 \text{ en } \Omega, B_j u = 0 \text{ en } \partial\Omega, j = 0, 1, \dots, m-1\}. \end{aligned} \quad (4.57)$$

Si el $N(\mathcal{L}) \neq \{0\}$, entonces no hay una única solución, ya que si u_0 es una solución, entonces $u_0 + w$ también es solución para cualquier $w \in N(\mathcal{L})$, ya que

$$\mathcal{L}(u_0 + w) = \mathcal{L}u_0 + \mathcal{L}w = \mathcal{L}u_0 = f_\Omega. \quad (4.58)$$

Así, los elementos del núcleo $N(\mathcal{L})$ de \mathcal{L} deberán ser excluidos del dominio $D(\mathcal{L})$ del operador \mathcal{L} , para poder asegurar la unicidad de la solución u .

Si ahora, introducimos el complemento ortogonal $N(\mathcal{L})^\perp$ del núcleo $N(\mathcal{L})$ del operador \mathcal{L} con respecto al producto interior L^2 , definiéndolo como

$$N(\mathcal{L})^\perp = \{v \in D(\mathcal{L}) \mid (v, w) = 0 \forall w \in N(\mathcal{L})\}. \quad (4.59)$$

De esta forma tenemos que

$$D(\mathcal{L}) = N(\mathcal{L}) \oplus N(\mathcal{L})^\perp \quad (4.60)$$

i.e. para toda $u \in D(\mathcal{L})$, u se escribe como $u = v + w$ donde $v \in N(\mathcal{L})^\perp$ y $w \in N(\mathcal{L})$. Además $N(\mathcal{L}) \cap N(\mathcal{L})^\perp = \{0\}$.

De forma similar, podemos definir los espacios anteriores para el problema adjunto

$$\begin{aligned} \mathcal{L}^*u &= f_\Omega \text{ en } \Omega \subset \mathbb{R}^n \\ &\left. \begin{array}{l} B_0^*u = 0 \\ B_1^*u = 0 \\ \vdots \\ B_{m-1}^*u = 0 \end{array} \right\} \text{ en } \partial\Omega \end{aligned} \quad (4.61)$$

y definimos

$$\begin{aligned} D(\mathcal{L}^*) &= H^s(\Omega) \cap N(B_0^*) \cap \dots \cap N(B_{m-1}^*) \\ &= \{u \in H^s(\Omega) \mid B_j^*u = 0 \text{ en } \partial\Omega, j = 0, 1, \dots, m-1\}. \end{aligned} \quad (4.62)$$

Entonces el problema elíptico con valores en la frontera de la Ec. (4.53) con $s \geq 2m$, puede reescribirse como, dado $\mathcal{L}^* : D(\mathcal{L}^*) \rightarrow H^{s-2m}(\Omega)$, hallar u que satisfaga

$$\mathcal{L}^*u = f_\Omega \quad \text{en } \Omega. \quad (4.63)$$

Definiendo para el operador \mathcal{L}^*

$$\begin{aligned} N(\mathcal{L}^*) &= \{u \in D(\mathcal{L}^*) \mid \mathcal{L}^*u = 0\} \\ &= \{u \in H^s(\Omega) \mid \mathcal{L}^*u = 0 \text{ en } \Omega, B_j^*u = 0 \text{ en } \partial\Omega, j = 0, 1, \dots, m-1\}. \end{aligned} \quad (4.64)$$

y

$$N(\mathcal{L}^*)^\perp = \{v \in D(\mathcal{L}^*) \mid (v, w)_{L^2} = 0 \ \forall w \in N(\mathcal{L}^*)\}. \quad (4.65)$$

Así, con estas definiciones, es posible ver una cuestión fundamental, esta es, conocer bajo qué condiciones el problema elíptico con valores en la frontera de la Ec. (4.53) con $s \geq 2m$ tiene solución y esta es única, esto queda resuelto en el siguiente teorema cuya demostración puede verse en [55] y [2].

Teorema 35 *Considerando el problema elíptico con valores en la frontera de la Ec. (4.53) con $s \geq 2m$ definido sobre un dominio Ω acotado con frontera $\partial\Omega$ suave. Entonces*

i) Existe al menos una solución si y sólo si $f \in N(\mathcal{L}^)^\perp$, esto es, si*

$$(f, v)_{L^2(\Omega)} = 0 \quad \forall v \in N(\mathcal{L}^*). \quad (4.66)$$

ii) Asumiendo que la solución u existe, esta es única si $u \in N(\mathcal{L})^\perp$, esto es, si

$$(u, w)_{L^2(\Omega)} = 0 \quad \forall w \in N(\mathcal{L}). \quad (4.67)$$

iii) Si existe una única solución, entonces existe una única constante $C > 0$, independiente de u , tal que

$$\|u\|_{H^s} \leq C \|f\|_{H^{s-2m}}. \quad (4.68)$$

Observación 1 *i) El teorema afirma que el operador \mathcal{L} es un operador suprayectivo de $D(\mathcal{L})$ sobre el subespacio de funciones en H^{s-2m} que satisface (4.67). Además el operador \mathcal{L} es inyectivo si el dominio es restringido al espacio de funciones que satisfagan a (4.66).*

ii) La parte (iii) del teorema puede interpretarse como un resultado de regularidad, en el sentido en que se muestra

$$u \in H^{s-2m}(\Omega) \quad \text{si } f \in H^s(\Omega). \quad (4.69)$$

Así, formalmente podemos definir el adjunto formal de la siguiente manera

Definición 36 *Sea \mathcal{L} un Operador Diferencial, decimos que un operador \mathcal{L}^* es su adjunto formal si satisface la siguiente condición*

$$w\mathcal{L}u - u\mathcal{L}^*w = \nabla \cdot \underline{\mathcal{Q}}(u, w) \quad (3.1)$$

tal que las funciones u y w pertenecen a un espacio lineal. Aquí $\underline{\mathcal{Q}}(u, w)$ es una funcional bilineal que representa términos de frontera.

Ejemplos de Operadores Adjuntos Formales A continuación se muestra mediante ejemplos el uso de la definición de operadores adjuntos formales y la parte correspondiente a términos de frontera.

A) Operador de la derivada de orden cero

La derivada de orden cero de una función u es tal que

$$\frac{d^n u}{dx^n} = u \quad (4.70)$$

es decir, $n = 0$, sea el operador

$$\mathcal{L}u = u \quad (4.71)$$

de la definición de operador adjunto tenemos que

$$w\mathcal{L}u = u\mathcal{L}^*w + \nabla \cdot \underline{\mathcal{Q}}(u, w) \quad (4.72)$$

entonces el término izquierdo es

$$w\mathcal{L}u = wu \quad (4.73)$$

de aquí

$$u\mathcal{L}^*w = uw \quad (4.74)$$

por lo tanto el operador adjunto formal es

$$\mathcal{L}^*w = w \quad (4.75)$$

nótese que el operador es auto-adjunto.

B) Operador de la derivada de primer orden

La derivada de primer orden en términos del operador es

$$\mathcal{L}u = c \frac{du}{dx} \quad (4.76)$$

de la definición de operador adjunto tenemos

$$w\mathcal{L}u = u\mathcal{L}w + \nabla \cdot \underline{\mathfrak{D}}(u, w) \quad (4.77)$$

desarrollando el lado izquierdo

$$\begin{aligned} w\mathcal{L}u &= wc \frac{du}{dx} \\ &= \frac{d(wcu)}{dx} - u \frac{d(cw)}{dx} \\ &= \frac{d(wcu)}{dx} - uc \frac{dw}{dx} \end{aligned} \quad (4.78)$$

por lo tanto, el operador adjunto formal es

$$\mathcal{L}^*w = -c \frac{dw}{dx} \quad (4.79)$$

y los términos de frontera son

$$\mathfrak{D}(u, w) = wcu \quad (4.80)$$

C) Operador Elíptico

El operador elíptico más sencillo es el Laplaciano

$$\mathcal{L}u \equiv -\Delta u = -\frac{\partial}{\partial x_i} \left(\frac{\partial u}{\partial x_i} \right) \quad (4.81)$$

de la ecuación del operador adjunto formal tenemos

$$\begin{aligned} w\mathcal{L}u &= -w \frac{\partial}{\partial x_i} \left(\frac{\partial u}{\partial x_i} \right) \\ &= -\frac{\partial}{\partial x_i} \left(w \frac{\partial u}{\partial x_i} \right) + \frac{\partial u}{\partial x_i} \frac{\partial w}{\partial x_i} \\ &= -\frac{\partial}{\partial x_i} \left(w \frac{\partial u}{\partial x_i} \right) + \frac{\partial}{\partial x_i} \left(u \frac{\partial w}{\partial x_i} \right) - u \frac{\partial}{\partial x_i} \left(\frac{\partial w}{\partial x_i} \right) \\ &= \frac{\partial}{\partial x_i} \left(u \frac{\partial w}{\partial x_i} - w \frac{\partial u}{\partial x_i} \right) - u \frac{\partial}{\partial x_i} \left(\frac{\partial w}{\partial x_i} \right) \end{aligned} \quad (4.82)$$

entonces, el operador adjunto formal es

$$\mathcal{L}^*w = -u \frac{\partial}{\partial x_i} \left(\frac{\partial w}{\partial x_i} \right) \quad (4.83)$$

es decir, el operador es autoadjunto. Notemos que la función bilineal $\underline{\mathcal{Q}}(u, w)$ es

$$\underline{\mathcal{Q}}(u, w) = u \frac{\partial w}{\partial x_i} - w \frac{\partial u}{\partial x_i}. \quad (4.84)$$

D) Consideremos el operador diferencial elíptico más general de segundo orden

$$\mathcal{L}u = -\nabla \cdot (\underline{a} \cdot \nabla u) + \nabla \cdot (\underline{b}u) + cu \quad (4.85)$$

de la definición de operador adjunto formal tenemos que

$$w\mathcal{L}u = u\mathcal{L}^*w + \nabla \cdot \underline{\mathcal{Q}}(u, w) \quad (4.86)$$

desarrollando el lado derecho de la ecuación anterior

$$\begin{aligned} w\mathcal{L}u &= w \left(-\nabla \cdot (\underline{a} \cdot \nabla u) + \nabla \cdot (\underline{b}u) + cu \right) \\ &= -w\nabla \cdot (\underline{a} \cdot \nabla u) + w\nabla \cdot (\underline{b}u) + wcu \end{aligned} \quad (4.87)$$

aplicando la igualdad de divergencia a los dos primeros sumandos se tiene que la ecuación anterior es

$$\begin{aligned} w\mathcal{L}u &= -\nabla \cdot (w\underline{a} \cdot \nabla u) + \underline{a} \cdot \nabla u \cdot \nabla w + \nabla \cdot (w\underline{b}u) \\ &\quad - \underline{b}u \cdot \nabla w + wcu \\ &= -\nabla \cdot (w\underline{a} \cdot \nabla u) + \nabla \cdot (u\underline{a} \nabla w) - u\nabla \cdot (\underline{a} \cdot \nabla w) + \nabla \cdot (w\underline{b}u) \\ &\quad - \underline{b}u \cdot \nabla w + wcu \\ &= \nabla \cdot [\underline{a} (u\nabla w - w\nabla u)] + \nabla \cdot (w\underline{b}u) - u\nabla \cdot (\underline{a} \cdot \nabla w) \\ &\quad - \underline{b}u \cdot \nabla w + wcu \end{aligned} \quad (4.88)$$

reordenando términos se tiene

$$\begin{aligned} w\mathcal{L}u &= -u\nabla \cdot (\underline{a} \cdot \nabla w) - \underline{b} \cdot \nabla w + ucw + \\ &\quad \nabla \cdot [\underline{a} (u\nabla w - w\nabla u) + (w\underline{b}u)] \end{aligned} \quad (4.89)$$

por lo tanto, el operador adjunto formal es

$$\mathcal{L}^*w = -\nabla \cdot (\underline{a} \cdot \nabla w) - \underline{b} \cdot \nabla w + cw \quad (4.90)$$

y el término correspondiente a valores en la frontera es

$$\underline{\mathcal{Q}}(u, w) = \underline{a} \cdot (u \nabla w - w \nabla u) + (w \underline{b} u). \quad (4.91)$$

E) La ecuación bi-armónica

Consideremos el operador diferencial bi-armónico

$$\mathcal{L}u = \Delta^2 u \quad (4.92)$$

entonces se tiene que

$$w \mathcal{L}u = u \mathcal{L}^* w + \nabla \cdot \underline{\mathcal{Q}}(u, w) \quad (4.93)$$

desarrollemos el término del lado derecho

$$\begin{aligned} w \mathcal{L}u &= w \Delta^2 u \\ &= w \nabla \cdot (\nabla \Delta u) \end{aligned} \quad (4.94)$$

utilizando la igualdad de divergencia

$$\nabla \cdot (sV) = s \nabla \cdot V + V \cdot \nabla s \quad (4.95)$$

tal que s es función escalar y V vector, entonces sea $w = s$ y $\nabla \Delta u = V$, se tiene

$$\begin{aligned} w \nabla \cdot (\nabla \Delta u) \\ = \nabla \cdot (w \nabla \Delta u) - \nabla \Delta u \cdot \nabla w \end{aligned} \quad (4.96)$$

ahora sea $s = \Delta u$ y $V = \nabla w$, entonces

$$\begin{aligned} &\nabla \cdot (w \nabla \Delta u) - \nabla \Delta u \cdot \nabla w \\ = &\nabla \cdot (w \nabla \Delta u) + \Delta u \nabla \cdot \nabla w - \nabla \cdot (\Delta u \nabla w) \\ = &\Delta w \nabla \cdot \nabla u + \nabla \cdot (w \nabla \Delta u - \Delta u \nabla w) \end{aligned} \quad (4.97)$$

sea $s = \Delta w$ y $V = \nabla u$, entonces

$$\begin{aligned} &\Delta w \nabla \cdot \nabla u + \nabla \cdot (w \nabla \Delta u - \Delta u \nabla w) \\ = &\nabla \cdot (\Delta w \nabla u) - \nabla u \cdot \nabla (\Delta w) + \nabla \cdot (w \nabla \Delta u - \Delta u \nabla w) \\ = &-\nabla u \cdot \nabla (\Delta w) + \nabla \cdot (w \nabla \Delta u + \Delta w \nabla u - \Delta u \nabla w) \end{aligned} \quad (4.98)$$

por último sea $s = u$ y $V = \nabla (\Delta w)$ y obtenemos

$$\begin{aligned} & -\nabla u \cdot \nabla (\Delta w) + \nabla \cdot (w \nabla \Delta u + \Delta w \nabla u - \Delta u \nabla w) \\ = & u \nabla \cdot (\nabla (\Delta w)) - \nabla \cdot (u \nabla (\Delta w)) + \nabla \cdot (w \nabla \Delta u + \Delta w \nabla u - \Delta u \nabla w) \end{aligned} \quad (4.99)$$

reordenando términos

$$w \mathcal{L}u = u \Delta^2 w + \nabla \cdot (w \nabla \Delta u + \Delta w \nabla u - \Delta u \nabla w - u \nabla \Delta w) \quad (4.100)$$

entonces se tiene que el operador adjunto formal es

$$\mathcal{L}^* w = \Delta^2 w \quad (4.101)$$

y los términos de frontera son

$$\underline{\mathfrak{D}}(u, w) = w \nabla \Delta u + \Delta w \nabla u - \Delta u \nabla w - u \nabla \Delta w. \quad (4.102)$$

4.4 Adjuntos Formales para Sistemas de Ecuaciones

En esta sección trabajaremos con funciones vectoriales, para ello necesitamos plantear la definición de operadores adjuntos formales para este tipo de funciones.

Definición 37 Sea $\underline{\underline{\mathcal{L}}}$ un operador diferencial, decimos que un operador $\underline{\underline{\mathcal{L}}}$ es su adjunto formal si satisface la siguiente condición

$$\underline{w} \underline{\underline{\mathcal{L}}} \underline{u} - \underline{u} \underline{\underline{\mathcal{L}}}^* \underline{w} = \nabla \cdot \underline{\mathfrak{D}}(\underline{u}, \underline{w}) \quad (4.103)$$

tal que las funciones \underline{u} y \underline{w} pertenecen a un espacio lineal. Aquí $\underline{\mathfrak{D}}(\underline{u}, \underline{w})$ representa términos de frontera.

Por lo tanto se puede trabajar con funciones vectoriales utilizando operadores matriciales.

A) Operador diferencial vector-valuado con elasticidad estática

Sea

$$\underline{\underline{\mathcal{L}}} \underline{u} = -\nabla \cdot \underline{\underline{\underline{C}}} : \nabla \underline{u} \quad (4.104)$$

de la definición de operador adjunto formal tenemos que

$$\underline{w} \underline{\underline{\mathcal{L}}} \underline{u} = \underline{u} \underline{\underline{\mathcal{L}}}^* \underline{w} + \nabla \cdot \underline{\mathfrak{D}}(\underline{u}, \underline{w}) \quad (4.105)$$

para hacer el desarrollo del término del lado derecho se utilizará notación indicial, es decir, este vector $\underline{w}\underline{\mathcal{L}}\underline{u}$ tiene los siguientes componentes

$$-w_i \left(\frac{\partial}{\partial x_j} \left(C_{ijpq} \frac{\partial u_p}{\partial x_q} \right) \right); \quad i = 1, 2, 3 \quad (4.106)$$

utilizando la igualdad de divergencia tenemos

$$\begin{aligned} & -w_i \left(\frac{\partial}{\partial x_j} \left(C_{ijpq} \frac{\partial u_p}{\partial x_q} \right) \right) \\ &= C_{ijpq} \frac{\partial u_p}{\partial x_q} \frac{\partial w_i}{\partial x_j} - \frac{\partial}{\partial x_j} \left(w_i C_{ijpq} \frac{\partial u_p}{\partial x_q} \right) \\ &= \frac{\partial}{\partial x_j} \left(u_i C_{ijpq} \frac{\partial w_i}{\partial x_j} \right) - u_i \frac{\partial}{\partial x_j} \left(C_{ijpq} \frac{\partial w_i}{\partial x_j} \right) - \frac{\partial}{\partial x_j} \left(w_i C_{ijpq} \frac{\partial u_p}{\partial x_q} \right) \end{aligned} \quad (4.107)$$

reordenado términos tenemos que la ecuación anterior es

$$\frac{\partial}{\partial x_j} \left(u_i C_{ijpq} \frac{\partial w_i}{\partial x_j} - w_i C_{ijpq} \frac{\partial u_p}{\partial x_q} \right) - u_i \frac{\partial}{\partial x_j} \left(C_{ijpq} \frac{\partial w_i}{\partial x_j} \right) \quad (4.108)$$

en notación simbólica tenemos que

$$\underline{w} \underline{\mathcal{L}} \underline{u} = -\underline{u} \nabla \cdot \left(\underline{\underline{\underline{C}}} : \nabla \underline{w} \right) + \nabla \cdot \left(\underline{u} \cdot \underline{\underline{\underline{C}}} : \nabla \underline{w} - \underline{w} \cdot \underline{\underline{\underline{C}}} : \nabla \underline{u} \right) \quad (4.109)$$

por lo tanto el operador adjunto formal es

$$\underline{\underline{\underline{\mathcal{L}}}}^* \underline{w} = -\nabla \cdot \left(\underline{\underline{\underline{C}}} : \nabla \underline{w} \right) \quad (4.110)$$

y los términos de frontera son

$$\underline{\mathcal{D}}(\underline{u}, \underline{w}) = \underline{u} \cdot \underline{\underline{\underline{C}}} : \nabla \underline{w} - \underline{w} \cdot \underline{\underline{\underline{C}}} : \nabla \underline{u} \quad (4.111)$$

El operador de elasticidad es **auto-adjunto formal**.

B) Métodos Mixtos a la Ecuación de Laplace

Operador Laplaciano

$$\underline{\underline{\underline{\mathcal{L}}}} \underline{u} = \Delta \underline{u} = \underline{f} \quad (4.112)$$

escrito en un sistema de ecuaciones se obtiene

$$\underline{\underline{\mathcal{L}}}u = \begin{bmatrix} 1 & -\nabla \\ \nabla \cdot & 0 \end{bmatrix} \begin{bmatrix} \underline{p} \\ u \end{bmatrix} = \begin{bmatrix} 0 \\ f \end{bmatrix} \quad (4.113)$$

consideraremos campos vectoriales de 4 dimensiones, estos son denotados por :

$$\underline{u} \equiv \{ \underline{p}, u \} \text{ y } \underline{w} = \{ \underline{q}, w \} \quad (4.114)$$

ahora el operador diferencial vector-valorado es el siguiente

$$\begin{aligned} \underline{\underline{\mathcal{L}}}u &= \begin{bmatrix} 1 & -\nabla \\ \nabla \cdot & 0 \end{bmatrix} \cdot \begin{bmatrix} \underline{p} \\ u \end{bmatrix} \\ &= \begin{bmatrix} \underline{p} - \nabla u \\ \nabla \cdot \underline{p} \end{bmatrix} \end{aligned} \quad (4.115)$$

entonces

$$\underline{w} \underline{\underline{\mathcal{L}}}u = \begin{bmatrix} \underline{q} \\ w \end{bmatrix} \cdot \begin{bmatrix} 1 & -\nabla \\ \nabla \cdot & 0 \end{bmatrix} \cdot \begin{bmatrix} \underline{p} \\ u \end{bmatrix} \quad (4.116)$$

utilizando la definición de operador adjunto

$$\underline{w} \underline{\underline{\mathcal{L}}}u = u \underline{\underline{\mathcal{L}}}w + \nabla \cdot \underline{\mathfrak{D}}(u, \underline{w}) \quad (4.117)$$

haciendo el desarrollo del término izquierdo se tiene que

$$\begin{aligned} \underline{w} \underline{\underline{\mathcal{L}}}u &= \begin{bmatrix} \underline{q} \\ w \end{bmatrix} \cdot \begin{bmatrix} 1 & -\nabla \\ \nabla \cdot & 0 \end{bmatrix} \cdot \begin{bmatrix} \underline{p} \\ u \end{bmatrix} \\ &= \begin{bmatrix} \underline{q} \\ w \end{bmatrix} \cdot \begin{bmatrix} \underline{p} - \nabla u \\ \nabla \cdot \underline{p} \end{bmatrix} \\ &= \underline{qp} - \underline{q} \nabla \cdot u + w \nabla \cdot \underline{p} \end{aligned} \quad (4.118)$$

aquí se utiliza la igualdad de divergencia en los dos términos del lado derecho y obtenemos

$$\begin{aligned} &\underline{qp} - \underline{q} \nabla \cdot u + w \nabla \cdot \underline{p} \\ &= \underline{qp} + u \nabla \cdot \underline{q} - \nabla \cdot (\underline{qu}) - \underline{p} \cdot \nabla w + \nabla \cdot (w \underline{p}) \\ &= \underline{p} (\underline{q} - \nabla w) + u \nabla \cdot \underline{q} + \nabla \cdot (w \underline{p} - u \underline{q}) \end{aligned} \quad (4.119)$$

si se agrupa los dos primeros términos en forma matricial, se tiene

$$\begin{aligned}
 & \underline{p}(\underline{q} - \nabla w) + u \nabla \cdot \underline{q} + \nabla \cdot (w \underline{p} - u \underline{q}) \quad (4.120) \\
 &= \begin{bmatrix} \underline{p} \\ u \end{bmatrix} \begin{bmatrix} \underline{q} - \nabla w \\ w \end{bmatrix} + \nabla \cdot (w \underline{p} - u \underline{q}) \\
 &= \begin{bmatrix} \underline{p} \\ u \end{bmatrix} \cdot \begin{bmatrix} 1 & -\nabla \\ \nabla \cdot & 0 \end{bmatrix} \cdot \begin{bmatrix} \underline{q} \\ w \end{bmatrix} + \nabla \cdot (w \underline{p} - u \underline{q})
 \end{aligned}$$

por lo tanto, el operador adjunto formal es

$$\begin{aligned}
 \underline{\mathcal{L}}^* \underline{w} &= \begin{bmatrix} 1 & -\nabla \\ \nabla \cdot & 0 \end{bmatrix} \cdot \begin{bmatrix} \underline{q} \\ w \end{bmatrix} \quad (4.121) \\
 &= \begin{bmatrix} \underline{q} - \nabla w \\ \nabla \cdot \underline{q} \end{bmatrix}
 \end{aligned}$$

y el término correspondiente a valores en la frontera es

$$\underline{\mathcal{D}}(\underline{u}, \underline{w}) = w \underline{p} - u \underline{q}. \quad (4.122)$$

C) Problema de Stokes

El problema de Stokes es derivado de la ecuación de Navier-Stokes, la cual es utilizada en dinámica de fluidos viscosos. En este caso estamos suponiendo que el fluido es estacionario, la fuerza gravitacional es nula y el fluido incompresible. Entonces el sistema de ecuaciones a ser considerado es

$$\begin{aligned}
 -\Delta \underline{u} + \nabla p &= f \quad (4.123) \\
 -\nabla \cdot \underline{u} &= 0
 \end{aligned}$$

se considerará un campo vectorial de 4 dimensiones. Ellos serán denotados por

$$\underline{U} = \{\underline{u}, p\} \text{ y } \underline{W} = \{\underline{w}, q\} \quad (4.124)$$

ahora el operador diferencial vector-valorado es el siguiente

$$\underline{\mathcal{L}} \underline{U} = \begin{bmatrix} -\Delta & \nabla \\ -\nabla \cdot & 0 \end{bmatrix} \cdot \begin{bmatrix} \underline{u} \\ p \end{bmatrix} \quad (4.125)$$

el desarrollo se hará en notación indicial, entonces tenemos que

$$\underline{W} \underline{\mathcal{L}} \underline{U} = \begin{cases} -\underline{w} \Delta \underline{u} + \underline{w} \nabla p \\ -q \nabla \cdot \underline{u} \end{cases} \quad (4.126)$$

usando notación indicial se obtiene

$$\begin{aligned}
 w_i \left(- \sum_j \frac{\partial^2 u_i}{\partial x_j^2} + \frac{\partial p}{\partial x_i} \right) &= - \sum_j w_i \frac{\partial^2 u_i}{\partial x_j^2} + w_i \frac{\partial p}{\partial x_i} \\
 &= \sum_j \frac{\partial w_i}{\partial x_j} \frac{\partial u_i}{\partial x_j^2} - \sum_j \frac{\partial}{\partial x_j} \left(w_i \frac{\partial u_i}{\partial x_j} \right) - \\
 &\quad p \frac{\partial w_i}{\partial x_i} + \frac{\partial}{\partial x_i} (w_i p)
 \end{aligned} \tag{4.127}$$

desarrollando la primera suma como la derivada de dos funciones se tiene

$$\begin{aligned}
 - \sum_j u_i \frac{\partial^2 w_i}{\partial x_j^2} + \sum_j \frac{\partial}{\partial x_j} \left(u_i \frac{\partial w_i}{\partial x_j} \right) \\
 - \sum_j \frac{\partial}{\partial x_j} \left(w_i \frac{\partial u_i}{\partial x_j} \right) - p \frac{\partial w_i}{\partial x_i} + \frac{\partial}{\partial x_i} (w_i p)
 \end{aligned} \tag{4.128}$$

reordenando términos tenemos

$$\begin{aligned}
 - \sum_j u_i \frac{\partial^2 w_i}{\partial x_j^2} - p \frac{\partial w_i}{\partial x_i} + \\
 \sum_j \frac{\partial}{\partial x_j} \left(u_i \frac{\partial w_i}{\partial x_j} - w_i \frac{\partial u_i}{\partial x_j} \right) + \frac{\partial}{\partial x_i} (w_i p)
 \end{aligned} \tag{4.129}$$

Ahora consideremos la ecuación 2 en Ec. (4.126), tenemos

$$- q \nabla \cdot \underline{u} \tag{4.130}$$

en notación indicial se tiene

$$\begin{aligned}
 - q \sum_i \frac{\partial u_i}{\partial x_i} &= - \sum_i q \frac{\partial u_i}{\partial x_i} \\
 &= \sum_i u_i \frac{\partial q}{\partial x_i} - \sum_i \frac{\partial}{\partial x_i} (q u_i)
 \end{aligned} \tag{4.131}$$

en la ecuación anterior se utilizó la igualdad de divergencia, entonces

agrupando las ecuaciones Ec. (4.129) y Ec. (4.131) se tiene

$$\begin{aligned}
 & w_i \left(- \sum_j \frac{\partial^2 u_i}{\partial x_j^2} + \frac{\partial p}{\partial x_i} \right) - q \sum_i \frac{\partial u_i}{\partial x_i} \\
 &= - \sum_j u_i \frac{\partial^2 w_i}{\partial x_j^2} - p \frac{\partial w_i}{\partial x_i} + \\
 & \quad \sum_j \frac{\partial}{\partial x_j} \left(u_i \frac{\partial w_i}{\partial x_j} - w_i \frac{\partial u_i}{\partial x_j} \right) + \frac{\partial}{\partial x_i} (w_i p) + \\
 & \quad \sum_i u_i \frac{\partial q}{\partial x_i} - \sum_i \frac{\partial}{\partial x_i} (q u_i)
 \end{aligned} \tag{4.132}$$

ordenando los términos tenemos

$$\begin{aligned}
 & - \sum_j u_i \frac{\partial^2 w_i}{\partial x_j^2} + \sum_i u_i \frac{\partial q}{\partial x_i} - p \frac{\partial w_i}{\partial x_i} \\
 & + \sum_j \frac{\partial}{\partial x_j} \left(u_i \frac{\partial w_i}{\partial x_j} - w_i \frac{\partial u_i}{\partial x_j} \right) + \frac{\partial}{\partial x_i} (w_i p) - \sum_i \frac{\partial}{\partial x_i} (q u_i)
 \end{aligned} \tag{4.133}$$

escribiendo la ecuación anterior en notación simbólica, se obtiene

$$- \underline{u} \Delta \underline{w} + \underline{u} \nabla q - p \nabla \cdot \underline{w} + \nabla \cdot (\underline{u} \nabla \underline{w} - \underline{w} \nabla \underline{u} + \underline{w} p - \underline{u} q) \tag{4.134}$$

por lo tanto, el operador adjunto formal es

$$\underline{\mathcal{L}}^* \underline{W} = \begin{bmatrix} -\Delta & \nabla \\ -\nabla \cdot & 0 \end{bmatrix} \cdot \begin{bmatrix} \underline{w} \\ q \end{bmatrix} \tag{4.135}$$

y el término de valores de frontera es

$$\underline{\mathcal{Q}}(\underline{u}, \underline{w}) = \underline{u} \nabla \underline{w} - \underline{w} \nabla \underline{u} + \underline{w} p - \underline{u} q. \tag{4.136}$$

4.5 Problemas Variacionales con Valor en la Frontera

Restringiéndonos ahora en problemas elípticos de orden 2 (problemas de orden mayor pueden ser tratados de forma similar), reescribiremos este en su forma variacional. La formulación variacional es más débil que la formulación

convencional ya que esta demanda menor suavidad de la solución u , sin embargo cualquier problema variacional con valores en la frontera corresponde a un problema con valor en la frontera y viceversa.

Además, la formulación variacional facilita el tratamiento de los problemas al usar métodos numéricos de ecuaciones diferenciales parciales, en esta sección veremos algunos resultados clave como es la existencia y unicidad de la solución de este tipo de problemas, para mayores detalles, ver [55] y [2].

Si el operador \mathcal{L} está definido por

$$\mathcal{L}u = -\nabla \cdot \underline{a} \cdot \nabla u + cu \quad (4.137)$$

con \underline{a} una matriz positiva definida, simétrica y $c \geq 0$, el problema queda escrito como

$$\begin{aligned} -\nabla \cdot \underline{a} \cdot \nabla u + cu &= f_\Omega \quad \text{en } \Omega \\ u &= g \quad \text{en } \partial\Omega. \end{aligned} \quad (4.138)$$

Si multiplicamos a la ecuación $-\nabla \cdot \underline{a} \cdot \nabla u + cu = f_\Omega$ por $v \in V = H_0^1(\Omega)$, obtenemos

$$-v (\nabla \cdot \underline{a} \cdot \nabla u + cu) = v f_\Omega \quad (4.139)$$

aplicando el teorema de Green (167) obtenemos la Ec. (4.50), que podemos re-escribir como

$$\int_{\Omega} (\nabla v \cdot \underline{a} \cdot \nabla u + cuv) d\underline{x} = \int_{\Omega} v f_\Omega d\underline{x}. \quad (4.140)$$

Definiendo el operador bilineal

$$a(u, v) = \int_{\Omega} (\nabla v \cdot \underline{a} \cdot \nabla u + cuv) d\underline{x} \quad (4.141)$$

y la funcional lineal

$$l(v) = \langle f, v \rangle = \int_{\Omega} v f_\Omega d\underline{x} \quad (4.142)$$

podemos reescribir el problema dado por la Ec. (4.43) de orden 2, haciendo uso de la forma bilineal $a(\cdot, \cdot)$ y la funcional lineal $l(\cdot)$.

Entonces entenderemos en el presente contexto un problema variacional con valores de frontera (VBVP) por uno de la forma: hallar una función u que pertenezca a un espacio de Hilbert $V = H_0^1(\Omega)$ y que satisfaga la ecuación

$$a(u, v) = \langle f, v \rangle \quad (4.143)$$

para toda función $v \in V$ donde $a(\cdot, \cdot)$ es una forma bilineal y $l(\cdot)$ es una funcional lineal.

Definición 38 *Sea V un espacio de Hilbert y sea $\|\cdot\|_V$ la norma asociada a dicho espacio, decimos que una forma bilineal $a(\cdot, \cdot)$ es continua si existe una constante $M > 0$ tal que*

$$|a(u, v)| \leq M \|u\|_V \|v\|_V \quad \forall u, v \in V \quad (4.144)$$

y es V -elíptico si existe una constante $\alpha > 0$ tal que

$$a(v, v) \geq \alpha \|v\|_V^2 \quad \forall v \in V \quad (4.145)$$

donde $\|\cdot\|_V$ es la norma asociada al espacio V .

Esto significa que una forma V -elíptico es una que siempre es no negativa y toma el valor de 0 sólo en el caso de que $v = 0$, i.e. es positiva definida.

Notemos que el problema (4.138) definido en $V = H_0^1(\Omega)$ reescrito como el problema (4.143) genera una forma bilineal V -elíptico cuyo producto interior sobre V es simétrico y positivo definido ya que

$$a(v, v) \geq \alpha \|v\|_V^2 > 0, \quad \forall v \in V, v \neq 0 \quad (4.146)$$

reescribiéndose el problema (4.143), en el cual debemos encontrar $u \in V$ tal que

$$a(u, v) = \langle f, v \rangle - a(u_0, v) \quad (4.147)$$

donde $u_0 = g$ en $\partial\Omega$, para toda $v \in V$.

Entonces, la cuestión fundamental, es conocer bajo qué condiciones el problema anterior tiene solución y esta es única, el teorema de Lax-Milgram nos da las condiciones bajo las cuales el problema (4.138) reescrito como el problema (4.143) tiene solución y esta es única, esto queda plasmado en el siguiente resultado.

Teorema 39 (*Lax-Milgram*)

Sea V un espacio de Hilbert y sea $a(\cdot, \cdot) : V \times V \rightarrow \mathbb{R}$ una forma bilineal continua V -elíptico sobre V . Además, sea $l(\cdot) : V \rightarrow \mathbb{R}$ una funcional lineal continua sobre V . Entonces

i) El VBVP de encontrar $u \in V$ que satisfaga

$$a(u, v) = \langle f, v \rangle, \forall v \in V \quad (4.148)$$

tiene una y sólo una solución;

ii) La solución depende continuamente de los datos, en el sentido de que

$$\|u\|_V \leq \frac{1}{\alpha} \|l\|_{V^*} \quad (4.149)$$

donde $\|\cdot\|_{V^*}$ es la norma en el espacio dual V^* de V y α es la constante de la definición de V -elíptico.

Más específicamente, considerando ahora V un subespacio cerrado de $H^m(\Omega)$ las condiciones para la existencia, unicidad y la dependencia continua de los datos queda de manifiesto en el siguiente resultado.

Teorema 40 Sea V un subespacio cerrado de $H^m(\Omega)$, sea $a(\cdot, \cdot) : V \times V \rightarrow \mathbb{R}$ una forma bilineal continua V -elíptico sobre V y sea $l(\cdot) : V \rightarrow \mathbb{R}$ una funcional lineal continua sobre V . Sea P un subespacio cerrado de V tal que

$$a(u + p, v + \bar{p}) = a(u, v) \quad \forall u, v \in V \text{ y } p, \bar{p} \in P. \quad (4.150)$$

También denotando por Q el subespacio de V consistente de las funciones ortogonales a P en la norma L^2 ; tal que

$$Q = \left\{ v \in V \mid \int_{\Omega} up dx = 0 \quad \forall p \in P \right\}, \quad (4.151)$$

y asumiendo que $a(\cdot, \cdot)$ es Q -elíptico: existe una constante $\alpha > 0$ tal que

$$a(q, q) \geq \alpha \|q\|_Q^2 \quad \text{para } q \in Q, \quad (4.152)$$

la norma sobre Q será la misma que sobre V . Entonces

i) Existe una única solución al problema de encontrar $u \in Q$ tal que

$$a(u, v) = \langle l, v \rangle, \quad \forall v \in V \quad (4.153)$$

si y sólo si las condiciones de compatibilidad

$$\langle l, p \rangle = 0 \quad \text{para } p \in P \quad (4.154)$$

se satisfacen.

ii) La solución u satisface

$$\|u\|_Q \leq \alpha^{-1} \|l\|_{Q^*} \quad (4.155)$$

(dependencia continua de los datos).

Otro aspecto importante es la regularidad de la solución, si la solución u al VBVP de orden $2m$ con $f \in H^{s-2m}(\Omega)$ donde $s \geq 2m$, entonces u pertenecerá a $H^s(\Omega)$ y esto queda de manifiesto en el siguiente resultado.

Teorema 41 *Sea $\Omega \subset \mathbb{R}^n$ un dominio suave y sea $u \in V$ la solución al VBVP*

$$a(u, v) = \langle f, v \rangle, \quad v \in V \quad (4.156)$$

donde $V \subset H^m(\Omega)$. Si $f \in H^{s-2m}(\Omega)$ con $s \geq 2m$, entonces $u \in H^s(\Omega)$ y la estimación

$$\|u\|_{H^s} \leq C \|f\|_{H^{s-2m}} \quad (4.157)$$

se satisface.

5 Métodos de Solución Aproximada para EDP

Ya que en general encontrar la solución a problemas con geometría diversa es difícil y en algunos casos imposible usando métodos analíticos. En el presente capítulo se prestará atención a varios aspectos necesarios para encontrar la solución aproximada de problemas variacionales con valor en la frontera (VBVP).

En este capítulo se considera el VBVP de la forma

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

donde

$$\mathcal{L}u = -\nabla \cdot \underline{\underline{a}} \cdot \nabla u + cu\tag{5.2}$$

con $\underline{\underline{a}}$ una matriz positiva definida, simétrica y $c \geq 0$, como un caso particular del operador elíptico definido por la Ec. (4.43) de orden 2, con $\Omega \subset R^2$ un dominio poligonal, es decir, Ω es un conjunto abierto acotado y conexo tal que su frontera $\partial\Omega$ es la unión de un número finito de polígonos.

La sencillez del operador \mathcal{L} nos permite facilitar la comprensión de muchas de las ideas básicas que se expondrán a continuación, pero tengamos en mente que esta es una ecuación que gobierna los modelos de muchos sistemas de la ciencia y la ingeniería, por ello es muy importante su solución.

Si multiplicamos a la ecuación $-\nabla \cdot \underline{\underline{a}} \cdot \nabla u + cu = f_\Omega$ por $v \in V = H_0^1(\Omega)$, obtenemos

$$-v(\nabla \cdot \underline{\underline{a}} \cdot \nabla u + cu) = vf_\Omega\tag{5.3}$$

aplicando el teorema de Green (167) obtenemos la Ec. (4.50), que podemos reescribir como

$$\int_\Omega (\nabla v \cdot \underline{\underline{a}} \cdot \nabla u + cuv) d\underline{\underline{x}} = \int_\Omega vf_\Omega d\underline{\underline{x}}.\tag{5.4}$$

Definiendo el operador bilineal

$$a(u, v) = \int_\Omega (\nabla v \cdot \underline{\underline{a}} \cdot \nabla u + cuv) d\underline{\underline{x}}\tag{5.5}$$

y la funcional lineal

$$l(v) = \langle f, v \rangle = \int_\Omega vf_\Omega d\underline{\underline{x}}\tag{5.6}$$

podemos reescribir el problema dado por la Ec. (5.1) de orden 2 en forma variacional, haciendo uso de la forma bilineal $a(\cdot, \cdot)$ y la funcional lineal $l(\cdot)$.

5.1 Método Galerkin

La idea básica detrás del método Galerkin es, considerando el VBVP, encontrar $u \in V = H_0^1(\Omega)$ que satisfaga

$$a(u, v) = \langle f, v \rangle \quad \forall v \in V \quad (5.7)$$

donde V es un subespacio de un espacio de Hilbert H (por conveniencia nos restringiremos a espacios definidos sobre los números reales).

El problema al tratar de resolver la Ec. (5.7) está en el hecho de que el espacio V es de dimensión infinita, por lo que resulta que en general no es posible encontrar el conjunto solución. En lugar de tener el problema en el espacio V , se supone que se tienen funciones linealmente independientes $\phi_1, \phi_2, \dots, \phi_N$ en V y definimos el espacio V^h a partir del subespacio dimensionalmente finito de V generado por las funciones ϕ_i , es decir,

$$V^h = \text{Generado} \{\phi_i\}_{i=1}^N, \quad V^h \subset V. \quad (5.8)$$

El índice $h = 1/N$ es un parámetro que estará entre 0 y 1, cuya magnitud da alguna indicación de cuan cerca V^h está de V , h se relaciona con la dimensión de V^h . Y como el número N de las funciones base se escoge de manera que sea grande y haga que h sea pequeño, en el límite, cuando $N \rightarrow \infty$, $h \rightarrow 0$.

Después de definir el espacio V^h , es posible trabajar con V^h en lugar de V y encontrar una función u_h que satisfaga

$$a(u_h, v_h) = \langle f, v_h \rangle \quad \forall v_h \in V^h. \quad (5.9)$$

Esta es la esencia del método Galerkin, notemos que u_h y v_h son sólo combinaciones lineales de las funciones base de V^h , tales que

$$u_h = \sum_{i=1}^N c_i \phi_i \quad \text{y} \quad v_h = \sum_{j=1}^N d_j \phi_j \quad (5.10)$$

donde v_h es arbitraria, como los coeficientes de d_j y sin pérdida de generalidad podemos hacer $v_h = \phi_j$. Así, para encontrar la solución u_h sustituimos las Ecs. (5.10) en la Ec. (5.9) y usando el hecho que $a(\cdot, \cdot)$ es una forma bilineal y $l(\cdot)$ es una funcional lineal se obtiene la ecuación

$$\sum_{i=1}^N a(\phi_i, \phi_j) c_i = \langle f, \phi_j \rangle \quad (5.11)$$

o más concisamente, como

$$\sum_{i=1}^N K_{ij}c_i - F_j = 0 \quad j = 1, 2, \dots, N \quad (5.12)$$

en la cual

$$K_{ij} = a(\phi_i, \phi_j) \quad \text{y} \quad F_j = \langle f, \phi_j \rangle \quad (5.13)$$

notemos que tanto K_{ij} y F_j pueden ser evaluados, ya que ϕ_i , $a(\cdot, \cdot)$ y $l(\cdot)$ son conocidas.

Entonces el problema se reduce a resolver el sistema de ecuaciones lineales

$$\sum_{i=1}^N K_{ij}c_i - F_j, \quad j = 1, 2, \dots, N \quad (5.14)$$

o más compactamente

$$\underline{\mathbb{K}}u = \underline{F} \quad (5.15)$$

en la cual $\underline{\mathbb{K}}$ y \underline{F} son la matriz y el vector cuyas entradas son K_{ij} y F_j respectivamente. Una vez que el sistema es resuelto, la solución aproximada u_h es encontrada.

Notemos que la forma bilineal $a(\cdot, \cdot)$ define un producto interior sobre V , si $a(\cdot, \cdot)$ es simétrica y V -elíptica, entonces las propiedades de linealidad y simetría son obvias, mientras que la propiedad de V -elípticidad de $a(\cdot, \cdot)$ es por

$$a(v, v) \geq \alpha \|v\|^2 > 0 \quad \forall v \neq 0, \quad (5.16)$$

además, si $a(\cdot, \cdot)$ es continua, entonces la norma $\|v\|_a \equiv a(v, v)$ generada por este producto interior es equivalente a la norma estándar sobre V , tal que si V es completa con respecto a la norma estándar, esta también es completa con respecto a la norma $\|v\|_a$.

Por otro lado, si el conjunto de funciones base $\{\phi_i\}_{i=1}^N$ se eligen de tal forma que sean ortogonales entre sí, entonces el sistema (5.12) se simplifica considerablemente, ya que

$$K_{ij} = a(\phi_i, \phi_j) = 0 \quad \text{si } i \neq j \quad (5.17)$$

y

$$K_{ii}c_i = F_i \quad \text{ó} \quad c_i = F_i/K_{ii}. \quad (5.18)$$

Así, el problema (5.1) definido en $V^h = H_0^1(\Omega)$ reescrito como el problema (5.7) genera una forma bilineal V^h -elíptica cuyo producto interior sobre V^h es simétrico y positivo definido ya que

$$a(v_h, v_h) \geq \alpha \|v_h\|_{V^h}^2 > 0, \quad \forall v_h \in V^h, v_h \neq 0 \quad (5.19)$$

reescribiéndose el problema (5.9) como el problema aproximado en el cual debemos encontrar $u_h \in V^h \subset V$ tal que

$$a(u_h, v_h) = \langle f, v_h \rangle - a(u_0, v_h) \quad (5.20)$$

donde $u_0 = g = 0$ en $\partial\Omega$, para toda $v_h \in V^h$, es decir

$$\int_{\Omega} (\nabla v_h \cdot \underline{a} \cdot \nabla u_h + cu_h v_h) dx dy = \int_{\Omega} f_{\Omega} v_h dx dy \quad (5.21)$$

para todo $v_h \in V^h$.

Entonces, el problema (5.1) al aplicarle el método Galerkin obtenemos (5.4), el cual podemos reescribirlo como (5.21). Aplicando el teorema de Lax-Milgram (39) a este caso particular, tenemos que este tiene solución única y esta depende continuamente de los datos.

Como un caso particular del teorema de Lax-Milgram (39) tenemos el siguiente resultado

Teorema 42 *Sea V^h un subespacio de dimensión finita de un espacio de Hilbert V , sea $a(\cdot, \cdot) : V^h \times V^h \rightarrow \mathbb{R}$ una forma bilineal continua y V -elíptica, y $l(\cdot) : V^h \rightarrow \mathbb{R}$ una funcional lineal acotada. Entonces existe una única función $u_h \in V^h$ tal que satisface*

$$a(u_h, v_h) = \langle l, v_h \rangle \quad \forall v_h \in V^h. \quad (5.22)$$

Además, si $l(\cdot)$ es de la forma

$$\langle l, v_h \rangle = \int_{\Omega} f_{\Omega} v_h dx \quad (5.23)$$

con $f \in L^2(\Omega)$, entonces

$$\|u_h\|_V \leq \frac{1}{\alpha} \|f\|_{L^2}, \quad (5.24)$$

donde α es la constante en (5.16).

El siguiente resultado nos da una condición suficiente para que la aproximación u_h del método Galerkin converja a la solución u del problema dado por la Ec. (5.7), para más detalle véase [2] y [55].

Teorema 43 *Sea V un subespacio cerrado de un espacio de Hilbert, y sea la forma bilineal $a(\cdot, \cdot) : V^h \times V^h \rightarrow \mathbb{R}$ continua V -elíptica y sea $l(\cdot)$ una funcional lineal acotada. Entonces existe una constante C , independiente de h , tal que*

$$\|u - u_h\|_V \leq C \inf_{v_h \in V^h} \|u - v_h\|_V \quad (5.25)$$

donde u es solución de (5.7) y u_h es solución de (5.20), consecuentemente, una condición suficiente para que la aproximación u_h del método Galerkin converja a la solución u del problema dado por la Ec. (5.7) es que exista una familia $\{V^h\}$ de subespacios con la propiedad de que

$$\inf_{v_h \in V^h} \|u - v_h\|_V \rightarrow 0 \quad \text{cuando} \quad h \rightarrow 0. \quad (5.26)$$

5.1.1 El Método de Residuos Pesados

Este método se basa en el método Galerkin, y se escogen subespacios U^h y V^h de tal manera que la dimensión $\dim U^h = \dim V^h = N$, eligiendo las bases como

$$\{\phi_i\}_{i=1}^N \text{ para } U^h \text{ y } \{\psi_j\}_{j=1}^N \text{ para } V^h \quad (5.27)$$

entonces

$$u_h = \sum_{i=1}^N c_i \phi_i \text{ y } v_h = \sum_{j=1}^N b_j \psi_j \quad (5.28)$$

donde los coeficientes b_j son arbitrarios ya que v_h es arbitraria.

Sustituyendo esta última expresión Ec. (5.28) en

$$(\mathcal{L}u_h - f, v_h) = 0 \quad (5.29)$$

se obtienen N ecuaciones simultáneas

$$\sum_{i=1}^N K_{ij} c_i = F_j \text{ con } j = 1, \dots, N$$

en la cual en la cual $\underline{\underline{K}}$ y \underline{F} son la matriz y el vector cuyas entradas son

$$K_{ij} = (\mathcal{L}\phi_i, \psi_j) \text{ y } F_j = (f, \psi_j)$$

donde (\cdot, \cdot) representa el producto interior asociado a L^2 . A la expresión

$$\tau(u_h) \equiv \mathcal{L}u_h - f \quad (5.30)$$

se le llama el residuo; si u_h es la solución exacta, entonces por supuesto el residuo se nulifica.

5.1.2 Método de Elemento Finito

El método Finite Elements Method (FEM) provee una manera sistemática y simple de generar las funciones base en un dominio con geometría Ω poligonal. Lo que hace al método de elemento finito especialmente atractivo sobre otros métodos, es el hecho de que las funciones base son polinomios definidos por pedazos (elementos Ω_i) que son no cero sólo en una pequeña parte de Ω , proporcionando a la vez una gran ventaja computacional al método ya que las matrices generadas resultan bandadas ahorrando memoria al implantarlas en una computadora.

Así, partiendo del problema aproximado (5.21), se elegirá una familia de espacios V^h ($h \in (0, 1)$) definido por el procedimiento de elementos finitos (descritos en las subsecciones siguientes en el caso de interpoladores lineales, para otros tipos de interpoladores, ver [10]), teniendo la propiedad de que V^h se aproxima a V cuando h se aproxima a cero en un sentido apropiado, esto es, por supuesto una propiedad indispensable para la convergencia del método Galerkin.

Mallado del dominio El Mallado o triangulación \mathcal{T}_h del dominio Ω es el primer aspecto básico, y ciertamente el más característico, el dominio $\Omega \subset \mathbb{R}^2$ es subdividido en E subdominios o elementos Ω_e llamados elementos finitos, tal que

$$\bar{\Omega} = \bigcup_{e=1}^E \bar{\Omega}_e$$

donde:

- Cada $\Omega_e \in \mathcal{T}_h$ es un polígono (rectángulo o triángulo) con interior no vacío ($\mathring{\Omega}_e \neq \emptyset$) y conexo.
- Cada $\Omega_e \in \mathcal{T}_h$ tiene frontera $\partial\Omega_e$ Lipschitz continua.
- Para cada $\Omega_i, \Omega_j \in \mathcal{T}_h$ distintos, $\mathring{\Omega}_i \cap \mathring{\Omega}_j = \emptyset$.

- El diámetro $h_i = \text{Diam}(\Omega_e)$ de cada Ω_e satisface $\text{Diam}(\Omega_e) \leq h$ para cada $e = 1, 2, \dots, E$.
- Cualquier cara de cualquier elemento $\Omega_i \in \mathcal{T}_h$ en la triangulación es también un subconjunto de la frontera $\partial\Omega$ del dominio Ω o una cara de cualquier otro elemento $\Omega_j \in \mathcal{T}_h$ de la triangulación, en este último caso Ω_i y Ω_j son llamados adyacentes.
- Los vértices de cada Ω_e son llamados nodos, teniendo N de ellos por cada elemento Ω_e .

Una vez que la triangulación \mathcal{T}_h del dominio Ω es establecida, se procede a definir el espacio de elementos finitos $\mathbb{P}^h[k]$ a través del proceso descrito a continuación.

Funciones Base A continuación describiremos la manera de construir las funciones base usadas por el método de elemento finito. En este procedimiento debemos tener en cuenta que las funciones base están definidas en un subespacio de $V = H^1(\Omega)$ para problemas de segundo orden que satisfacen las condiciones de frontera.

Las funciones base deberán satisfacer las siguientes propiedades:

- Las funciones base ϕ_i son acotadas y continuas, i.e. $\phi_i \in C(\Omega_e)$.
- Existen ℓ funciones base por cada nodo del polígono Ω_e , y cada función ϕ_i es no cero solo en los elementos contiguos conectados por el nodo i .
- $\phi_i = 1$ en cada i nodo del polígono Ω_e y cero en los otros nodos.
- La restricción ϕ_i a Ω_e es un polinomio, i.e. $\phi_i \in \mathbb{P}_k[\Omega_e]$ para alguna $k \geq 1$ donde $\mathbb{P}_k[\Omega_e]$ es el espacio de polinomios de grado a lo más k sobre Ω_e .

Decimos que $\phi_i \in \mathbb{P}_k[\Omega_e]$ es una base de funciones y por su construcción es evidente que estas pertenecen a $H^1(\Omega)$. Al conjunto formado por todas las funciones base definidas para todo Ω_e de Ω será el espacio $\mathbb{P}^h[k]$ de funciones base, i.e.

$$\mathbb{P}^h[k] = \bigcup_{e=1}^E \mathbb{P}_k[\Omega_e]$$

estas formarán las funciones base globales.

Solución aproximada Para encontrar la solución aproximada elegimos el espacio $\mathbb{P}^h[k]$ de funciones base, como el espacio de funciones lineales ϕ_i definidas por pedazos de grado menor o igual a k (en nuestro caso $k = 1$), entonces el espacio a trabajar es

$$V^h = \text{Generado} \{ \phi_i \in \mathbb{P}^h[k] \mid \phi_i(x) = 0 \text{ en } \partial\Omega \}. \quad (5.31)$$

La solución aproximada de la Ec. (5.21) al problema dado por la Ec. (5.1) queda en términos de

$$\int_{\Omega} (\nabla\phi_i \cdot \underline{a} \cdot \nabla\phi_j - c\phi_i\phi_j) dx dy = \int_{\Omega} f_{\Omega}\phi_j dx dy \quad (5.32)$$

si definimos el operador bilineal

$$K_{ij} \equiv a(\phi_i, \phi_j) = \int_{\Omega} (\nabla\phi_i \cdot a_{ij} \cdot \nabla\phi_j - c\phi_i\phi_j) dx dy \quad (5.33)$$

y la funcional lineal

$$F_j \equiv \langle f, \phi_j \rangle = \int_{\Omega} f_{\Omega}\phi_j dx dy \quad (5.34)$$

entonces la matriz $\underline{\underline{K}} \equiv [K_{ij}]$, los vectores $\underline{u} \equiv (u_1, \dots, u_N)$ y $\underline{F} \equiv (F_1, \dots, F_N)$ definen el sistema lineal (que es positivo definido)

$$\underline{\underline{K}}\underline{u} = \underline{F} \quad (5.35)$$

donde \underline{u} será el vector solución a la Ec. (5.35) cuyos valores serán la solución al problema dado por la Ec. (5.21) que es la solución aproximada a la Ec. (5.1) en los nodos interiores de Ω .

Un Caso más General Sea el operador elíptico (caso simétrico) en el dominio Ω , y el operador definido por

$$\begin{aligned} \mathcal{L}u &= f_{\Omega} \quad \text{en } \Omega \setminus \Sigma \\ u &= g \quad \text{en } \partial\Omega \\ [u]_{\Sigma} &= J_0 \\ [a_n \cdot \nabla u]_{\Sigma} &= J_1 \end{aligned} \quad (5.36)$$

donde

$$\mathcal{L}u = -\nabla \cdot \underline{a} \cdot \nabla u + cu \quad (5.37)$$

conjuntamente con una partición $\Pi = \{\Omega_1, \dots, \Omega_E\}$ de Ω . Multiplicando por la función w obtenemos

$$w\mathcal{L}u = -w\nabla \cdot \underline{a} \cdot \nabla u + cwu = wf_\Omega \quad (5.38)$$

entonces si $w(x)$ es tal que $[w] = 0$ (es decir w es continua) y definimos

$$a(u, w) = \sum_{i=1}^E \int_{\Omega_i} (\nabla u \cdot \underline{a} \cdot \nabla w + cwu) d\underline{x} \quad (5.39)$$

tal que $a(u, w)$ define un producto interior sobre

$$H^1(\Omega) = H^1(\Omega_1) \oplus H^1(\Omega_2) \oplus \dots \oplus H^1(\Omega_E).$$

Entonces, reescribimos la Ec. (5.38) como

$$\begin{aligned} a(u, w) &= \int_{\Omega} wf d\underline{x} + \sum_{i=1}^E \int_{\partial\Omega} wa_n \cdot \nabla u d\underline{s} \\ &= \int_{\Omega} wf_\Omega d\underline{x} + \int_{\partial\Omega} wa_n \cdot \nabla u d\underline{s} - \int_{\Sigma} w [a_n \cdot \nabla u]_\Sigma d\underline{s}. \end{aligned} \quad (5.40)$$

Sea $u_0(x)$ una función que satisface las condiciones de frontera y J_0 una función que satisface las condiciones de salto, tal que

- i) $u_0(x) = g(x)$ en $\partial\Omega$
- ii) $[u_0(x)]_\Sigma = J_0$

y sea $u(x) = u_0(x) + v(x)$. Entonces $u(x)$ satisface la Ec. (5.39) si y sólo si $v(x)$ satisface

$$a(u, w) = \int_{\Omega} wf_\Omega d\underline{x} - \langle u_0, w \rangle - \int_{\Sigma} J_1 w d\underline{s} \quad (5.41)$$

para toda w tal que $w(x) = 0$ en $\partial\Omega$. Sea $\{\phi_i\}$ una base de un subespacio de dimensión finita V^h definido como

$$V^h = \{\phi_i \mid \phi_i \in C^1(\Omega_i), \forall i, \phi_i = 0 \text{ en } \partial\Omega \text{ y } \phi_i \in C^0(\Omega)\}. \quad (5.42)$$

La solución por elementos finitos de (5.41) se obtiene al resolver el sistema lineal

$$\underline{\underline{K}}u = \underline{F} \quad (5.43)$$

donde

$$K_{ij} = a(\phi_i, \phi_j) \quad (5.44)$$

y

$$F_j = \int_{\Omega} \phi_j f_{\Omega} d\underline{x} - a(u_0, \phi_j) - \int_{\Sigma} J_1 \phi_j d\underline{s} \quad (5.45)$$

esta solución será la solución en los nodos interiores de Ω .

5.2 Método de Penalización Interior

En la década de los 70s se desarrolló el método de Interior Penalty (IP) de forma independiente del método Galerkin para ecuaciones elípticas y parabólicas resultando en dos métodos independientes en los que se usan elementos finitos discontinuos, y un número grande de variantes se han introducido y estudiado. Las penalizaciones fueron primeramente introducidas en el método de Elementos finitos como una forma de imponer condiciones de frontera tipo Dirichlet débiles más que incorporarlas a las condiciones de frontera dentro del espacio de elementos finitos [63].

Sea Ω un dominio y sea el operador elíptico

$$\begin{aligned} -\Delta u &= f & \text{en } \Omega \\ u &= 0 & \text{en } \partial\Omega \end{aligned} \quad (5.46)$$

claramente

$$\int_{\Omega} \nabla u \cdot \nabla v d\underline{x} - \int_{\partial\Omega} \frac{\partial u}{\partial n} v d\underline{s} = \int_{\Omega} f v d\underline{x} \quad (5.47)$$

para toda función de prueba v suficientemente suave. Puesto que u se nulifica en la frontera, tenemos también que $B(u, v) = \int f v d\underline{x}$ donde

$$B(u, v) = \int_{\Omega} \nabla u \cdot \nabla v d\underline{x} - \int_{\partial\Omega} \frac{\partial u}{\partial n} v d\underline{s} - \int_{\partial\Omega} \frac{\partial v}{\partial n} u d\underline{s} + \int_{\partial\Omega} \eta u v d\underline{s} \quad (5.48)$$

para cualquier función de peso η . El método entonces determina una solución aproximada u_h en un subespacio de elementos finitos de $H^1(\Omega)$ tal que $B(u_h, v_h) = \int f v_h d\underline{x}$ para todo v_h en el mismo espacio. Notemos que el segundo término de la forma bilineal B surge para asegurar que el método es

consistente. El tercer término fue adicionado para que el problema discreto sea simétrico (y de manera que el método es verdaderamente variacional -la solución discreta minimiza $B(u, u)/2 - \int fu$ sobre el espacio de elementos finitos). Finalmente el último término es el término de penalización, el cual es necesario para garantizar la estabilidad.

Se muestra que si η es tomado como C/h donde h es el tamaño del elemento y C es una constante suficientemente grande, entonces la solución discreta converge a la solución exacta con orden óptimo en H^1 y L^2 .

Un método de penalidad diferente para imponer condiciones de frontera tipo Dirichlet no incluye cualquiera de los términos segundo o tercero en la Ec.(5.48), y usa como peso de penalización $h^{-\sigma}$ para alguna $\sigma \geq 0$. A causa de la omisión del término de consistencia, el método y su análisis incluyen un error de consistencia.

Otra interesante posibilidad es la de incluir todos los términos en la Ec.(5.48) pero cambiando el signo del tercer término en B . La forma bilineal ya no será simétrica, pero tiene una propiedad coercitiva favorable, a saber, $B(u, u) \geq \int |\nabla u|^2$, no importando cual $\eta \geq 0$ se escoja.

5.3 Método Galerkin Discontinuo

En el presente capítulo se dará un esquema para el entendimiento, comparación y análisis de varios métodos de Galerkin Discontinuo que han sido propuestos para el tratamiento de problemas elípticos. Esta clase incluye los llamados métodos de penalización interior [63].

Mallado del dominio El Mallado o triangulación \mathcal{T}_h del dominio Ω , sin pérdida de generalidad, consideremos el dominio $\Omega \subset \mathbb{R}^2$, el cual es subdividido en E subdominios o elementos Ω_e llamados elementos, tal que

$$\bar{\Omega} = \bigcup_{e=1}^E \bar{\Omega}_e \quad (5.49)$$

donde:

- Cada $\Omega_e \in \mathcal{T}_h$ es un polígono (rectángulo o triángulo) con interior no vacío ($\dot{\Omega}_e \neq \emptyset$) y conexo.
- Cada $\Omega_e \in \mathcal{T}_h$ tiene frontera $\partial\Omega_e$ Lipschitz continua.

- Para cada $\Omega_i, \Omega_j \in \mathcal{T}_h$ distintos, $\mathring{\Omega}_i \cap \mathring{\Omega}_j = \emptyset$.
- El diámetro $h_i = \text{Diam}(\Omega_e)$ de cada Ω_e satisface $\text{Diam}(\Omega_e) \leq h$ para cada $e = 1, 2, \dots, E$.
- Cualquier cara de cualquier elemento $\Omega_i \in \mathcal{T}_h$ en la triangulación es también un subconjunto de la frontera $\partial\Omega$ del dominio Ω o una cara de cualquier otro elemento $\Omega_j \in \mathcal{T}_h$ de la triangulación, en este último caso Ω_i y Ω_j son llamados adyacentes.
- Los vértices de cada Ω_e son llamados nodos, teniendo N de ellos por cada elemento Ω_e .

5.3.1 Generalización del Método Galerkin Discontinuo

Sea Ω un dominio y se el operador elíptico

$$\begin{aligned} -\Delta u &= f & \text{en } \Omega \\ u &= 0 & \text{en } \partial\Omega \end{aligned} \quad (5.50)$$

donde el dominio Ω se asume como un dominio poligonal y f es una función dada en $L^2(\Omega)$. Para obtener la formulación débil sobre la cual la discretización se basa, reescribimos el anterior problema como sigue

$$\sigma = \nabla u, \quad -\nabla \cdot \sigma = f \quad \text{en } \Omega \quad (5.51)$$

$$u = 0 \quad \text{en } \partial\Omega. \quad (5.52)$$

Sea K la clausura de un subconjunto abierto de Ω con frontera por pedazos suave. Si multiplicamos la anterior ecuación por funciones de prueba e integramos formalmente sobre K , tenemos

$$\int_K \sigma \cdot \tau dx = - \int_K u \nabla \cdot \tau dx + \int_{\partial K} u n_k \cdot \tau ds \quad (5.53)$$

$$\int_K \sigma \cdot \nabla v dx = \int_K f v dx + \int_{\partial K} \sigma \cdot n_K v ds \quad (5.54)$$

donde n_K es el vector normal unitario exterior a ∂K . Esta es la formulación débil que buscábamos. Con esto en mente podemos ahora definir el método generalizado Galerkin Discontinuo.

Denotamos por \mathcal{T}_h una triangulación del dominio Ω en polígonos K , y por $P(K)$ un espacio de dimensión finita de funciones suaves, típicamente polinomios, definidos sobre el polígono K . Este espacio es usado para aproximar

la variable u . Además, denotamos por $\Sigma(K)$ otro espacio de dimensión finita de funciones suaves que serán usadas para aproximar la variable auxiliar σ .

Sean

$$V_h = \{v \in L^2(\Omega) \mid v|_K \in P(k), \quad \forall K \in \mathcal{T}_h\} \quad (5.55)$$

$$\Sigma_h = \left\{ \tau \in (L^2(\Omega))^2 \mid \tau|_K \in \Sigma(k), \quad \forall K \in \mathcal{T}_h \right\} \quad (5.56)$$

y consideramos la siguiente formulación débil:

Encontrar $u_h \in V_h$ y $\sigma_h \in \Sigma_h$ tal que para toda $K \in \mathcal{T}_h$ tenemos

$$\int_K \sigma_h \cdot \tau dx = - \int_K u_h \nabla \cdot \tau dx + \sum_{e \subset \partial K} \int_e h_u^{e,K} n_k \cdot \tau ds, \quad \forall \tau \in \Sigma(K) \quad (5.57)$$

$$\int_K \sigma_h \cdot \nabla v dx = \int_K f v dx + \sum_{e \subset \partial K} \int_e h_\sigma^{e,K} \cdot n_k v ds, \quad \forall v \in P(K) \quad (5.58)$$

donde las sumas son tomadas sobre los bordes del polígono K , y el flujo numérico $h_\sigma^{e,K}$ y $h_u^{e,K}$ son aproximaciones a $\sigma|_e = \nabla u|_e$ y a $u|_e$ respectivamente sobre las caras de la triangulación.

Por ejemplo, para elementos triangulares, podemos tomar $P(K)$ como el conjunto de polinomios de grado $p \geq 1$ y $\Sigma(K)$ como el conjunto de todos los polinomios del campo vectorial de grado $p - 1$ o p . La elección de la forma de construir los flujos es crucial, algunas propiedades básicas que deben de compartir todas las elecciones de flujo se dan a continuación:

1. Localidad.- Sea $K = K_1$ un elemento de la triangulación, y sea e uno de sus bordes. Asumimos primero que e es un borde interior de nuestra triangulación, tal que existe un segundo elemento K_2 que comparte el borde e con K_1 . Entonces asumimos que $h_\sigma^{e,K}$ y $h_u^{e,K}$ dependen de las restricciones $u_h|_{K_i}$ y $\sigma_h|_{K_i}$ de u_h y σ_h a K_i , $i = 1, 2$. Más precisamente, de modo local tenemos

$$h_\sigma^{e,K} = h_\sigma^{e,K} \left(u_h|_{K_1}, \sigma_h|_{K_1}, u_h|_{K_2}, \sigma_h|_{K_2} \right) \quad (5.59)$$

en los ejemplos, estas funcionales dependen de la forma particular de $h_\sigma^{e,K}$ y $h_u^{e,K}$, ya que dependen sólo de las trazas de $u_h|_{K_i}$, $\nabla u_h|_{K_i}$ y $\sigma_h|_{K_i}$ sobre el borde e . Ya que u_h , ∇u_h y σ_h son en general discontinuas a través de e , la traza de $u_h|_{K_1}$ sobre e será diferente que la traza de $u_h|_{K_2}$ sobre e , y de forma similar ∇u_h y σ_h las cuales tendrán dos

diferentes trazas sobre e . Así que, $h_\sigma^{e,K}$ y $h_u^{e,K}$ dependen linealmente de seis cantidades

$$\left(u_{h|_{K_1}}\right)_e, \left(\nabla u_{h|_{K_1}}\right)_e, \left(\sigma_{h|_{K_1}}\right)_e, \quad (5.60)$$

$$\left(u_{h|_{K_2}}\right)_e, \left(\nabla u_{h|_{K_2}}\right)_e, \left(\sigma_{h|_{K_2}}\right)_e. \quad (5.61)$$

En nuestro particular caso de un problema homogéneo con condiciones de frontera tipo Dirichlet, los flujos sobre los bordes de la frontera tienen la misma dependencia funcional sobre esas seis trazas, siempre que se interpreten las trazas próximas a K_2 como sigue:

$$\left(u_{h|_{K_2}}\right)_e = 0, \quad (5.62)$$

$$\left(\nabla u_{h|_{K_2}}\right)_e = \left(\nabla u_{h|_{K_1}}\right)_e \text{ y } \left(\sigma_{h|_{K_2}}\right)_e = \left(\sigma_{h|_{K_1}}\right)_e. \quad (5.63)$$

Finalmente, es importante notar que en todos los métodos se analizará $h_u^{e,K}$ la cual no depende de $\sigma_{h|_{K_i}}$ (ni sobre $\nabla u_{h|_{K_i}}$, la cual es menos importante).

2. Consistencia.- En todos los métodos se considera como consistente en el sentido que, en una forma funcional descrito como

$$h_\sigma^{e,K} \left(u_{|_{K_1}}, \nabla u_{|_{K_1}}, u_{|_{K_2}}, \nabla u_{|_{K_2}}\right) = \nabla u|_e \quad (5.64)$$

$$h_u^{e,K} \left(u_{|_{K_1}}, \nabla u_{|_{K_1}}, u_{|_{K_2}}, \nabla u_{|_{K_2}}\right) = u|_e \quad (5.65)$$

donde u es una función suave que satisface las condiciones de frontera.

3. Conservación.- Todos los métodos satisfacen

$$h_\sigma^{e,K_1} = h_\sigma^{e,K_2} \quad (5.66)$$

donde e es un borde que comparten los elementos K_1 y K_2 , y de tal forma que podemos escribir de forma simplificada h_σ^e . De tal forma que la propiedad de conservación la podemos escribir como: Si S es la unión de alguna colección de elementos, entonces, tomando a v como idénticamente la unidad en la Ec.(5.58) y sumando sobre K contenida en S tenemos

$$\int_S f dx + \sum_{e \subset \partial S} \int_e h_\sigma^e \cdot n ds = 0. \quad (5.67)$$

Cerramos esta sección con varios comentarios adicionales concernientes a las propiedades antes mencionadas.

- Como se vio, si $h_u^{e,K}$ no depende de σ_h , entonces la variable auxiliar σ_h puede ser eliminada localmente en términos de u_h y ∇u_h , usando la Ec.(5.57). Cuando se usan triángulos, se usa la base ortonormal de Dubiner convirtiendo esta eliminación en trivial.
- En todos los métodos se considera h_σ^e depende de cualquiera de las dos, de las trazas de ∇u_h o de σ_h , pero no de ambas. Aquéllas categorías, para las cuales la matriz de carga tiende a ser matriz dispersa incluye a los métodos de Penalización Interior y Baumann y Oden.
- La mayoría de los métodos satisfacen adicionalmente a la propiedad de conservación dada por la Ec.(5.66), la propiedad análoga $h_u^{e,K_1} = h_u^{e,K_2}$ en cuyo caso escribimos h_u^e . Y nos referiremos a esta como métodos completamente conservativos, estos métodos generan después de eliminar σ_h una matriz de carga simétrica excepto para los métodos de Baumann y Oden y en este caso son conocidos como métodos de penalidad pura. Todos los métodos que se consideran aquí son completamente conservativos.
- Notemos que, en vista de la Ec.(5.58) sólo la componente normal $h_\sigma^{e,K} \cdot n_k$ de $h_\sigma^{e,K}$ participa en el método, la componente tangencial es irrelevante. En la práctica, la componente normal depende sólo de las trazas normales.

5.3.2 Flujos Numéricos Independientes de ∇u_h

Sea e un borde que comparten los elementos K_1 y K_2 . Definiendo también los vectores normales n_1 y n_2 sobre e apuntando hacia el exterior de K_1 y K_2 respectivamente. Si v es una función sobre $K_1 \cup K_2$ pero con posibles discontinuidades a través de e , sea v_i que denota $(v|_{K_i})|_e$, $i = 1, 2$.

Para una función escalar v definimos

$$\dot{v} = \frac{1}{2}(v_1 + v_2), \quad [[v]] = v_1 n_1 + v_2 n_2 \quad (5.68)$$

si τ es una función vector valuada, tenemos

$$\dot{\tau} = \frac{1}{2}(\tau_1 + \tau_2), \quad [[\tau]] = \tau_1 \cdot n_1 + \tau_2 \cdot n_2 \quad (5.69)$$

notemos que el salto $[[v]]$ de una función escalar v es un vector paralelo a n y que $[[\tau]]$ es el salto de la componente normal de la función vectorial τ , siendo esta una cantidad escalar. La ventaja de esta definición es que no depende del asignamiento de un orden a los elementos K_i .

Aquí consideraremos que el método es determinado por la siguiente elección del flujo numérico

$$\begin{aligned} h_\sigma^{e,K} &= \dot{\sigma}_h - \alpha^e [[u_h]] + \beta^e [[\sigma_h]] \\ h_u^{e,K} &= \dot{u}_h + \gamma^e \cdot [[u_h]] \end{aligned} \quad (5.70)$$

donde β^e y γ^e son funciones vector valuadas sobre e . A menudo ellas son constantes y en muchos métodos ellas se toman como nulas. El término $\alpha^e ([[u_h]])$ puede ser tomada simplemente como

$$\alpha^e ([[u_h]]) = \eta^e [[u_h]] \quad (5.71)$$

para alguna constante o función η^e . Otra posibilidad es definir el operador $r_e : L^1(e) \rightarrow \Sigma_h$ definida como

$$\int_\Omega r_e(q) \cdot \tau dx = - \int_e q \cdot \dot{\tau} ds \quad (5.72)$$

para todo $\tau \in \Sigma_h$ y $q \in L^1(e)$, y el conjunto

$$\alpha^e ([[u_h]]) = \eta^e \widehat{r^e} [[u_h]]. \quad (5.73)$$

Primero reescribimos el método insertando el flujo de la Ec.(5.70) dentro de la ecuación de Galerkin dadas por las Ecs.(5.57) y (5.58) y tomando sobre $K \in \mathcal{T}_h$. Denotando por \mathcal{E}_h el conjunto de todas los bordes, obteniendo

$$\begin{aligned} \int_\Omega \sigma_h \cdot \tau dx &= \sum_K \int_K \nabla u_h \cdot \tau dx + \\ &\sum_{e \in \mathcal{E}_h} \int_e (\gamma^e \cdot [[u_h]] [[\tau]] - [[u_h]] \cdot \dot{\tau}) ds \end{aligned} \quad (5.74)$$

$$\begin{aligned} \sum_K \int_K \sigma_h \cdot \nabla v dx &= \int_\Omega f v dx + \\ &\sum_{e \in \mathcal{E}_h} \int_e (\dot{\sigma}_h - \alpha^e ([[u_h]]) + \beta^e [[\sigma_h]]) \cdot [[v]] ds \end{aligned} \quad (5.75)$$

para toda $\tau \in \Sigma_h, v \in V_h$. Si tomamos que todos los α^e, β^e y γ^e se nulifican, recuperamos el método de Galerkin Discontinuo original. Este método puede ser inestable al menos para mallas uniformes, sin embargo la estabilidad se logra si α^e es un operador positivo. Definiendo α^e por la Ec.(5.73) con $\eta^e > 0$ (pueden ser β^e y γ^e zero) obteniendo las variantes de Bassi y Rebay, definiendo α^e por la Ec.(5.71), $\eta^e > 0$ obtenemos el método LDG.

Continuando, podemos eliminar σ_h para reescribir el método en términos de u_h solamente (esto es lo usualmente preferido en la implementación). Para hacer esto, definimos dos operadores R y L . El operador $R : V_h \rightarrow \Sigma_h$ dado por $R(v) = \sum_{e \in \mathcal{E}_h} r_e([[u_h]])$, o equivalentemente,

$$\int_{\Omega} R(\varphi) \cdot \tau dx = - \sum_{e \in \mathcal{E}_h} \int_e [[\varphi]] \cdot \dot{\tau} ds \quad (5.76)$$

para toda $\tau \in \Sigma_h$ y el operador $L : L^1(\cup \mathcal{E}_h) \rightarrow \Sigma_h$ que es definido por

$$\int_{\Omega} L(\varphi) \cdot \tau dx = \sum_{e \in \mathcal{E}_h} \int_e \varphi \cdot [[\tau]] ds \quad (5.77)$$

para toda $\tau \in \Sigma_h$.

Denotamos por P_{Σ} la L^2 -proyección sobre Σ_h , entonces podemos reescribir la Ec.(5.74) como

$$\sigma_h = P_{\Sigma}(\nabla u_h) + R(u_h) + L(\gamma \cdot [[u_h]]) \quad (5.78)$$

y la Ec.(5.75) como

$$\begin{aligned} \sum_K \int_K \sigma_h \cdot \nabla v dx &= \int_{\Omega} f v dx + \int_{\Omega} \sigma_h \cdot (-R(v) + L(\beta \cdot [[v]])) \quad (5.79) \\ &\quad - \sum_{e \in \mathcal{E}_h} \int_e \alpha^e([[u_h]]) \cdot [[v]] ds \end{aligned}$$

donde β y γ son funciones sobre $\cup \mathcal{E}_h$ las cuales están dadas por β^e y γ^e respectivamente, sobre cada borde e . Finalmente insertando la Ec.(5.78) en la Ec.(5.79), obtenemos

$$\sum_K \int_K (P_{\Sigma}(\nabla u_h) + R(u_h) + L(\gamma \cdot [[u_h]])) \cdot (\nabla v + R(v) + L(\beta \cdot [[v]]) dx$$

$$+ \sum_{e \in \mathcal{E}_h} \int_e \alpha^e ([[u_h]]) \cdot [[v]] ds = \int_{\Omega} f v dx. \quad (5.80)$$

Notemos que la segunda suma del lado izquierdo de la Ec.(5.80) es simétrica con respecto a u_h y v ya que

$$\begin{aligned} & \sum_{e \in \mathcal{E}_h} \int_e \alpha^e ([[u_h]]) \cdot [[v]] ds = \\ & = \begin{cases} \sum_{e \in \mathcal{E}_h} \int_e \eta^e [[u_h]] \cdot [[v]] ds, & \text{si } \alpha^e \text{ es definido por Ec.(5.71)} \\ \sum_{e \in \mathcal{E}_h} \int_e \eta^e r_e ([[u_h]]) \cdot r_e [[v]] ds, & \text{si } \alpha^e \text{ es definido por Ec.(5.73)}. \end{cases} \end{aligned} \quad (5.81)$$

de este modo es claro que la matriz de carga simétrica es obtenida si elegimos $\beta^e = -\gamma^e$ para toda e . Esta elección es usada por el método LDG.

En la práctica la inclusión de $\nabla P(K) \subset \Sigma(K)$ generalmente es suficiente. En el caso de la proyección P_{Σ} no es requerida en la Ec.(5.80).

Finalmente, notemos que el soporte de v es contenido en un solo elemento K , entonces el soporte de $R(v)$ el cual generalmente contiene a todos los elementos que contienen el borde de K . Consecuentemente el producto $R(u_h) \cdot R(v)$ en la Ec.(5.80) la cual generalmente tiene un gran impacto negativo en la dispersión de la matriz de carga. Este problema es mucho menos severo cuando el flujo numérico es independiente de σ_h .

5.3.3 Flujos Numéricos Independientes de σ_h

Primeramente consideremos, en lugar de la Ec.(5.70) el siguiente flujo numérico

$$\begin{aligned} h_{\sigma}^{e,K} &= \widehat{\nabla} u_h - \alpha^e [[u_h]] + \beta^e [[\nabla u_h]] \\ h_u^{e,K} &= \dot{u}_h + \gamma^e [[u_h]] \end{aligned} \quad (5.82)$$

donde β^e y γ^e son funciones vector valuadas sobre e . Procediendo a la eliminación de la variable σ_h como en la sección anterior. Pero usando las definiciones de R y L en las Ecs.(5.76) y (5.77) respectivamente, obtenemos

$$\begin{aligned} & \sum_K \int_K (P_{\Sigma}(\nabla u_h) + R(u_h) + L(\gamma \cdot [[u_h]])) \cdot \nabla v \\ & + \nabla u_h \cdot (R(v) - L(\beta \cdot [[v]])) dx \\ & + \sum_{e \in \mathcal{E}_h} \int_e \alpha^e ([[u_h]]) \cdot [[v]] ds = \int_{\Omega} f v dx. \end{aligned} \quad (5.83)$$

escogiendo $\beta = \gamma = 0$ y α escogido como la Ec.(5.71), recuperamos el método de Penalización Interior, mientras que para $\beta = \gamma = 0$ y α escogido como la Ec.(5.73) procedemos a recuperar el método original de Galerkin Discontinuo de Bassi y Rebay sobre algunas suposiciones generales, y para elementos triangulares, el esquema de estabilidad y convergencia óptima se da siempre y cuando $\eta^e > 3$, donde este número representa, en esencia el número de bordes por elemento.

Notemos que el número de entradas no cero de la matriz de carga es reducida a un mínimo, esto es debido a que el término $R(u_h) \cdot R(v)$ que aparece en la Ec.(5.80) ya no está presente en la Ec.(5.83).

Considerando ahora otra familia de flujos numéricos, consideremos

$$\begin{aligned} h_\sigma^e &= \zeta \widehat{\nabla} u_h - \alpha^e ([[u_h]]) \\ h_u^{e,K} &= \dot{u}_h + \delta [[u_h]] \cdot n_K \end{aligned} \quad (5.84)$$

donde ζ y δ son parámetros reales. Diferentes opciones de estos parámetros son seleccionadas en los diferentes métodos de Galerkin Discontinuo. Notemos que para $\delta \neq 0$ corresponde a métodos en los cuales no es totalmente conservativo y para $\zeta \neq 1$ la consistencias es violada.

Usando la Ec.(5.84) en las Ecs(5.57) y (5.58) y procedemos a eliminar σ_h como antes, obtenemos

$$\begin{aligned} \sum_K \int_K (P_\Sigma(\nabla u_h) \cdot \nabla v + (1 - 2\delta) R(u_h) \cdot \nabla v + \zeta \nabla u_h \cdot R(v)) dx \\ + \sum_{e \in \mathcal{E}_h} \int_e \alpha^e ([[u_h]]) \cdot [[v]] ds = \int_\Omega f v dx \end{aligned} \quad (5.85)$$

para $\delta = 1, \zeta = 1, \alpha^e = 0$ y $\nabla P(K) \subset \Sigma(K)$ (tal que se $\nabla P(K)$, P_Σ se reduce al operador inclusión y puede ser suprimido), esto es exactamente el método Galerkin Discontinuo de Baumann y Oden. Para ver esto, la anterior ecuación puede ser reescrita. Para iniciar notemos que

$$\int_\Omega \nabla u \cdot R(v) dx = - \sum_{e \in \mathcal{E}_h} \int_e [[v]] \widehat{\nabla} u ds = - \sum_K \int_{\partial K} ([v]) \frac{\partial u}{\partial n_K} ds \quad (5.86)$$

donde seleccionamos en cada elemento K , para cada $e \in \partial K$

$$([v]) = \frac{1}{2} (v^{int} - v^{ext})_e \quad (5.87)$$

con obvio entendimiento de los símbolos. Con esta notación y cuando $\nabla P(K) \subset \Sigma(K)$, la Ec.(5.85) puede ser reescrita como

$$\begin{aligned} \sum_K \int_K \nabla u_h \cdot \nabla v dx + \int_{\partial K} \left((2\delta - 1) ([u_h]) \frac{\partial v}{\partial n} - \zeta ([u_h]) \frac{\partial u_h}{\partial n} \right) ds \quad (5.88) \\ + \sum_{e \in \mathcal{E}_h} \int_e \alpha^e ([[u_h]]) \cdot [[v]] ds = \int_{\Omega} f v dx \end{aligned}$$

el cuál es el método Galerkin Discontinuo de Baumann y Oden cuando $\delta = \zeta = 1$ y $\alpha^e = 0$. Este método requiere algunas suposiciones adicionales, como es el hecho de que los polinomios deberán de ser de grado mayo o igual a dos. La situación cobra importancia cuando α^e es tomada como en la Ec.(5.71) o Ec.(5.73) con $\eta^e > 0$.

Por otro lado, tomando $\delta = 1/2$ y $\zeta = 0$ en la Ec.(5.84), la Ec.(5.85) quedaría como

$$\sum_K \int_K P_{\Sigma}(\nabla u_h) \cdot \nabla v dx + \sum_{e \in \mathcal{E}_h} \int_e \alpha^e ([[u_h]]) \cdot [[v]] ds = \int_{\Omega} f v dx. \quad (5.89)$$

esto, cuando $\nabla P(K) \subset \Sigma(K)$, puede ser visto como una extensión del método de Babuška-Zlámal de Penalización Interior.

5.3.4 Distintos tipos de Métodos Galerkin Discontinuo

En esta unificación de diversos métodos de Galerkin Discontinuo, en esta sección se resumen las diferentes opciones de flujo que se necesitan para obtener las diversas variantes del método. Para todas las variantes del método $P(K)$ es el espacio de polinomios estándar y $\Sigma(K)$ es tomado tal que contiene $\nabla P(K)$.

Podemos ver que esta división en clases subdivide de forma natural aquellos métodos completamente conservativos y los parcialmente conservativos, por otro lado, divide aquellos cuyo flujo es independiente de σ_h y aquellos que no lo son. Podemos decir que los métodos completamente conservativos generan problemas simétricos cuando los parámetros de su flujo numérico están adecuadamente definidos, y los métodos parcialmente conservativos generan métodos no simétricos.

También notemos que cuyos métodos en los cuales el flujo numérico es independiente de σ_h produce matrices de carga con un marcado número de entradas distintas de cero.

Método	$h_\sigma^{e,K}$	$h_u^{e,K}$
Bassi-Rebay 1	$\dot{\sigma}_h$	\dot{u}_h
Brezze et al. 1	$\dot{\sigma}_h - \eta^e r_e \widehat{[[u_h]]}$	\dot{u}_h
LDG	$\dot{\sigma}_h - \eta^e [[u_h]] + \beta^e [[\sigma_h]]$	$\dot{u}_h + \gamma^e [[u_h]]$
IP	$\dot{u}_h - \eta^e [[u_h]]$	\dot{u}_h
Bassi-Rebay 2	$\dot{u}_h - \eta^e r_e \widehat{[[u_h]]}$	\dot{u}_h
Baumman-Oden	\dot{u}_h	$\dot{u}_h - [[u_h]] \cdot n_K$
Babuška-Zlámal	$-\eta^e [[u_h]]$	$u_h _K$
Brezze et al. 2	$-\eta^e r_e \widehat{[[u_h]]}$	$u_h _K$

5.4 Método Discontinuo Enriquecido

El método estándar de elementos finitos que se basa en polinomios continuos definidos por pedazos mediante la aproximación Galerkin, esta es óptima para el operador de Laplace, en el sentido de que este minimiza el error en la norma de energía o en la semi-norma H^1 , esta propiedad asegura buen desempeño en el cálculo sobre mallado no muy fino. Sin embargo, un buen desempeño sobre cualquier mallado no está garantizado para el método de elementos finitos, principalmente en presencia de gradientes grandes y oscilaciones rápidas.

Numerosos métodos se han desarrollado para salvar estas deficiencias, la gran mayoría de ellos se basan en modificaciones del método Galerkin, motivada por el método FETI para descomposición de dominio no conforme con el uso de multiplicadores de Lagrange, el método de Discontinuo Enriquecido [64] propone una discretización con elementos finitos estándar con un campo polinomial en el cual cada elemento es enriquecido por un espacio libre de soluciones que gobiernan el problema homogéneo con coeficientes constantes. Este enriquecimiento es fácil de obtener y es virtualmente independiente de la geometría y el orden del polinomio usado en la discretización. De este modo, las características de las ecuaciones diferenciales son incluidas en la aproximación.

El concepto de métodos de elementos finitos con multiplicadores de Lagrange para hacer cumplir las restricciones de frontera son bien conocidos y estos han sido exitosamente aplicados a el análisis estructural de sistemas modelados por diferentes tipos de elementos.

Sea $\Omega \subset \mathbb{R}^n$ un dominio con frontera suave $\partial\Omega$, por simplicidad conside-

raremos el siguiente problema con condiciones de frontera Dirichlet: Encontrar $u : \bar{\Omega} \rightarrow \mathbb{R}$ tal que

$$\mathcal{L}u = f \text{ en } \Omega \quad (5.90)$$

$$u = g \text{ sobre } \partial\Omega \quad (5.91)$$

donde $f : \Omega \rightarrow \mathbb{R}$ y $g : \partial\Omega \rightarrow \mathbb{R}$ son funciones dadas, el operador \mathcal{L} es considerado como de segundo orden.

Particionando el dominio Ω en E subdominios $\{\Omega_1, \dots, \Omega_E\}$ sin traslape con fronteras $\partial\Omega_i$, con $i = 1, \dots, E$ tal que

$$\bar{\Omega} = \bigcup_{i=1}^E \bar{\Omega}_i \quad (5.92)$$

donde

$$\bigcap_{i=1}^E \Omega_i = \emptyset \quad (5.93)$$

y denotamos a la unión de los elementos interiores por

$$\tilde{\Omega} = \bigcup_{i=1}^E \Omega_i \quad (5.94)$$

similarmente, la unión de los elementos de la frontera es denotado por

$$\tilde{\partial\Omega} = \bigcup_{i=1}^E \partial\Omega_i \quad (5.95)$$

y a los elementos en la interfase o los elementos de la frontera interior es

$$\Gamma = \tilde{\partial\Omega} \setminus \partial\Omega. \quad (5.96)$$

Un ejemplo de un dominio Ω y su descomposición en subdominios Ω_i y cada Ω_i a su vez descompuesto en Ω_e subdominios se muestra en la figura:

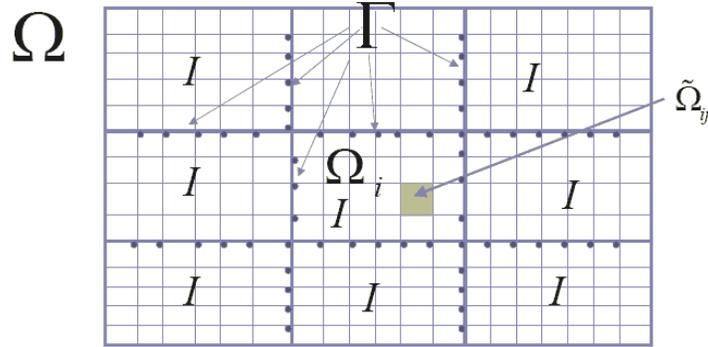


Figura 2: Dominio Ω descompuesto en subdominios Ω_i , con $i = 1, 2, \dots, 9$.

Sea $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j$ donde $\partial\Omega_i$ y $\partial\Omega_j$ son las fronteras de dos subregiones adyacentes, entonces definimos como la traza a la restricción de v^i a Γ_{ij} . Pero como Γ_{ij} , para dos subregiones vecinas hay dos trazas definidas una que corresponde a v^i y otra a v^j , entonces se requiere introducir la siguiente notación para poderlas distinguir entre si:

$$v_+ \equiv Tr(v^i) \quad (5.97)$$

cuando Ω_i cae del lado positivo de Γ_{ij} y

$$v_- \equiv Tr(v^j) \quad (5.98)$$

en caso contrario. Aquí $Tr(v)$ designa al operador traza de la función v . En general $v_+ \neq v_-$ ya que se trabaja con espacios de funciones definidas por tramos.

Observación 2 *Notemos que al considerar una función w en Ω , su definición en Γ es innecesaria, ya que la medida de Lebesgue de Γ es cero. Si la traza de w_α es definida en casi todos lados salvo un conjunto de medida cero sobre $\partial\Omega_\alpha$ para $\alpha = 1, \dots, E$, entonces tal traza es también definida en Γ . En particular, si la traza de w_α esta definida sobre $\partial\Omega_\alpha$ para cada $\alpha = 1, \dots, E$, entonces ellas definen dos funciones definidas en casi todos lados salvo un conjunto de medida cero sobre Γ , denotadas por (w_+, w_-) correspondientes a los lados de trazas positivas y negativas de Γ respectivamente.*

Definición 44 *El salto de v sobre Γ de funciones definidas por pedazos como*

$$[[w]] \equiv w_+ - w_- \quad (5.99)$$

y el promedio como

$$\dot{w} \equiv \frac{1}{2}(w_+ + w_-) \quad (5.100)$$

respectivamente.

5.4.1 Formulación Variacional Híbrida con Continuidad Débil

La fórmula variacional del problema con condiciones de frontera dado por la Ecs.(5.90) y (5.91) está dado en términos de el conjunto de soluciones de prueba

$$\mathcal{V} = L^2(\Omega) \cap H^1(\widetilde{\partial\Omega}) \quad (5.101)$$

estas funciones posiblemente sean discontinuas a través de los elementos de frontera, similarmente, las funciones no necesariamente satisfacen las condiciones de frontera Dirichlet.

La continuidad entre elementos y las condiciones de frontera Dirichlet son ambas impuestas débilmente por los multiplicadores de Lagrange en $H^{-1/2}(\widetilde{\partial\Omega})$, sea

$$H(\text{div}; \Omega) = \{\mathbf{p} \mid \mathbf{p} \in (L^2(\Omega))^n, \text{div}\mathbf{p} \in L^2(\Omega)\} \quad (5.102)$$

y tomando $\mathbf{p} \in W = H(\text{div}; \Omega)$ entonces las trazas normales de \mathbf{p} sobre $\partial\Omega_i$ son tomados como los multiplicadores de Lagrange. Estas trazas normales $\mathbf{p} \cdot \mathbf{n}$ están bien definidas por pertenecer a $H^{-1/2}(\widetilde{\partial\Omega})$ y satisfacen

$$\langle \mathbf{p} \cdot \mathbf{n}, v \rangle_{\widetilde{\partial\Omega}} = (\nabla v, \mathbf{p})_{\Omega_i} + (v, \text{div}\mathbf{p})_{\Omega_i} \quad (5.103)$$

aquí $\langle \cdot, \cdot \rangle$ es la dualidad entre los pares $H^{-1/2}(\partial\Omega)$ y $H^{1/2}(\partial\Omega)$ y los sub-índices denotan dominios de integración distintos de $\partial\Omega$, y (\cdot, \cdot) es el producto interior en $L^2(\Omega)$ y los sub-índices denotan los dominios de integración distintos de Ω . El vector normal unitario que apunta hacia afuera de la frontera es denotado por \mathbf{n} . No son requeridos grados de libertad adicionales en la aproximación por multiplicadores del Lagrange como la traza normal de \mathbf{p} , comparada con las funciones escalares definidas sobre los elementos de la frontera.

Ahora, buscaremos el punto estacionario $u \in \mathcal{V}$ y $\mathbf{p} \in \mathcal{W}$ de los multiplicadores de Lagrange

$$\Pi(u, \mathbf{p}) = \frac{1}{2}a(u, u) - \langle \mathbf{p} \cdot \mathbf{n}, v \rangle_{\tilde{\partial\Omega}} - L(u) - L_b(\mathbf{p}) \quad (5.104)$$

admitiendo para las discontinuidades, el operador bilineal $a(\cdot, \cdot)$ definido sobre los elementos interiores $\tilde{\Omega}$, satisfaciendo

$$a(v, u) = (v, \mathcal{L}u)_{\tilde{\Omega}} + \langle \mathcal{L}_b u, v \rangle_{\tilde{\partial\Omega}} \quad (5.105)$$

aquí, \mathcal{L}_b es el operador de frontera correspondiente a \mathcal{L} . Los términos representando los datos son

$$\begin{aligned} L(v) &= (v, f) \\ L_b(\mathbf{q}) &= -\langle \mathbf{q} \cdot \mathbf{n}, g \rangle \end{aligned} \quad (5.106)$$

para funciones f y g suficientemente suaves.

Múltiples métodos de estabilización incluyendo saltos supone el operador de frontera a través de los elementos de la interfase. Tales términos son derivados directamente de las ecuaciones gobernantes del método variacional multiescala. La presente formulación impone continuidad del campo en sí mismo.

5.4.2 Formulación Débil

El punto estacionario de la funcional dada por la Ec.(5.104) es obtenida por el establecimiento en la primera variación a cero. En forma particionada, esto conduce a

$$a(v, u) - \langle \mathbf{p} \cdot \mathbf{n}, v \rangle_{\tilde{\partial\Omega}} = L(v) \quad (5.107)$$

$$-\langle \mathbf{q} \cdot \mathbf{n}, v \rangle_{\tilde{\partial\Omega}} = L_b(\mathbf{q}) \quad (5.108)$$

aquí, $v \in \mathcal{V}$ y $\mathbf{q} \in \mathcal{W}$ son variaciones arbitrarias de u y \mathbf{p} , respectivamente.

La clave de las condiciones de estabilidad para la formulación mixta y híbrida son descritos por el Teorema de Brezzi. Ellas se necesitan verificar para el problema dimensional finito. Estas condiciones restringe la selección de la interpolación de elementos finitos que puede usarse para un aplicación particular. La discretización de las Ecs.(5.107) y (5.108) conducen a una diagonal por bloque típicamente de ceros.

La correspondiente ecuación Euler-Lagrange típica es

$$\mathcal{L}u = f, \text{ en } \tilde{\Omega} \quad (5.109)$$

$$[[u]] = 0, \text{ sobre } \Gamma \quad (5.110)$$

$$u = g, \text{ sobre } \partial\Omega \quad (5.111)$$

$$\mathbf{p} \cdot \mathbf{n} = \mathcal{L}_b u, \text{ sobre } \tilde{\partial\Omega} \quad (5.112)$$

esta última ecuación facilita una interpretación de los multiplicadores de Lagrange. Por ejemplo, si \mathcal{L}_b es la derivada normal, entonces $\mathbf{p} = \nabla u$ en $\tilde{\Omega}$.

5.4.3 Aproximación Galerkin

Buscamos aproximar la solución $u^h \in \mathcal{V}^h \subset \mathcal{V}$ de la forma

$$u^h = u^P + u^Q \quad (5.113)$$

aquí, $u^P \in \mathcal{V}^P \subset H^1(\Omega)$ son las funciones polinomiales continuas estándar definidas por pedazos de elementos finitos en la escala gruesa y $u^Q \in \mathcal{V}^Q$ es el campo enriquecido. Distinto es en la escala fina, las cuales tienen un rol similar, u^Q puede ser discontinua a través de los elementos de frontera. Esto nos permite burlar los inconvenientes al intentar aproximar la escala fina global de las funciones de Green del método variacional multiescala, y la pérdida de los efectos globales esperados por la restricción de los residuales libres que se nulifican en la traza de los elementos de frontera. Un adicional beneficio potencial es que mejora la aproximación de soluciones discontinuas.

Esto proporciona gran flexibilidad en la selección de \mathcal{V}^Q en este esquema. Asumimos que la aproximación de soluciones particulares para \mathcal{V}^P es satisfactoria. El enriquecimiento debe por lo consiguiente contener soluciones de la ecuación parcial homogénea que no son representadas por el subyacente campo polinomial, en este método, el campo enriquecido puede enteramente capturar la solución homogénea, más que simplemente mejorar el campo polinomial.

Débil imposición de continuidad permite el uso de espacios libres de soluciones como base para el enriquecimiento. Consecuentemente la potencial dificultad de problemas con valor de frontera a nivel de elemento no pueda ser resuelto, ni analítica ni numéricamente. El relativamente simple espacio libre de soluciones es aplicable a prácticamente cualquier geometría y orden polinomial que se le aplique al elemento.

Resumiendo, se emplea la suma directa de las relaciones $\mathcal{V}^h = \mathcal{V}^P \oplus \mathcal{V}^Q$, donde $u^Q \in \mathcal{V}^Q$ y \mathcal{V}^Q es generada por las soluciones de

$$\mathcal{L}u^Q = 0 \text{ en } \mathbb{R}^n \quad (5.114)$$

que no están contempladas en las bases polinomiales. Entonces estas funciones son empleadas a un nivel de elementos, típicamente se emplean soluciones del caso con coeficientes constantes, el cual es fácil de obtener.

El tratamiento de funciones de peso es consistente con la Ec.(5.113), a saber $v^h = v^P + v^Q$ y $\mathbf{q} \in \mathcal{W}^h$. Por el método Galerkin, se busca $u^h \in \mathcal{V}^h$ y $\mathbf{p} \in \mathcal{W}^h$ tal que para todo $\{v^h, \mathbf{q}^h\} \in \mathcal{V}^h \times \mathcal{W}^h$ satisfagan

$$a(v^h, u^h) - \langle \mathbf{p}^h \cdot \mathbf{n}, v^h \rangle_{\widetilde{\partial\Omega}} = L(v^h) \quad (5.115)$$

$$- \langle \mathbf{q}^h \cdot \mathbf{n}, v^h \rangle_{\widetilde{\partial\Omega}} = L_b(\mathbf{q}^h) \quad (5.116)$$

estas ecuaciones pueden ser descompuestas como sigue

$$a(v^P, u^P) + a(v^P, u^Q) - \langle \mathbf{p}^h \cdot \mathbf{n}, v^P \rangle_{\widetilde{\partial\Omega}} = L(v^P) \quad (5.117)$$

$$a(v^Q, u^P) + a(v^Q, u^Q) - \langle \mathbf{p}^h \cdot \mathbf{n}, v^Q \rangle_{\widetilde{\partial\Omega}} = L(v^Q) \quad (5.118)$$

$$- \langle \mathbf{q}^h \cdot \mathbf{n}, u^P \rangle_{\widetilde{\partial\Omega}} - \langle \mathbf{q}^h \cdot \mathbf{n}, v^Q \rangle_{\widetilde{\partial\Omega}} = L_b(\mathbf{q}^h) \quad (5.119)$$

Debido a la naturaleza discontinua de \mathcal{V}^Q , la Ec.(5.118) puede ser usada para eliminar u^Q por condensación estática en el nivel de elementos. Este procedimiento proporciona una aproximación local (y por tanto económica) de efecto global a escala fina sobre las escalas gruesas. La escala fina es derivada por los elementos interiores residuales $L(v^Q) - a(v^Q, u^P)$, y el inter elemento y las discontinuidades de frontera $\langle \mathbf{p}^h \cdot \mathbf{n}, v^Q \rangle_{\widetilde{\partial\Omega}}$.

5.4.4 Condensación Estática

Más que meramente un recurso conceptual, la eliminación local de u^Q , conduce a la formulación $u^P - \mathbf{p}$, es propuesto como un procedimiento práctico que simplifica y condiciona la formulación, a fin de reducir el costo computacional. De esta forma, el costo de resolver la matriz del problema que resulta del método es virtualmente independiente de la dimensión de u^Q .

El campo de enriquecimiento generalmente contiene varios grados de libertad en cada elemento. Consecuentemente, la condensación estática es presentada en esta sección en términos de ecuaciones discretas. Para la

advección-difusión el enriquecimiento puede contener un solo grado de libertad.

Consideremos una partición del sistema global de las ecuaciones discretas

$$\begin{bmatrix} \mathbf{K}^{PP} & \mathbf{K}^{PQ} & \mathbf{K}^{PC} \\ \mathbf{K}^{QP} & \mathbf{K}^{QQ} & \mathbf{K}^{QC} \\ \mathbf{K}^{CP} & \mathbf{K}^{CQ} & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{u}^P \\ \mathbf{u}^Q \\ \mathbf{p} \end{Bmatrix} = \begin{Bmatrix} \mathbf{F}^P \\ \mathbf{F}^Q \\ \mathbf{F}^C \end{Bmatrix} \quad (5.120)$$

aquí, \mathbf{u}^P , \mathbf{u}^Q y \mathbf{p} son vectores conteniendo los grados de libertad de u^P , u^Q y p^h respectivamente. Las matrices de la Ec.(5.120) provienen de los términos de las ecuaciones de la formulación Galerkin de acuerdo a las siguientes relaciones

$$\begin{aligned} a(v^P, u^P) &\rightarrow \mathbf{K}^{PP} \\ a(v^P, u^Q) &\rightarrow \mathbf{K}^{PQ} \\ -\langle \mathbf{p}^h \cdot \mathbf{n}, v^P \rangle &\rightarrow \mathbf{K}^{PC} \\ a(v^Q, u^P) &\rightarrow \mathbf{K}^{QP} \\ a(v^Q, u^Q) &\rightarrow \mathbf{K}^{QQ} \\ -\langle \mathbf{p}^h \cdot \mathbf{n}, v^Q \rangle_{\partial\tilde{\Omega}} &\rightarrow \mathbf{K}^{QC} \\ -\langle \mathbf{q}^h \cdot \mathbf{n}, v^P \rangle_{\partial\tilde{\Omega}} &\rightarrow \mathbf{K}^{CP} \\ -\langle \mathbf{q}^h \cdot \mathbf{n}, v^Q \rangle_{\partial\tilde{\Omega}} &\rightarrow \mathbf{K}^{CQ} \\ L(v^P) &\rightarrow \mathbf{F}^P \\ L(v^Q) &\rightarrow \mathbf{F}^Q \\ L_b(\mathbf{q}^h) &\rightarrow \mathbf{F}^C \end{aligned} \quad (5.121)$$

debido a la continuidad de u^P , los arreglos \mathbf{K}^{PC} y \mathbf{K}^{CP} son vacíos excepto a lo largo de la frontera del dominio $\partial\Omega$.

El sistema global es obtenido del ensamble de los elementos de los arreglos, el montaje de los nodos polinomiales de los grados de libertad es convencional, los coeficientes del enriquecimiento son la generalización de los grados de libertad internos a cada elemento. La restricción de los grados de libertad están definidos en los elementos de frontera: Vértices, esquinas y caras en una, dos y tres dimensiones respectivamente. El arreglo de elementos es

$$\mathbf{k}^i = \begin{bmatrix} \mathbf{k}^{PP} & \mathbf{k}^{PQ} & \mathbf{k}^{PC} \\ \mathbf{k}^{QP} & \mathbf{k}^{QQ} & \mathbf{k}^{QC} \\ \mathbf{k}^{CP} & \mathbf{k}^{CQ} & 0 \end{bmatrix} \quad (5.122)$$

con la obvia correspondencia entre la matriz global y de elementos. Notemos que para resultados óptimos del siguiente procedimiento a nivel de elementos, los términos provienen de $\langle \mathbf{p}^h \cdot \mathbf{n}, v^P \rangle_{\partial\Omega_i}$ y $\langle \mathbf{q}^h \cdot \mathbf{n}, u^P \rangle_{\partial\Omega_i}$ deben de esta

contenidas en \mathbf{k}^{PC} y \mathbf{k}^{CP} respectivamente, aunque el ensamble cancele a casi todas las entradas a lo largo de la frontera del dominio.

Los grados de libertad del enriquecimiento son eliminados a nivel de elementos para obtener

$$\tilde{\mathbf{k}}^i = \begin{bmatrix} \tilde{\mathbf{k}}^{PP} & \tilde{\mathbf{k}}^{PC} \\ \tilde{\mathbf{k}}^{CP} & \tilde{\mathbf{k}}^{CC} \end{bmatrix} \quad (5.123)$$

donde

$$\begin{aligned} \tilde{\mathbf{k}}^{PP} &= \mathbf{k}^{PP} - \mathbf{k}^{PQ} (\mathbf{k}^{QQ})^{-1} \mathbf{k}^{QP} \\ \tilde{\mathbf{k}}^{PC} &= \mathbf{k}^{PC} - \mathbf{k}^{PQ} (\mathbf{k}^{QQ})^{-1} \mathbf{k}^{QC} \\ \tilde{\mathbf{k}}^{CP} &= \mathbf{k}^{CP} - \mathbf{k}^{CQ} (\mathbf{k}^{QQ})^{-1} \mathbf{k}^{QP} \\ \tilde{\mathbf{k}}^{CC} &= -\mathbf{k}^{CQ} (\mathbf{k}^{QQ})^{-1} \mathbf{k}^{QC} \end{aligned} \quad (5.124)$$

la condensación estática elimina la diagonal por bloques de ceros de la matriz sin condensación.

El sistema global que resulta, reduce $u^P - \mathbf{p}$ la formulación la cual es obtenida como un ensamble del arreglo de elementos. Este sistema en particular es adecuado para soluciones iterativas. La solución para el campo eliminado es obtenida como un post-procesamiento con cada elemento.

5.4.5 Aproximación de los Multiplicadores de Lagrange

Una amplia revisión de técnicas de aproximación de los multiplicadores de Lagrange se han empleado en este método, nos concentraremos en el escalamiento de las funciones base de los multiplicadores de Lagrange usando el factor de escala s con la dimensión de \mathcal{L}_b . Para \mathcal{L} dada, s se escoge tal que los coeficientes de las entradas de las matrices correspondientes a \mathbf{p} y a u sean del mismo orden de magnitud, con la idea de mejorar el condicionamiento de la matriz.

Consideremos a los multiplicadores de Lagrange que sean constantes a lo largo de las caras de un triángulo. Esto se consigue en el presente esquema como las trazas normales de

$$\mathbf{p}^h(x, y) = s \begin{Bmatrix} c_1 + c_3x \\ c_2 + c_3y \end{Bmatrix}, \text{ con } (x, y) \in \Omega_i \quad (5.125)$$

en las caras del triángulo, originalmente denotada por RT_0 . En este caso $\text{div} \mathbf{p}^h = \text{const}$ en Ω_i .

En un triángulo con los multiplicadores de Lagrange que varían linealmente a lo largo de las caras, denotado por BDM, se consigue al considerar nodos en los tres vértices, con la interpolación lineal estándar de los valores en los nodos de \mathbf{p}^h . Los seis grados de libertad de estos elementos puede ser reemplazada por los seis componentes normales de \mathbf{p}^h sobre $\partial\Omega_i$ (dos por cara). Sin embargo, la representación nodal es particular de la adecuada estructura convencional de datos de los elementos finitos.

La aproximación para cuadriláteros se define en términos de las coordenadas naturales en el cuadrado de referencia que está alineado con los ejes cartesianos. En el caso de que la aproximación se especifique en términos de la componente normal de las caras de los elementos, el mapeo del dominio físico es llevado a cabo por un cambio de variables conocida como la transformación de Piola tal que la componente normal es preservada. En el caso, cuando los valores nodales son usados en conjunción con la integración de acuerdo a la integración del lado derecho de la Ec.(5.103) muy probablemente se deberá de usar funciones isoparamétricas.

Considerando multiplicadores de Lagrange que son constantes a lo largo de las caras del cuadrado de referencia. Es obtenida en el presente esquema como las trazas normales de

$$\mathbf{p}^h(x, y) = s \left\{ \begin{array}{l} c_1 + c_2x \\ c_3 + c_4y \end{array} \right\} \quad (5.126)$$

en las caras del cuadrado original denotado BDFM y la cual coincide con RT_0 para rectángulos. En este caso $div\mathbf{p}^h = const$ en Ω_i como para RT_0 . La traza normal es constante, como es requerido y la aproximación puede ser especificada por las cuatro componentes normales de \mathbf{p}^h sobre la frontera.

5.4.6 Condiciones de Frontera Neumann y Robin

Hasta aquí se consideró sólo el caso de condiciones de frontera Dirichlet. Sin embargo la formulación preserva la estructura de las matrices a nivel de elementos de las Ecs.(5.122) y (5.123) en presencia de condiciones de frontera tipo Neumann y Robin.

Considerando una partición de la frontera del dominio $\partial\Omega = \partial\Omega_D \cup \partial\Omega_R$ donde $\partial\Omega_D \cap \partial\Omega_R = \emptyset$. Se asume que las condiciones sobre la frontera tipo Dirichlet sobre toda la frontera del dominio dada por la Ec.(5.91) es reemplazada por

$$u = g \text{ sobre } \partial\Omega_D \quad (5.127)$$

$$\mathcal{L}_b u + \alpha u = \beta \text{ sobre } \partial\Omega_R \quad (5.128)$$

aquí, $g : \partial\Omega_D \rightarrow \mathbb{R}$, $\alpha : \partial\Omega_R \rightarrow \mathbb{R}$ y $\beta : \partial\Omega_R \rightarrow \mathbb{R}$ son funciones dadas. La Ec.(5.128) representa las condiciones de frontera tipo Robin, también las condiciones tipo Neumann en el caso espacial de $\alpha = 0$.

Extendiendo \mathbf{p} a $\partial\Omega_R$ como sigue

$$\mathcal{W} = \{\mathbf{p} \mid \mathbf{p} \in \mathbf{H}(\text{div}, \Omega), \mathbf{p} \cdot \mathbf{n} = -\beta \text{ sobre } \partial\Omega_R\} \quad (5.129)$$

y la funcional dada por la Ec.(5.104) es modificada por

$$\Pi(u, \mathbf{p}) = \frac{1}{2}a(u, u) + \frac{1}{2}\langle \alpha u, u \rangle_{\partial\Omega_R} - \langle \mathbf{p} \cdot \mathbf{n}, v \rangle_{\widetilde{\partial\Omega}} - L(u) - L_b(\mathbf{p}). \quad (5.130)$$

Esto conduce a la modificación de la forma débil

$$a(v, u) + \langle \alpha u, v \rangle_{\partial\Omega_R} - \langle \mathbf{p} \cdot \mathbf{n}, v \rangle_{\widetilde{\partial\Omega}} = L(v) \quad (5.131)$$

$$- \langle \mathbf{q} \cdot \mathbf{n}, u \rangle_{\widetilde{\partial\Omega}} = -L_b(\mathbf{q}) \quad (5.132)$$

donde,

$$\mathbf{q} \in \mathcal{W}_0 = \{\mathbf{q} \mid \mathbf{q} \in \mathbf{H}(\text{div}, \Omega), \mathbf{q} \cdot \mathbf{n} = 0 \text{ sobre } \partial\Omega_R\}. \quad (5.133)$$

La Ec.(5.112) de Euler-Lagrange es ahora reemplazada por

$$\mathbf{p} \cdot \mathbf{n} = \mathcal{L}_b u \text{ sobre } \Gamma \cup \partial\Omega_D \quad (5.134)$$

$$\mathbf{p} \cdot \mathbf{n} = \mathcal{L}_b u + \alpha u \text{ sobre } \partial\Omega_R \quad (5.135)$$

para condiciones de frontera tipo Neumann ($\alpha = 0$) la definición de \mathbf{p} no es cambiada.

La discretización de las formulaciones anteriores están contenidas en las matrices a nivel de elementos de las Ecs.(5.122) y (5.123). El proceso de ensamble ahora cuenta con la imposición de los valores $\mathbf{p} \cdot \mathbf{n}$ sobre $\partial\Omega_R$ como una condición esencial de frontera, de la misma manera que las condiciones de frontera tipo Dirichlet son impuestas en el método de elementos finitos convencional. En otras palabras, los grados de libertad asociados al nivel de elementos con $\mathbf{p} \cdot \mathbf{n}$ sobre $\partial\Omega_R$ no son ensamblados dentro de los coeficientes de la matriz global. En cambio, para datos inhomogéneos ($\beta \neq 0$), se usan

los términos del lado derecho. El proceso de discretización es el dado por las siguientes relaciones

$$\begin{aligned}
 a(v^P, u^P) + \langle \alpha u^P, v^P \rangle_{\partial\Omega_R} &\rightarrow \mathbf{K}^{PP} \\
 a(v^P, u^Q) + \langle \alpha u^Q, v^P \rangle_{\partial\Omega_R} &\rightarrow \mathbf{K}^{PQ} \\
 - \langle \mathbf{p}^h \cdot \mathbf{n}, v^P \rangle_{\partial\Omega_D} &\rightarrow \mathbf{K}^{PC} \\
 a(v^Q, u^P) + \langle \alpha u^P, v^Q \rangle_{\partial\Omega_R} &\rightarrow \mathbf{K}^{QP} \\
 a(v^Q, u^Q) + \langle \alpha u^Q, v^Q \rangle_{\partial\Omega_R} &\rightarrow \mathbf{K}^{QQ} \\
 - \langle \mathbf{p}^h \cdot \mathbf{n}, v^Q \rangle_{\Gamma \cup \partial\Omega_D} &\rightarrow \mathbf{K}^{QC} \\
 - \langle \mathbf{q}^h \cdot \mathbf{n}, v^P \rangle_{\partial\Omega_D} &\rightarrow \mathbf{K}^{CP} \\
 - \langle \mathbf{q}^h \cdot \mathbf{n}, v^Q \rangle_{\Gamma \cup \partial\Omega_D} &\rightarrow \mathbf{K}^{CQ} \\
 L(v^P) + \langle \beta, v^P \rangle_{\partial\Omega_R} &\rightarrow \mathbf{F}^P \\
 L(v^Q) + \langle \beta, v^Q \rangle_{\partial\Omega_R} &\rightarrow \mathbf{F}^Q \\
 L_b(\mathbf{q}^h) &\rightarrow \mathbf{F}^C
 \end{aligned} \tag{5.136}$$

siendo estas las modificaciones para condiciones de frontera tipo Neumann y Robin.

5.5 Métodos de Elementos Finitos Mixtos e Híbridos

En algunas circunstancias es usual aproximar una ecuación diferencial parcial, en algún dominio Ω el cual es a su vez descompuesto en subdominios Ω_E no traslapados, ya sea por que los métodos numéricos usados sean de diferente espacio dimensional y/o por que las mallas no sean iguales en los diferentes subdominios.

El distintivo de estos métodos de aproximación es que, en principio, la solución aproximada puede no ser continua en la sección de unión en las interfaces del subdominio. Es por ello que la aproximación Galerkin no puede ser usada para este tipo de problemas, ya que el espacio de dimensión finita usado hecho de funciones discontinuas no es un subespacio de $H^1(\Omega)$.

Las alternativas para este tipo de problemas consisten en dar una nueva formulación débil del problema el cual sea compatible con los espacios usados Ω_E de manera independiente. Esto se puede hacer mediante los métodos mixtos e híbridos [65].

5.5.1 Métodos Mixtos

Sea $\Omega \subset \mathbb{R}^n$ un dominio y se el operador elíptico

$$\begin{aligned} -\Delta u &= f \quad \text{en } \Omega \\ u &= 0 \quad \text{en } \partial\Omega \end{aligned} \quad (5.137)$$

donde el dominio Ω se asume como un dominio poligonal y f es una función dada en $L^2(\Omega)$. Para obtener la formulación débil sobre la cual la discretización se basa, reescribimos el anterior problema como sigue

$$\begin{aligned} \sigma &= \nabla u, \quad \nabla \cdot \sigma = -f \quad \text{en } \Omega \\ u &= 0 \quad \text{en } \partial\Omega. \end{aligned} \quad (5.138)$$

Entonces la Ec.(5.138) nos lleva directamente al siguiente problema silla: Encontrar $(\sigma, u) \in L^2(\Omega)^n \times H_0^1(\Omega)$ tal que

$$\begin{aligned} (\sigma, \tau)_{0,\Omega} - (\tau, \nabla u)_{0,\Omega} &= 0, \quad \forall \tau \in L^2(\Omega)^n \\ -(\sigma, \nabla v)_{0,\Omega} &= -(f, v), \quad \forall v \in H_0^1(\Omega) \end{aligned} \quad (5.139)$$

estas ecuaciones pueden ser tratadas en el esquema general de problemas con punto silla con los espacios

$$\begin{aligned} X &= L^2(\Omega)^n, \quad M = H_0^1(\Omega) \\ a(\sigma, \tau) &= (\sigma, \tau)_{0,\Omega}, \quad b(\tau, v) = -(\tau, \nabla v)_{0,\Omega} \end{aligned} \quad (5.140)$$

las formas lineales son continuas, donde a es L^2 -elíptica. Para checar la condición inf-sup, usaremos la desigualdad de Friedrichs Ec.(21): Dada $v \in H_0^1(\Omega)$, consideremos el cociente que aparece en la condición para $\tau = -\nabla v \in L^2(\Omega)^n$

$$\frac{b(\tau, v)}{\|\tau\|_0} = \frac{-(\tau, \nabla v)_{0,\Omega}}{\|\tau\|_0} = \frac{(\nabla v, \nabla v)_{0,\Omega}}{\|\nabla v\|_0} = |v|_1 \geq \frac{1}{c} \|v\|_1. \quad (5.141)$$

Entonces c proviene de la desigualdad de Friedrichs y depende solamente de Ω , el problema de punto silla dado por la Ec.(5.139) es estable.

Es fácil construir elementos finitos adecuados para una triangulación \mathcal{T}_h , sin pérdida de generalidad, consideremos el dominio $\Omega \subset \mathbb{R}^2$, el cual es subdividido en E subdominios o elementos Ω_e llamados elementos, tal que

$$\bar{\Omega} = \bigcup_{e=1}^E \bar{\Omega}_e \quad (5.142)$$

donde:

- Cada $\Omega_e \in \mathcal{T}_h$ es un polígono (rectángulo o triángulo) con interior no vacío ($\mathring{\Omega}_e \neq \emptyset$) y conexo.
- Cada $\Omega_e \in \mathcal{T}_h$ tiene frontera $\partial\Omega_e$ Lipschitz continua.
- Para cada $\Omega_i, \Omega_j \in \mathcal{T}_h$ distintos, $\mathring{\Omega}_i \cap \mathring{\Omega}_j = \emptyset$.
- El diámetro $h_i = \text{Diam}(\Omega_e)$ de cada Ω_e satisface $\text{Diam}(\Omega_e) \leq h$ para cada $e = 1, 2, \dots, E$.
- Cualquier cara de cualquier elemento $\Omega_i \in \mathcal{T}_h$ en la triangulación es también un subconjunto de la frontera $\partial\Omega$ del dominio Ω o una cara de cualquier otro elemento $\Omega_j \in \mathcal{T}_h$ de la triangulación, en este último caso Ω_i y Ω_j son llamados adyacentes.
- Los vértices de cada Ω_e son llamados nodos, teniendo N de ellos por cada elemento Ω_e .

Elijamos $k \geq 1$, y definimos los conjuntos

$$\begin{aligned} X_h &= (\mathcal{M}^{k-1})^n = \{\sigma_h \in L^2(\Omega)^n \mid \sigma_h|_{\mathcal{T}} \in P_{k-1} \text{ para } \mathcal{T} \in \mathcal{T}_h\} \quad (5.143) \\ M_h &= \mathcal{M}_{0,0}^k = \{v_h \in H_0^1(\Omega) \mid v_h|_{\mathcal{T}} \in P_k \text{ para } \mathcal{T} \in \mathcal{T}_h\}. \end{aligned}$$

notemos que sólo las funciones en M_h son continuas. Ya que $\nabla M_h \subset X_h$, podemos verificar la condición inf-sup de la misma forma que para problemas continuos.

El problema de punto silla con pares diferentes del tipo

$$H(\text{div}, \Omega) = \{\tau \in L^2(\Omega)^n \mid \text{div } \tau \in L^2(\Omega)\} \quad (5.144)$$

con la norma del operador de divergencia

$$\|\tau\|_{H(\text{div}, \Omega)} = (\|\tau\|_0^2 + \|\text{div } \tau\|_0^2)^{1/2}. \quad (5.145)$$

Buscamos $(\sigma, u) \in H(\text{div}, \Omega) \times L^2(\Omega)$ tal que

$$(\sigma, \tau)_{0, \Omega} + (\text{div } \tau, u)_{0, \Omega} = 0, \quad \forall \tau \in H(\text{div}, \Omega) \quad (5.146)$$

$$(\text{div } \sigma, v)_{0, \Omega} = -(f, v)_{0, \Omega}, \quad \forall v \in L^2(\Omega).$$

Para aplicar la teoría general, tenemos los espacios

$$X = H(\text{div}, \Omega), \quad M = L^2(\Omega) \quad (5.147)$$

$$a(\sigma, \tau) = (\sigma, \tau)_{0,\Omega}, \quad b(\tau, v) = (\operatorname{div} \tau, v)_{0,\Omega}$$

claramente la forma lineal es continua. Entonces ya que la $\operatorname{div} \tau = 0$ para τ en el núcleo de V , tenemos que

$$a(\tau, \tau) = \|\tau\|_0^2 = \|\tau\|_0^2 + \|\operatorname{div} \tau\|_0^2 = \|\tau\|_{H(\operatorname{div}, \Omega)}^2. \quad (5.148)$$

Esto establece la elípticidad de a sobre el núcleo. Además, para $v \in L^2$ dada, existe $w \in C_0^\infty(\Omega)$ con $\|v - w\|_{0,\Omega} \leq \frac{1}{2} \|v\|_{0,\Omega}$. Sea $\xi = \inf \{x_1, x \in \Omega\}$ y

$$\begin{aligned} \tau_1(x) &= \int_{\xi}^{x_1} w(t, x_2, \dots, x_n) dt \\ \tau_i(x) &= 0, \quad \text{para } i > 0. \end{aligned} \quad (5.149)$$

Entonces obviamente $\operatorname{div} \tau = \frac{\partial \tau_1}{\partial x_1} = w$, y el mismo argumento se aplica para la desigualdad de Friedrich dada por $\|\tau\|_0 \leq c \|w\|_0$. Por ello

$$\frac{b(\tau, v)}{\|\tau\|_{H(\operatorname{div}, \Omega)}} \geq \frac{(w, v)_{0,\Omega}}{(1+c)\|w\|_{0,\Omega}} \geq \frac{1}{2(1+c)} \|v\| \quad (5.150)$$

y así la condición inf-sup es satisfecha.

A primera vista, aparentemente la solución existe sólo si $u \in L^2$. Sin embargo, $u \in H_0^1(\Omega)$, y entonces $C_0^\infty(\Omega)^n \subset H(\operatorname{div}, \Omega)$, la primera ecuación de la Ec.(5.146) dice que en particular que

$$\int_{\Omega} u \frac{\partial \tau_i}{\partial x_i} dx = - \int_{\Omega} \sigma_i \tau_i dx, \quad \text{para } \tau_i \in C_0^\infty(\Omega). \quad (5.151)$$

Por lo tanto u posee una derivada débil $\frac{\partial u}{\partial x_i} = \sigma_i$, y como $u \in H^1(\Omega)$. Ahora, la Ec.(5.146) junto con la fórmula de Green y $\nabla u = \sigma$ implica

$$\begin{aligned} \int_{\partial\Omega} u \cdot \tau v ds &= \int_{\Omega} \nabla u \cdot \tau dx + \int_{\Omega} \operatorname{div} \tau u dx \\ &= \int_{\Omega} \sigma \cdot \tau dx + \int_{\Omega} \operatorname{div} \tau u dx = 0 \end{aligned} \quad (5.152)$$

ya que se satisface para toda $\tau \in C^\infty(\Omega)^n$, tenemos que $u = 0$ sobre la frontera en el sentido más general. i.e. de hecho $u \in H_0^1(\Omega)$.

En el caso estándar la condición natural de frontera es $\frac{\partial u}{\partial \nu} = 0$, pero aquí la condición natural de frontera es $u = 0$.

Notemos que la Ec.(5.139) caracteriza la solución del problema variacional

$$\begin{aligned} \frac{1}{2}(\sigma, \sigma)_0 - (f, u) &\rightarrow \min & (5.153) \\ \nabla u - \sigma &= 0 \end{aligned}$$

donde el multiplicador de Lagrange coincide con σ , y puede ser eliminado de la ecuación, por otro lado la Ec.(5.146) suge del problema variacional

$$\begin{aligned} \frac{1}{2}(\sigma, \sigma)_0 &\rightarrow \min & (5.154) \\ \operatorname{div} \sigma + f &= 0 \end{aligned}$$

en este caso, el multiplicador de Lagrange coincide con u de la Ec.(5.138).

Algunas veces la Ec.(5.139) con $X = L^2$ y $M = H_0^1$ es llamado método primal mixto, mientras la Ec.(5.146) con $X = H(\operatorname{div})$ y $M = L^2$ es llamado el método mixto dual. El dual del problema variacional mixto está relacionado con la estimación del error.

Teorema 45 (*Prager y Synge*)

Sea $\sigma \in H(\operatorname{div})$, $v \in H_0^1(\Omega)$ y asumase que $\operatorname{div} \sigma + f = 0$. Además, sea u una solución de la ecuación de Poisson $\Delta u = -f$ con condiciones de frontera cero. Entonces

$$|u - v|_1^2 + \|\operatorname{grad} u - \sigma\|_0^2 = \|\operatorname{grad} v - \sigma\|_0^2. \quad (5.155)$$

Los Elementos de Raviart-Thomas Los elementos de Raviart y Thomas son adecuados para el problema de punto silla dado por la Ec.(5.146). Sea $k \geq 0$, $\Omega \subset \mathbb{R}^2$, y supóngase una triangulación regular \mathcal{T}_h y tal que

$$X_h = RT_k = \left\{ \tau \in L^2(\Omega)^2 \mid \tau|_T = \begin{pmatrix} a_T \\ b_T \end{pmatrix} + c_T \begin{pmatrix} x \\ y \end{pmatrix}, \right. \quad (5.156)$$

$a_T, b_T, c_T \in P_k$ para $T \in \mathcal{T}_h$, $\tau \cdot \nu$ es continuo sobre la frontera inter-elemento}

$$M_h = \mathcal{M}^k(\mathcal{T}_h) = \left\{ v \in L^2(\Omega) \mid v|_T \in P_k \text{ para } T \in \mathcal{T}_h \right\} \quad (5.157)$$

la continuidad de la componente normal sobre la frontera asegura la conformabilidad $X_h \subset H(\operatorname{div}, \Omega)$.

Por conveniencia, consideramos elementos de Raviart-Thomas sólo para $k = 0$. La construcción depende estrechamente de las siguientes aseveraciones. Las funciones en $(P_1)^2$ que tienen la forma

$$p = \begin{pmatrix} a \\ b \end{pmatrix} + c \begin{pmatrix} x \\ y \end{pmatrix} \quad (5.158)$$

están caracterizadas por el hecho de que $n \cdot p$ es constante en cada línea $\alpha x + \beta y = cte$ siempre que n sea ortogonal a la línea. Por lo tanto, dado un triángulo T , la componente normal es constante y puede ser prescrita sobre cada borde de T , formalmente, el elemento de Raviart-Thomas es la tripleta

$$(T, (P_0)^2 + x \cdot P_0, n_i p(z_i)) \quad (5.159)$$

con $i = 1, 2, 3$ con z_i en el punto medio de cada borde i .

La resolubilidad del problema de interpolación es fácilmente verificada, dado un vértice a_i de T , podemos hallar un vector $v_i \in \mathbb{R}^2$ tal que es las proyecciones sobre las normales de los bordes adyacentes tienen valores prescritos. Ahora determinamos $p \in (P_1)^2$ tal que

$$p(a_i) = v_i, \text{ con } i = 1, 2, 3 \quad (5.160)$$

es la forma intermedia $p \in (P_1)^2$ que las componentes normales son lineales en cada borde del triángulo. Ellas son planos constantes, ya que por construcción llegan al mismo valor en ambos vértices del borde. De este modo las funciones construidas necesariamente pertenecen a un conjunto específico de $(P_1)^2$.

Un operador de Interpolación Sea $k \geq 0$ y T un triángulo. Definimos

$$\rho_T : H^1(T) \rightarrow RT_k(T) \quad (5.161)$$

por

$$\int_e (q - \rho_T q) \cdot n p_k ds = 0 \quad (5.162)$$

para todo $p_k \in P_k$ y cada borde $e \in \partial T$ y

$$\int_e (q - \rho_T q) \cdot p_{k-1} dx = 0 \quad (5.163)$$

para todo $p_{k-1} \in P_{k-1}^2$ (si $k \geq 1$).

Dada una triangulación \mathcal{T} sobre Ω , definimos $\rho_\Omega : H^1(\Omega) \rightarrow RT_k$ localmente por

$$(\rho_\Omega q)|_T = \rho_T(q|_T) \quad (5.164)$$

para toda $T \in \mathcal{T}$.

Nos restringimos al caso cuando $k = 0$. Recordando que la componente normal de $v \in RT_0$ es constante sobre cada borde. La Ec.(5.162) expresa que coincide con el valor promedio del componente normal de la función dada. Esto se satisface por la solución del problema de interpolación.

Del teorema de la integral de Gauss se concluye que

$$\int_T \operatorname{div}(q - \rho_T q) dx = \sum_{e \in \partial T} \int_e (q - \rho_T q) \cdot n ds = 0. \quad (5.165)$$

por otro lado, los elementos de Raviart-Thomas son lineales por pedazos y $\alpha = \operatorname{div} \rho_T q$ es constante sobre T . Por la Ec.(5.165) α es el valor promedio de $\operatorname{div} v$ sobre T . Por lo tanto α es la constante con la menor desviación de $\operatorname{div} q$. Tal que se ha establecido la siguiente propiedad para $k = 0$.

Propiedad Minimal Dada una triangulación \mathcal{T} sobre Ω . Sea Π_k la L^2 -proyección sobre \mathcal{M}^k . Entonces tenemos que para todo $q \in H^1(\Omega)$

$$\operatorname{div}(q - \rho_T q) = \Pi_k \operatorname{div} q \quad (5.166)$$

esta ecuación es llamada la propiedad del diagrama conmutativo

$$\begin{array}{ccc} H^1(\Omega) & \xrightarrow{\operatorname{div}} & L^2(\Omega) \\ \rho_\Omega \downarrow & & \downarrow \Pi_k \\ RT_k & \xrightarrow{\operatorname{div}} & \mathcal{M}^k \end{array} \quad (5.167)$$

Lema 46 *La función*

$$\operatorname{div} : RT_0 \rightarrow \mathcal{M}^0 \quad (5.168)$$

es suprayectiva.

Lema 47 *Sea \mathcal{T}_h sea una triangulación regular del dominio Ω . Entonces*

$$\|q - \rho_\Omega q\|_{H(\operatorname{div}, \Omega)} \leq ch |q|_1 + \inf_{v_h \in \mathcal{M}^0} \|\operatorname{div} q - v_h\|_0. \quad (5.169)$$

Ahora la estimación del error de la solución por elementos finitos de la Ec.(5.138)

$$\|\sigma - \sigma_h\|_{H(\text{div}, \Omega)} + \|u - u_h\|_0 \leq c \left(h |\sigma|_1 + h \|u\|_1 + \inf_{f_h \in \mathcal{M}^0} \|f - f_h\|_0 \right). \quad (5.170)$$

Implementación y Postprocesamiento En principio, la discretización genera un sistema indefinido de ecuaciones. Puede ser llevado a un sistema positivo definido mediante un procedimiento.

Para ello se elige los gradientes que pertenezcan al subespacio de $H(\text{div}, \Omega)$, primeramente admitimos gradientes en $L^2(\Omega)^2$, y después explícitamente se requiere $\text{div } \sigma_h \in L^2(\Omega)$. Equivalentemente, requerimos que los componentes normales $\sigma_h n$ no existan saltos sobre los bordes. Para lograr esto, forzamos la continuidad de $\sigma_h n$ sobre los bordes mediante una restricción explícita. Esto introduce a los multiplicadores de Lagrange.

La aproximación de las funciones para σ_h no requieren condiciones de continuidad, y en cada función base tiene soporte sobre un sólo triángulo. Si eliminamos las variables asociadas por condensación estática, las ecuaciones resultantes son dispersas como antes del proceso de eliminación. En adición, hemos eludido la costosa construcción de la base de elementos de Raviart-Thomas.

Una ventaja adicional es que los multiplicadores de Lagrange pueden ser considerados como una aproximación por elementos finitos de u sobre los bordes.

Normas Dependientes de la Malla para los Elementos de Raviart-Thomas El cálculo de elementos finitos con elementos de Raviart-Thomas puede ser analizada desde el esquema de método mixto primal, i.e. con el par $H^1(\Omega), L^2(\Omega)$. Ya que la componente tangencial de las funciones en la Ec.(5.156) pueden tener saltos en la frontera entre elementos, en este contexto los elementos no son conformes y necesitamos normas que dependan de la

mallas

$$\begin{aligned} \|\tau\|_{0,h} &= \left(\|\tau\|_0^2 + h \sum_{e \in \Gamma_h} \|\tau n\|_{0,e}^2 \right)^{1/2} \\ |v|_{1,h} &= \left(\sum_{T \in \Gamma_h} |v|_{1,T}^2 + h^{-1} \sum_{e \in \Gamma_h} \|[[v]]\|_{0,e}^2 \right)^{1/2} \end{aligned} \quad (5.171)$$

donde $\Gamma_h = \cup_T (\partial T \cap \Omega)$ es el conjunto de fronteras entre elementos. Sobre los bordes de Γ_h el salto $[[v]]$ de v y la componente normal $\tau \cdot n$ de τ están bien definidas. Nótese que ambas $\tau \cdot n$ y $[[v]]$ cambian de signo si la orientación de los bordes se toman en sentido contrario. Por lo tanto, el producto es independiente de la orientación.

La continuidad de la forma bilineal $a(\cdot, \cdot)$ es obvia. La coercitividad se sigue de

$$\|\tau\|_{0,h} \leq C \|\tau\|_0 \quad (5.172)$$

para toda $\tau \in RT_k$ el cual es obtenido por el escalamiento del argumento estándar. La forma bilineal $b(\cdot, \cdot)$ es reescrita por el uso de la fórmula de Green

$$b(\tau, v) = - \sum_{T \in \Gamma_h} \int_T \tau \cdot \text{grad } v dx + \int_{\Gamma_h} [[v]] \tau n ds \quad (5.173)$$

ahora, la continuidad con respecto a la norma dada por la Ec.(5.171) es inmediata.

Lema 48 *La condición inf-sup es*

$$\sup_{\tau \in RT_k} \frac{b(\tau, v)}{\|\tau\|_{0,h}} \geq \beta |v|_{1,h} \quad (5.174)$$

para todo $v \in \mathcal{M}^k$ se satisface con la constante $\beta > 0$ la cual depende solamente de k y de la forma del parámetro de la triangulación \mathcal{T}_h .

Comportamiento Débil de los Métodos Mixtos En el método primal mixto dado por la Ec.(5.139) genera una forma débil de la forma cuadrática $a(\cdot, \cdot)$, sea $u_h \in M_h \subset H_0^1(\Omega)$ y $\sigma_h \in X_h \subset L^2(\Omega)$ la solución del método mixto

$$(\sigma_h, \tau)_{0,\Omega} - (\tau, \nabla u_h)_{0,\Omega} = 0, \quad \forall \tau \in X_h \quad (5.175)$$

$$-(\sigma_h, \nabla v)_{0,\Omega} = -(f, v), \quad \forall v \in M_h$$

si $E_h = \nabla M_h \subset X_h$, entonces la primera ecuación implica $\sigma_h = \nabla u_h$, y la Ec.(5.175) es equivalente al clásico tratamiento de la ecuación de Poisson con el espacio de elementos finitos M_h , siendo este caso poco interesante.

El caso más interesante es cuando $E_h \subsetneq X_h$. Sea $P_h : L^2(\Omega) \rightarrow X_h$ la proyección ortogonal sobre X_h . La primera ecuación en la Ec.(5.175) se lee

$$\sigma_h = P_h(\nabla u_h) \quad (5.176)$$

y la segunda

$$(P_h \nabla u_h, \nabla v)_{0,\Omega} = (f, v) \quad (5.177)$$

para toda $v \in M_h$.

Esta es la ecuación débil para el problema de minimización relajado

$$\frac{1}{2} \int_{\Omega} [P_h \nabla v_h]^2 dx - \int_{\Omega} f v_h \rightarrow \min_{v_h \in M_h} \quad (5.178)$$

sólo la parte del gradiente que se proyecta sobre X_h contribuye a la energía en la formulación variacional. El grado de debilidad es fijada por la elección del espacio destino de la proyección.

Otra caracterización de las ecuaciones variacionales dada por la Ec.(5.175) puede ser reescrita en la forma en la cual genera ecuaciones lineales con una matriz positiva definida. Escogemos un subespacio \tilde{E}_h del complemento ortogonal L^2 con respecto a X_h tal que

$$\nabla M_h \subset X_h \oplus \tilde{E}_h. \quad (5.179)$$

El método mixto dado por la Ec.(5.175) es equivalente a la formulación variacional

$$(\nabla u_h, \nabla v)_{0,\Omega} + (\tilde{\varepsilon}_h, \nabla v)_{0,\Omega} = (f, v)_{0,\Omega} \quad (5.180)$$

para toda $v \in M_h$ y

$$(\nabla u_h, \tilde{\eta})_{0,\Omega} + (\tilde{\varepsilon}_h, \eta)_{0,\Omega} = 0 \quad (5.181)$$

para toda $\eta \in \tilde{E}_h$, si el espacio \tilde{E}_h es enriquecido con gradientes que satisfacen la Ec.(5.179). Aquí la relajación de la forma variacional y de la proyección P_h son definidas por \tilde{E}_h , i.e por el complemento ortogonal del espacio destino.

La estabilidad del método mixto puede establecerse por el enriquecimiento de elementos y se muestra que la estabilidad no depende de la elección del espacio \tilde{E}_h .

Lema 49 *Los espacios X_h y M_h satisfacen la condición inf-sup con la constante $\beta \geq 0$ si y sólo si satisface la desigualdad fuerte de Cauchy*

$$(\nabla u_h, \tilde{\eta}_h)_{0,\Omega} \leq \sqrt{1 - \beta^2} \|\nabla v_h\|_0 \|\eta_h\|_0 \quad (5.182)$$

para todo $v_h \in M_h$ y $\eta_h \in \tilde{E}_h$.

5.5.2 Métodos Híbridos

Sea Ω un dominio, consideremos el caso en el cual el dominio es dividido en dos subdominios Ω_1 y Ω_2 por una frontera interior suave Γ . Consideramos el caso de un problema con condiciones de frontera tipo Dirichlet con coeficiente variable $a(x)$, donde $a(x)$ es discontinuo sobre Γ . La formulación variacional de este problema es:

Encontrar $u \in H_0^1(\Omega)$ tal que

$$\int_{\Omega_1} a_1(x) \text{grad } u \cdot \text{grad } v dx + \int_{\Omega_2} a_2(x) \text{grad } u \cdot \text{grad } v dx = \int_{\Omega} f v dx \quad (5.183)$$

para toda $v \in H_0^1(\Omega)$. Definiendo $u_1 = u|_{\Omega_1}$ y $u_2 = u|_{\Omega_2}$, la interpretación formal del problema dado por la Ec.(5.183) queda en la forma

$$\begin{cases} -\text{div}(a_1(x) \text{grad } u_1) = f, & \text{en } \Omega_1 \\ -\text{div}(a_2(x) \text{grad } u_2) = f, & \text{en } \Omega_2 \\ u_1|_{\Gamma \cap \partial\Omega_1} = 0, u_2|_{\Gamma \cap \partial\Omega_2} = 0 \end{cases} \quad (5.184)$$

$$u_1 = u_2, \text{ sobre } \Gamma \quad (5.185)$$

$$a_1 \frac{\partial u_1}{\partial n_1} + a_2 \frac{\partial u_2}{\partial n_2} = 0, \text{ sobre } \Gamma$$

donde n_1 y n_2 son las normales exteriores en Γ a los subdominios Ω_1 y Ω_2 respectivamente. La condición de continuidad dada por la Ec.(5.185) está implícitamente contenida en la formulación variacional. Un importante caso especial se tiene cuando $a_1(x) = a_2(x) = 1$, se tiene el siguiente resultado.

Sea u una solución del problema de Dirichlet

$$\begin{cases} -\Delta u = f \\ u|_{\Gamma} = 0 \end{cases} \quad (5.186)$$

donde Γ es la frontera interna de los subdominios Ω_1 y Ω_2 del dominio Ω . Entonces el problema anterior es equivalente a buscar la solución u del problema

$$\begin{cases} -\Delta u_1 = f, & \text{en } \Omega_1 \\ -\Delta u_2 = f, & \text{en } \Omega_2 \\ u_1|_{\Gamma \cap \partial\Omega_1} = 0, u_2|_{\Gamma \cap \partial\Omega_2} = 0 \end{cases} \quad (5.187)$$

$$\begin{aligned} u_1 &= u_2, & \text{sobre } \Gamma & \quad (5.188) \\ \frac{\partial u_1}{\partial n_1} + \frac{\partial u_2}{\partial n_2} &= 0, & \text{sobre } \Gamma & \end{aligned}$$

para mostrar este resultado es necesario definir propiedades de las derivadas normales $\frac{\partial u_1}{\partial n_1}$ y $\frac{\partial u_2}{\partial n_2}$ sobre Γ . Esto requiere de alguna regularidad de f , por ejemplo $f \in L^2(\Omega)$.

Método de Descomposición de Dominio para el Problema Dirichlet

Consideraremos ahora el problema de Laplace con condiciones de frontera sobre el dominio Ω , sea el problema de minimización definido sobre $H_0^1(\Omega)$ y donde $f \in L^2(\Omega)$ es dada

$$\inf_{v \in H_0^1(\Omega)} \frac{1}{2} \int_{\Omega} |\text{grad } v|^2 dx - \int_{\Omega} f v dx \quad (5.189)$$

donde $|\text{grad } v|^2 = \left| \frac{\partial v}{\partial x_1} \right|^2 + \left| \frac{\partial v}{\partial x_2} \right|^2 = \text{grad } v \cdot \text{grad } v$. Este problema tiene una única solución u caracterizada porque $u \in H_0^1(\Omega)$ y

$$\int_{\Omega} \text{grad } u \cdot \text{grad } v dx = \int_{\Omega} f v dx \quad (5.190)$$

para toda $v \in H_0^1(\Omega)$.

Si u es solución, entonces satisface, en el sentido de distribuciones

$$\begin{cases} -\Delta u = f, & \text{en } \Omega \\ u|_{\Gamma} = 0 \end{cases} \quad (5.191)$$

el cual es un problema estándar de Dirichlet. Si $H_0^1(\Omega)$ es reemplazado por $H_{0,D}^1(\Omega)$, entonces del problema dado por la Ec.(5.191) se convierte en un problema mixto

$$\begin{cases} -\Delta u = f, & \text{en } \Omega \\ u|_D = 0 \\ \frac{\partial u}{\partial n}|_N = 0 \end{cases} \quad (5.192)$$

en el cual tendremos condiciones de frontera tipo Dirichlet en D y condiciones de frontera tipo Neumann en N . En particular para $N = \Gamma$, tendremos un problema de Neumann. Notemos que el problema de minimización dado por la Ec.(5.189) sobre $H^1(\Omega)$ en lugar de $H_0^1(\Omega)$ pudiendo definir a u salvo una constante aditiva, y requerimos las condiciones de compatibilidad $\int_{\Omega} f dx = 0$, que pueden ser necesarias de la Ec.(5.190), tomando $v = 1$ en Ω .

Denotando por $H^{-1/2}(\Gamma)$ al espacio dual de $H^{1/2}(\Gamma)$ y tomando

$$g \in H^{-1/2}(\Gamma) \quad (5.193)$$

entonces, consideramos la funcional

$$\frac{1}{2} \int_{\Omega} |\text{grad } v|^2 dx - \int_{\Omega} f v dx - \langle g, v \rangle \quad (5.194)$$

donde $\langle \cdot, \cdot \rangle$ denota la dualidad entre $H^{-1/2}(\Gamma)$ y $H^{1/2}(\Gamma)$, en algunas ocasiones se suele escribir esto como $\int_{\Gamma} g v ds$ en lugar de $\langle g, v \rangle$. Minimizando la Ec.(5.194) sobre $H_{0,D}^1(\Omega)$ genera el problema

$$\begin{cases} -\Delta u = f, & \text{en } \Omega \\ u|_D = 0 \\ \frac{\partial u}{\partial n}|_N = g \end{cases} \quad (5.195)$$

cuando $D = \emptyset$ la solución es definida salvo una constante aditiva y podemos elegir f y g tal que $\int_{\Omega} f dx - \int_{\Gamma} g ds = 0$.

Consideremos el problema dado por la Ec. (5.195) en el dominio Ω , el cual es subdividido en E subdominios Ω_i , $i = 1, \dots, E$ sin traslape, es decir

$$\Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j \quad \text{y} \quad \bar{\Omega} = \bigcup_{i=1}^E \bar{\Omega}_i, \quad (5.196)$$

y al conjunto

$$\Gamma = \bigcup_{i=1}^E \Gamma_{\delta}, \quad \text{si } \Sigma_i = \partial\Omega_i \setminus \partial\Omega \quad (5.197)$$

lo llamaremos la frontera interior del dominio Ω , denotamos por H al diámetro $H_i = \text{Diam}(\Omega_i)$ de cada Ω_i que satisface $\text{Diam}(\Omega_i) \leq H$ para cada $i = 1, 2, \dots, E$, además, cada subdominio Ω_i es descompuesto en una mallado fino

\mathcal{T}_h de K subdominios mediante una triangulación Ω_e de modo que esta sea conforme, denotamos por h al diámetro $h_i = \text{Diam}(\Omega_e)$ de cada Ω_e que satisface $\text{Diam}(\Omega_e) \leq h$ para cada $e = 1, 2, \dots, K$ de cada $i = 1, 2, \dots, E$.

Entonces el método de descomposición de dominio para el problema de Dirichlet es reescrito como la funcional

$$J(v) = \sum_{i=1}^E \left\{ \frac{1}{2} \int_{\Omega_i} |\text{grad } v|^2 dx - \int_{\Omega_i} f v dx \right\} \quad (5.198)$$

introduciendo el espacio funcional

$$X(\Omega) = \{v \mid v \in L^2(\Omega), v|_{\Omega_i} \in H^1(\Omega_i)\} \approx \prod_{i=1}^E H^1(\Omega_i) \quad (5.199)$$

podemos extender $J(v)$ sobre $X(\Omega)$. Por otra parte $H_0^1(\Omega)$ es un subespacio cerrado de $X(\Omega)$ y podemos considerar " $v \in H_0^1(\Omega)$ " como una restricción lineal sobre $v \in X(\Omega)$. Esta restricción fija que sobre $e_{ij} = \partial\Omega_i \cap \partial\Omega_j$ tenemos, en $H^{-1/2}(e_{ij})$ que $u_i = u_j$, donde $u_i = u|_{\Omega_i}$. Aplicando un procedimiento en el cual se imponen restricciones mediante los Multiplicadores de Lagrange adecuadamente elegidos en $H^{-1/2}(e_{ij})$, pero es mejor introducir $\underline{q} \in H(\text{div}, \Omega)$ y usar como multiplicadores la traza normal de \underline{q} sobre $\partial\Omega_i$. Esto genera el problema de punto silla

$$\inf_{v \in X(\Omega)} \sup_{\underline{q} \in H(\text{div}, \Omega)} \sum_{i=1}^E \left\{ \frac{1}{2} \int_{\Omega_i} |\text{grad } v|^2 dx - \int_{\partial\Omega_i} \underline{q} \cdot \underline{n}_i v ds - \int_{\Omega_i} f v dx \right\} \quad (5.200)$$

para los cuales se tienen las siguientes condiciones de optimización: para $i = 1, \dots, E$, encontrar $u_i \in H^1(\Omega_i)$ tal que

$$\int_{\Omega_i} \text{grad } u_i \cdot \text{grad } v_i dx = \int_{\Omega_i} f v_i dx + \int_{\partial\Omega_i} \underline{p} \cdot \underline{n}_i v_i ds \quad (5.201)$$

para toda $v_i \in H^1(\Omega_i)$

$$\sum_{i=1}^E \int_{\partial\Omega_i} \underline{q} \cdot \underline{n}_i u_i ds = 0 \quad (5.202)$$

para todo $\underline{q} \in H(\text{div}, \Omega)$, esta última condición expresa la continuidad de u en la interfase e_{ij} y la condición $u|_{\Gamma} = 0$. La condición dada por la Ec.(5.201) muestra que u_i es solución en Ω_i del problema de Neumann

$$\begin{cases} -\Delta u_i = f, & \text{en } \Omega_i \\ \frac{\partial u_i}{\partial n_i} = \underline{p} \cdot \underline{n}_i, & \text{en } \partial\Omega_i \end{cases} \quad (5.203)$$

la resolución de este problema requiere una condición de compatibilidad (hacer $v_i = 1$ en la Ec.(5.201))

$$\int_{\partial\Omega_i} \underline{p} \cdot \underline{n}_i ds + \int_{\Omega_i} f dx = 0 \quad (5.204)$$

sobre cada subdominio Ω_i . Esta condición puede ser escrita como

$$\int_{\Omega_i} (\operatorname{div} \underline{p} + f) dx = 0. \quad (5.205)$$

De la Ec.(5.204) tenemos que el multiplicador $\underline{p} \cdot \underline{n}$ puede ser visto como la derivada normal de u . En efecto, cuando el equilibrio se alcanza, tenemos en las interfases

$$\frac{\partial u_i}{\partial n_i} = \underline{p} \cdot \underline{n}_i = -\underline{p} \cdot \underline{n}_j = -\frac{\partial u_j}{\partial n_j} \quad (5.206)$$

y $u_i = u_j$. Un adecuado levantamiento de \underline{p} en cada Ω_i para tener $\operatorname{div} \underline{p} + f = 0$ siempre se puede hacer por las Ecs.(5.204) y (5.205).

Dual del Problema del método de Descomposición de Dominio

Considerando el dual del problema de la formulación de punto silla anterior. Primeramente un comentario, si tomamos el ínfimo sobre la parte constante de $v \in X(\Omega)$ sobre cada Ω_i tenemos una restricción a la Ec.(5.205) sobre \underline{p} . Es por lo tanto posible suponer $\operatorname{div} \underline{p} + f = 0$, esto puede ser hecho por las modificaciones a p que es interna a Ω_i (esto es, no es necesario modificar $\underline{p} \cdot \underline{n}_i$) y es transparente la formulación de la Ec.(5.200). Escribimos

$$\int_{\partial\Omega_i} \underline{q} \cdot \underline{n}_i v ds = \int_{\Omega_i} \operatorname{div} \underline{q} v dx + \int_{\Omega_i} \underline{q} \operatorname{grad} v dx \quad (5.207)$$

y obtenemos de la Ec.(5.200)

$$\sup_{\operatorname{div} \underline{q} + f = 0} \inf_{v_i \in H^1(\Omega_i)/\mathbb{R}} \sum_{i=1}^E \left\{ \frac{1}{2} \int_{\Omega_i} |\operatorname{grad} v_i|^2 dx - \int_{\Omega_i} \underline{q} \cdot \operatorname{grad} v_i dx \right\} \quad (5.208)$$

de esta última ecuación, es evidente obtener que $\underline{p}_i = \underline{p}|_{\Omega_i}$,

$$\operatorname{grad} u_i = P \left(\underline{p}_i \right) \quad (5.209)$$

donde P es el operador proyección en $(L^2(\Omega_i))^2$ sobre $\text{grad}(H^1(\Omega_i))$, donde

$$(L^2(\Omega))^n = \{\text{grad } H^1(\Omega)\} \oplus \{\text{rot } H_0^1(\Omega)\} \quad (5.210)$$

de la cual v_i puede ser eliminado y reescribir el problema dual

$$\sup_{\underline{q} \in H(\text{div}, \Omega) \text{ y } \text{div } \underline{q} + f = 0} -\frac{1}{2} \sum_{i=1}^E \int_{\Omega_i} \left| P(\underline{q}_i) \right|^2 dx \quad (5.211)$$

el cual se puede reescribir como

$$\inf \frac{1}{2} \int_{\Omega} |\underline{q}|^2 dx \quad (5.212)$$

para todo $\underline{q} \in \left\{ \underline{q} \in (L^2(\Omega_i))^2 \mid \text{div } \underline{q} + f = 0 \right\}$, en esta última ecuación se ve que el operador proyección no es necesario.

Método Dual Híbrido Ahora consideremos el problema dual a la Ec.(5.212), esto es,

$$\inf_{\underline{q} \in H(\text{div}, \Omega) \text{ y } \text{div } \underline{q} + f = 0} \frac{1}{2} \int_{\Omega} |\underline{q}|^2 dx \quad (5.213)$$

aplicando el principio de la descomposición de dominio a este problema, introducimos

$$Y(\Omega) = \left\{ \underline{q} \mid \underline{q}|_{\Omega_i} \in H(\text{div}, \Omega_i) \right\} \approx \prod_{i=1}^E H(\text{div}, \Omega_i) \quad (5.214)$$

así, $H(\text{div}, \Omega)$ es ahora un subespacio cerrado de $Y(\Omega)$ caracterizado por

$$\sum_{i=1}^E \int_{\Omega_i} (\underline{p} \cdot \underline{n}_i) v ds = 0 \quad (5.215)$$

para toda $v \in H_0^1(\Omega)$.

Entonces podemos transformar la Ec.(5.213) en un problema de punto silla

$$\inf_{\underline{q} \in Y(\Omega)} \sup_{v \in H_0^1(\Omega)} \sum_{i=1}^E \left\{ \frac{1}{2} \int_{\Omega_i} |\underline{q}_i|^2 dx + \int_{\partial\Omega_i} \underline{q}_i \cdot \underline{n}_i v ds \right\} \quad (5.216)$$

bajo las restricciones locales

$$\text{div } \underline{q} + f = 0, \text{ sobre } \Omega_i, \quad (5.217)$$

la ventaja de esta formulación es que es fácil encontrar \underline{q}_i que satisfaga la Ec.(5.217).

6 Método FETI

Algunas de las formulaciones más conocidas de los métodos de descomposición de dominio están basadas en un análisis de las condiciones de transmisión entre las interfaces del subdominio, los cuales usan el operador de Steklov-Poincaré. Nos interesa el estudio sistemático de algunas de las mejores formulaciones de métodos de descomposición de dominio que se basen en la aplicación del operador de Steklov-Poincaré. Estos métodos están basados en una aproximación especial de las fórmulas de Green aplicable a funciones discontinuas.

El concepto unificador básico de la teoría, consiste en interpretar los métodos de descomposición de dominio como procedimientos para obtener información acerca de la solución en la frontera interior la cual separa el subdominio de cada uno de los otros, suficiente para definir problemas bien planteados en cada uno de los subdominios - referidos como problemas locales-. De esta manera, la solución puede ser reconstruida al resolver cada uno de los problemas locales exclusivamente.

La familia de algoritmos FETI (Finite Element Tearing and Interconnecting) y Neumann-Neumann [8] y [58] son de los métodos mejor conocidos y más probados para la resolución de ecuaciones diferenciales parciales elípticas. Ellos son métodos iterativos de subestructuración Sec.(19.7) y comparten muchos componentes algorítmicos, tales como soluciones locales para ambos problemas con condiciones de frontera Neumann y Dirichlet sobre las subregiones en donde el problema fue particionado.

6.1 Conceptos Básicos

En este capítulo se considerarán problemas con valor en la frontera (VBVP) de la forma

$$\begin{aligned} -\nabla u &= f & \text{en } \Omega \\ u &= g & \text{en } \partial\Omega \end{aligned} \tag{6.1}$$

entonces el problema dado por la Ec. (6.1) se reescribe como: hallar $\underline{u} \in H_0^1(\Omega)$ tal que $a(\underline{u}, \underline{v}) = l(\underline{v})$.

Consideremos el problema dado por la Ec. (6.1) en el dominio Ω , el cual es en general subdividido en E subdominios Ω_i , $i = 1, \dots, E$ sin traslape, es

decir

$$\Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j \quad \text{y} \quad \bar{\Omega} = \bigcup_{i=1}^E \bar{\Omega}_i, \quad (6.2)$$

y al conjunto

$$\Gamma = \bigcup_{i=1}^E \Gamma_i, \quad \text{si } \Gamma_i = \partial\Omega_i \setminus \partial\Omega \quad (6.3)$$

lo llamaremos la frontera interior del dominio Ω , denotamos por H al diámetro $H_i = \text{Diam}(\Omega_i)$ de cada Ω_i que satisface $\text{Diam}(\Omega_i) \leq H$ para cada $i = 1, 2, \dots, E$, además, cada subdominio Ω_i es descompuesto en una mallado fino \mathcal{T}_h de K subdominios mediante una triangulación Ω_e de modo que esta sea conforme, denotamos por h al diámetro $h_i = \text{Diam}(\Omega_e)$ de cada Ω_e que satisface $\text{Diam}(\Omega_e) \leq h$ para cada $e = 1, 2, \dots, K$ de cada $i = 1, 2, \dots, E$.

Un ejemplo de un dominio Ω y su descomposición en subdominios Ω_i y cada Ω_i a su vez descompuesto en Ω_e subdominios como se muestra en la figura:

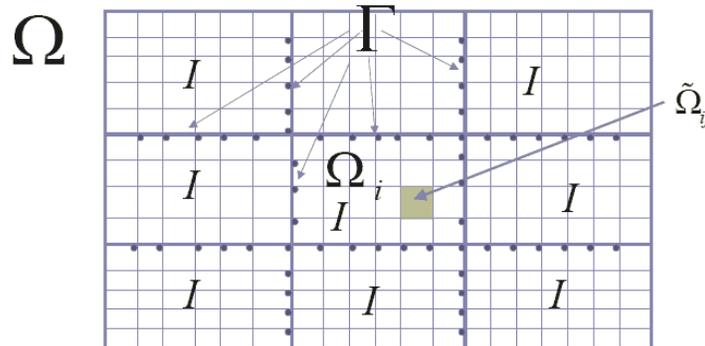


Figura 3: Dominio Ω descompuesto en una partición gruesa de 3×3 y cada subdominio Ω_i en una partición fina de 7×5 .

Sin pérdida de generalidad tomemos $g = 0$ en $\partial\Omega$, notemos que siempre es posible poner el problema de la Ec. (6.1) como uno con condiciones de frontera Dirichlet que se nulifiquen mediante la adecuada manipulación del término del lado derecho de la ecuación.

Denotemos por $V^h(\Omega_i)$ al espacio de funciones definidas por pedazos que son generadas por el método de elemento finito estándar las cuales se nulifican

en $\partial\Omega_i \cap \partial\Omega$.

La aproximación por elementos finitos de un problema elíptico que es continuo a través de Γ se denota por $V^h(\Omega)$.

Si consideramos el dominio Ω particionado en dos subdominios Ω_1 y Ω_2 , la formulación dada por la Ec. (6.1), es equivalente al siguiente problema acoplado

$$\begin{aligned} -\Delta u_1 &= f && \text{en } \Omega_1 \\ u_1 &= 0 && \text{en } \partial\Omega_1 \setminus \Gamma \\ u_1 &= u_2 && \text{en } \Gamma \\ \frac{\partial u_1}{\partial n_1} &= \frac{\partial u_2}{\partial n_2} && \text{en } \Gamma \\ -\Delta u_2 &= f && \text{en } \Omega_2 \\ u_2 &= 0 && \text{en } \partial\Omega_2 \setminus \Gamma \end{aligned} \quad (6.4)$$

aquí u_i es la restricción de u a Ω_i y n_i es el vector normal a Ω_i . La condición sobre la interfase Γ es llamada las condiciones de transmisibilidad y ellas son equivalentes a la igualdad de cualquier combinación lineal independiente de trazas de funciones y sus derivadas normales, a las derivadas normales en muchos ámbitos se les conoce también como flujo.

Considerando una triangulación del dominio y una aproximación de la Ec. (6.1) por el método de elemento finito ver sección (24.4). Tal aproximación da origen a un sistema lineal

$$\underline{\underline{A}}u = \underline{b} \quad (6.5)$$

donde la matriz $\underline{\underline{A}}$ es simétrica y positiva definida, la cual para una malla de diámetro h , típicamente tiene un número de condicionamiento sobre el orden de $1/h^2$. Aquí

$$\underline{\underline{A}} = \begin{bmatrix} \underline{\underline{A}}_{II}^1 & 0 & \underline{\underline{A}}_{I\Gamma}^1 \\ 0 & \underline{\underline{A}}_{II}^2 & \underline{\underline{A}}_{I\Gamma}^2 \\ \underline{\underline{A}}_{\Gamma I}^1 & \underline{\underline{A}}_{\Gamma I}^2 & \underline{\underline{A}}_{\Gamma\Gamma} \end{bmatrix}, \underline{u} = \begin{bmatrix} u_{I1} \\ u_{I2} \\ u_{\Gamma} \end{bmatrix} \text{ y } \underline{b} = \begin{bmatrix} b_{I1} \\ b_{I2} \\ b_{\Gamma} \end{bmatrix} \quad (6.6)$$

se ha particionado en los grados de libertad de los nodos interiores de Ω_1 y Ω_2 y los nodos de la frontera interior Γ . La matriz de rigidez $\underline{\underline{A}}$ y el vector de carga \underline{b} se obtienen por subensamble de las correspondientes contribuciones de ambos subdominios, i.e.

$$\underline{\underline{A}}_i = \begin{bmatrix} \underline{\underline{A}}_{II}^i & \underline{\underline{A}}_{I\Gamma}^i \\ \underline{\underline{A}}_{\Gamma I}^i & \underline{\underline{A}}_{\Gamma\Gamma} \end{bmatrix} \text{ y } \underline{b}_i = \begin{bmatrix} b_{Ii} \\ b_{\Gamma} \end{bmatrix} \text{ con } i = 1, 2 \quad (6.7)$$

son las matrices y vectores de rigidez para el problema de Poisson con condiciones de frontera Dirichlet sobre $\partial\Omega_i \setminus \Gamma$ y condiciones de Neumann sobre Γ , así tenemos que

$$\underline{A}_{\Gamma\Gamma} = \underline{A}_{\Gamma\Gamma}^1 + \underline{A}_{\Gamma\Gamma}^2 \text{ y } \underline{b}_{\Gamma_i} = \underline{b}_{\Gamma_1} + \underline{b}_{\Gamma_2}. \quad (6.8)$$

De la Ec. (6.4), se buscará una aproximación a las condiciones de transmisión, mediante una aproximación a las derivadas normales sobre Γ . Suponiendo la existencia de la solución exacta local \underline{u}_i , su derivada normal puede ser definida como una funcional lineal usando la fórmula de Green Ec. (167). Así, si ϕ_j es una base nodal de funciones para los nodos de Γ , se tiene de la Ec. (6.4) la siguiente expresión

$$\int_{\Gamma} \frac{\partial \underline{u}_i}{\partial n_i} \phi_j d\underline{s} = \int_{\Omega_i} (\Delta \underline{u}_i \phi_j + \nabla \underline{u}_i \cdot \nabla \phi_j) d\underline{x} = \int_{\Omega_i} (-\underline{f} \phi_j + \nabla \underline{u}_i \cdot \nabla \phi_j) d\underline{x}. \quad (6.9)$$

Una aproximación λ_i de la funcional representante de la derivada normal puede ser encontrada mediante el reemplazo de la solución exacta \underline{u}_i en el lado derecho de la ecuación anterior con la aproximación por elementos finitos. Corriendo j sobre todos los nodos de Γ y usando la definición de la matriz de carga local, se introduce la expresión

$$\lambda_i = \underline{A}_{\Gamma I}^i \underline{u}_{I_i} + \underline{A}_{\Gamma\Gamma}^i \underline{u}_{\Gamma_i} - \underline{b}_{\Gamma_i}. \quad (6.10)$$

Notemos que esta expresión coincide con el residual correspondiente a los nodos sobre Γ del problema de Poisson con condiciones de Neumann sobre Γ .

Usando la Ec. (6.10), se aproxima la Ec. (6.4) mediante

$$\begin{aligned} \underline{A}_{II}^1 \underline{u}_{I_1} + \underline{A}_{I\Gamma}^1 \underline{u}_{\Gamma_1} &= \underline{b}_{I_1} \\ \underline{u}_{\Gamma_1} &= \underline{u}_{\Gamma_2} = \underline{u}_{\Gamma} \\ \left(\underline{A}_{\Gamma I}^1 \underline{u}_{I_1} + \underline{A}_{\Gamma\Gamma}^1 \underline{u}_{\Gamma_1} - \underline{b}_{\Gamma_1} \right) &= - \left(\underline{A}_{\Gamma I}^2 \underline{u}_{I_2} + \underline{A}_{\Gamma\Gamma}^2 \underline{u}_{\Gamma_2} - \underline{b}_{\Gamma_2} \right) = \underline{\lambda}_{\Gamma} \\ \underline{A}_{II}^2 \underline{u}_{I_1} + \underline{A}_{I\Gamma}^2 \underline{u}_{\Gamma_2} &= \underline{b}_{I_2} \end{aligned} \quad (6.11)$$

nótese que la primera y última ecuación es la discretización del problema de Poisson para las funciones interiores \underline{u}_{I_i} con condiciones de frontera tipo Dirichlet que se nulifican sobre $\partial\Omega_i \setminus \Gamma$ y es igual al valor común \underline{u}_{Γ} sobre Γ . Alternativamente, la primera y tercera ecuación provee una discretización

del problema de Poisson en Ω_1 para la función local \underline{u}_1 con datos de frontera Neumann igual a $\underline{\lambda}_\Gamma$ y que se nulifique sobre $\partial\Omega_1 \setminus \Gamma$, de forma análoga se fórmula un problema con datos de frontera en Ω_2 con las ecuaciones tres y cuatro.

Es posible también obtener una ecuación para la traza de la solución exacta sobre Γ trabajando directamente con el problema continuo Ec. (6.10) el correspondiente operador es llamado Steklov-Poincaré. El complemento de Schur sección (19.7) es una aproximación de la ecuación de Steklov-Poincaré, determinado directamente por la aproximación mediante el método de elemento finito, particularmente, por la aproximación de la derivada normal Ec. (6.10).

6.1.1 Una Ecuación para el Flujo Usando el Complemento de Schur

Consideremos el sistema de ecuaciones lineales dado por la Ec. (6.5), donde la matriz $\underline{\underline{A}}$, \underline{u} y \underline{b} se definen como en Ec. (6.6). En el primer paso de la gran mayoría de los métodos iterativos de descomposición de dominio, los nodos interiores \underline{u}_{I_i} desconocidos en los subdominios son eliminados. Esto corresponde a un factorización por bloques de la matriz de la Ec. (6.6) en

$$\underline{\underline{A}} = \begin{bmatrix} \underline{\underline{I}} & 0 & 0 \\ 0 & \underline{\underline{I}} & 0 \\ \underline{\underline{A}}_{\Gamma I}^1 \left(\underline{\underline{A}}_{II}^1\right)^{-1} & \underline{\underline{A}}_{\Gamma I}^2 \left(\underline{\underline{A}}_{II}^2\right)^{-1} & \underline{\underline{I}} \end{bmatrix} \begin{bmatrix} \underline{\underline{A}}_{II}^1 & 0 & \underline{\underline{A}}_{I\Gamma}^1 \\ 0 & \underline{\underline{A}}_{II}^2 & \underline{\underline{A}}_{I\Gamma}^2 \\ 0 & 0 & \underline{\underline{S}} \end{bmatrix} \quad (6.12)$$

resultando el sistema lineal

$$\begin{pmatrix} \underline{\underline{A}}_{II}^1 & 0 & \underline{\underline{A}}_{I\Gamma}^1 \\ 0 & \underline{\underline{A}}_{II}^2 & \underline{\underline{A}}_{I\Gamma}^2 \\ 0 & 0 & \underline{\underline{S}} \end{pmatrix} \underline{u} = \begin{bmatrix} \underline{b}_{I_1} \\ \underline{b}_{I_2} \\ \underline{g}_\Gamma \end{bmatrix}. \quad (6.13)$$

Aquí $\underline{\underline{I}}$ es la matriz identidad y

$$\underline{\underline{S}} = \underline{\underline{A}}_{\Gamma\Gamma} - \underline{\underline{A}}_{\Gamma I}^1 \left(\underline{\underline{A}}_{II}^1\right)^{-1} \underline{\underline{A}}_{I\Gamma}^1 - \underline{\underline{A}}_{\Gamma I}^2 \left(\underline{\underline{A}}_{II}^2\right)^{-1} \underline{\underline{A}}_{I\Gamma}^2 \quad (6.14)$$

es la matriz del complemento de Schur véase sección (19.7) relativo a las incógnitas sobre Γ . También definimos el complemento de Schur local como

$$\underline{\underline{S}}^i = \underline{\underline{A}}_{\Gamma\Gamma}^i - \underline{\underline{A}}_{\Gamma I}^i \left(\underline{\underline{A}}_{II}^i\right)^{-1} \underline{\underline{A}}_{I\Gamma}^i \quad (6.15)$$

con $i = 1, 2$. Pudiendo encontrar el complemento de Schur para \underline{u}_Γ mediante el sistema

$$\underline{S}\underline{u}_\Gamma = \underline{g}_\Gamma \quad (6.16)$$

donde

$$\begin{aligned} \underline{S} &= \underline{S}^1 + \underline{S}^2 \\ \underline{g}_\Gamma &= \underline{g}_{\Gamma_1} + \underline{g}_{\Gamma_2} \\ \underline{g}_{\Gamma_1} &= \left(\underline{b}_{\Gamma_1} - \underline{A}_{\Gamma I}^1 \left(\underline{A}_{II}^1 \right)^{-1} \underline{b}_{I_1} \right) \\ \underline{g}_{\Gamma_2} &= \left(\underline{b}_{\Gamma_2} - \underline{A}_{\Gamma I}^2 \left(\underline{A}_{II}^2 \right)^{-1} \underline{b}_{I_2} \right) \end{aligned} \quad (6.17)$$

nótese que una vez encontrado \underline{u}_Γ mediante la resolución del sistema de la Ec. (6.16) los nodos interiores pueden ser encontrados usando

$$\underline{u}_{I_i} = \left(\underline{A}_{II}^i \right)^{-1} \left(\underline{b}_{I_i} - \underline{A}_{I\Gamma}^i \underline{u}_\Gamma \right). \quad (6.18)$$

Para derivar una ecuación para la derivada normal $\underline{\lambda}_\Gamma$ sobre Γ se usa los valores $\underline{\lambda}_\Gamma = \underline{\lambda}_{\Gamma_1} = -\underline{\lambda}_{\Gamma_2}$ de frontera interior que son desconocidos tercera ecuación de la Ec. (6.11) y resolver el sistema local de problemas de Neumann para encontrar \underline{u}_1 y \underline{u}_2 , i.e.

$$\begin{bmatrix} \underline{A}_{II}^i & \underline{A}_{I\Gamma}^i \\ \underline{A}_{\Gamma I}^i & \underline{A}_{\Gamma\Gamma}^i \end{bmatrix} \begin{bmatrix} \underline{u}_{I_i} \\ \underline{u}_{\Gamma_i} \end{bmatrix} = \begin{bmatrix} \underline{b}_{I_i} \\ \underline{b}_{\Gamma_i} + \underline{\lambda}_{\Gamma_i} \end{bmatrix}, \quad \text{con } i = 1, 2 \quad (6.19)$$

usando factorización por bloques de las matrices locales, se tiene

$$\underline{u}_i^\Gamma = \left(\underline{S}^i \right)^{-1} \left(\underline{g}_{\Gamma_i} + \underline{\lambda}_{\Gamma_i} \right) \quad (6.20)$$

donde \underline{g}_{Γ_i} es dado como en la Ec. (6.17). Usando la segunda ecuación del problema de la Ec. (6.11) la cual crea \underline{u}_{Γ_1} y \underline{u}_{Γ_2} iguales, encontramos la ecuación para el flujo, y está dada por

$$\underline{F}\underline{\lambda}_\Gamma = \underline{d}_\Gamma \quad (6.21)$$

con

$$\begin{aligned} \underline{F} &= \left(\underline{S}^1 \right)^{-1} + \left(\underline{S}^2 \right)^{-1} \\ \underline{d}_\Gamma &= \underline{d}_{\Gamma_1} + \underline{d}_{\Gamma_2} \\ \underline{d}_{\Gamma_1} &= \left(\underline{S}^1 \right)^{-1} \underline{g}_{\Gamma_1} \\ \underline{d}_{\Gamma_2} &= \left(\underline{S}^2 \right)^{-1} \underline{g}_{\Gamma_2}. \end{aligned} \quad (6.22)$$

6.1.2 Extensión Armónica Discreta

El espacio de funciones armónicas discretas es un subespacio importante, relacionado directamente con el complemento de Schur y de los valores de los nodos sobre Γ .

Denotaremos por $\mathcal{H}_i(\underline{u}_\Gamma)$ a la proyección de los nodos $\underline{u}_\Gamma \in \Gamma$ a los nodos interiores \underline{u}_{I_i} pertenecientes al subdominio Ω_i .

Definición 50 Una función \underline{u}_i definida sobre Ω_i es llamada armónica discreta sobre Ω_i si

$$\underline{A}_{II}^i \underline{u}_{I_i} + \underline{A}_{I\Gamma}^i \underline{u}_\Gamma = 0. \quad (6.23)$$

Notemos que $\underline{u}_i = \mathcal{H}_i(\underline{u}_\Gamma)$ es completamente definido por sus valores sobre $\partial\Omega_i \cap \Gamma$ y es ortogonal -en el producto interior $a(\cdot, \cdot)$ - al espacio $V^h \cap H_0^1(\Omega_i)$.

El espacio $\widetilde{V}^h \subset V^h(\Omega)$ global de funciones armónicas discretas consistentes de funciones armónicas discretas sobre cada subdominio Ω_i . Una función \underline{u} pertenece a \widetilde{V}^h si y sólo si

$$\underline{A}_{II}^i \underline{u}_I + \underline{A}_{I\Gamma}^i \underline{u}_\Gamma = 0 \quad (6.24)$$

y es completamente definida por sus valores sobre Γ . El espacio \widetilde{V}^h es ortogonal -en el producto interior $a(\cdot, \cdot)$ - a todos los espacios interiores

$$V^h \cap H_0^1(\Omega_i), \text{ con } i = 1, \dots, E. \quad (6.25)$$

Denotaremos la extensión armónica discreta por pedazos de \underline{u}_Γ por $\mathcal{H}(\underline{u}_\Gamma)$.

6.2 One-Level FETI

6.2.1 Algoritmos en Dos Subdominios

En esta subsección se mostrarán dos algoritmos, uno corresponde al Neumann-Neumann y otro al Dirichlet-Dirichlet, este último conocido como FETI.

El Algoritmo Neumann-Neumann Si consideramos el dominio Ω particionado en dos subdominios Ω_1 y Ω_2 , la formulación dada por la Ec. (6.1), puede ser expresada como un problema iterativo en el se inicia con un valor

supuesto \underline{u}_{Γ_0} el primer paso en este algoritmo consiste en resolver un problema con condiciones de Frontera Dirichlet en cada Ω_i con datos \underline{u}_{Γ_0} sobre Γ , para después resolver un problema en cada subdominio con condiciones de frontera Neumann sobre Γ eligiendo como la derivada normal la diferencia de la solución de los dos problemas con condiciones de frontera Dirichlet. Los valores sobre Γ de la solución de dichos problemas Neumann es empleado para corregir el valor inicial \underline{u}_{Γ_0} y encontrar el nuevo valor de \underline{u}_{Γ_1} , y así continuar de manera iterativa para $n \geq 0$, y el algoritmo queda expresado como

$$(D_i) \left\{ \begin{array}{l} -\Delta \underline{u}_i^{n+1/2} = f \quad , \text{ en } \Omega_i \\ \underline{u}_i^{n+1/2} = 0 \quad , \text{ en } \partial\Omega_i \setminus \Gamma \\ \underline{u}_i^{n+1/2} = \underline{u}_{\Gamma}^n \quad , \text{ en } \Gamma \end{array} \right\}, \text{ con } i = 1, 2 \quad (6.26)$$

$$(N_i) \left\{ \begin{array}{l} -\Delta \underline{v}_i^{n+1} = 0 \quad , \text{ en } \Omega_i \\ \underline{v}_i^{n+1} = 0 \quad , \text{ en } \partial\Omega_i \setminus \Gamma \\ \frac{\partial \underline{v}_i^{n+1}}{\partial \underline{n}_i} = \frac{\partial \underline{u}_1^{n+1/2}}{\partial \underline{n}_1} + \frac{\partial \underline{u}_2^{n+1/2}}{\partial \underline{n}_2} \quad , \text{ en } \Gamma \end{array} \right\}, \text{ con } i = 1, 2$$

$$\underline{u}_{\Gamma}^{n+1} = \underline{u}_{\Gamma}^n - \theta (\underline{v}_1^{n+1} + \underline{v}_2^{n+1}) \quad \text{en } \Gamma$$

para algún adecuado $\theta \in (0, \theta_{m\acute{a}x})$. Usando una aproximación a la derivada normal, se derivará un método iterativo para el problema discreto. Definiendo los vectores $\underline{w}_i = \underline{u}_{I_i}$ y $\underline{r}_i = \underline{v}_{I_i}$ entonces se tiene que

$$(D_i) \left\{ \underline{A}_{II}^i \underline{w}_i^{n+1/2} + \underline{A}_{I\Gamma}^i \underline{u}_{\Gamma}^n = \underline{b}_{I_i}, \text{ con } i = 1, 2 \right\} \quad (6.27)$$

$$(N_i) \left\{ \left[\begin{array}{cc} \underline{A}_{II}^i & \underline{A}_{I\Gamma}^i \\ \underline{A}_{\Gamma I}^i & \underline{A}_{\Gamma\Gamma}^i \end{array} \right] \left[\begin{array}{c} \underline{w}_i^{n+1} \\ \underline{r}_i^{n+1} \end{array} \right] = \left[\begin{array}{c} 0 \\ \underline{t}_{\Gamma} \end{array} \right] \right\}, \text{ con } i = 1, 2$$

$$\underline{u}_{\Gamma}^{n+1} = \underline{u}_{\Gamma}^n - \theta (\underline{\eta}_1^{n+1} + \underline{\eta}_2^{n+1})$$

donde el residual \underline{t}_{Γ} es definido como

$$\underline{t}_{\Gamma} = \left(\underline{A}_{\Gamma I}^1 \underline{w}_1^{n+1/2} + \underline{A}_{\Gamma\Gamma}^1 \underline{u}_{\Gamma}^n - \underline{b}_{\Gamma_1} \right) + \left(\underline{A}_{\Gamma I}^2 \underline{w}_2^{n+1/2} + \underline{A}_{\Gamma\Gamma}^2 \underline{u}_{\Gamma}^n - \underline{b}_{\Gamma_2} \right) \quad (6.28)$$

en vista de la tercera ecuación de la Ec. (6.11).

Eliminando $\underline{w}_i^{n+1/2}, \underline{r}_i^{n+1/2}$ de la Ec. (6.27) entonces (D_i) queda dado por

$$\underline{t}_{\Gamma} = - \left(\underline{g}_{\Gamma} - \underline{S} \underline{u}_{\Gamma}^n \right), \quad (6.29)$$

el cual muestra que la diferencia \underline{t}_Γ del flujo local es igual a menos el residual del sistema del complemento de Schur. Usando factorización por bloques de las matrices locales \underline{A}_i , los problemas (N_i) , quedan en términos de

$$\underline{\eta}_i^{n+1} = (\underline{S}^i)^{-1} \underline{t}_\Gamma = - (\underline{S}^i)^{-1} (\underline{g}_\Gamma - \underline{S}u_\Gamma^n). \quad (6.30)$$

Por lo tanto, encontramos

$$\underline{u}_\Gamma^{n+1} - \underline{u}_\Gamma^n = \theta \left((\underline{S}^1)^{-1} + (\underline{S}^2)^{-1} \right) (\underline{g}_\Gamma - \underline{S}u_\Gamma^n) \quad (6.31)$$

lo cual muestra que el algoritmo Neumann-Neumann es también un sistema iterativo preconditionado de Richardson para el sistema del complemento de Schur, con preconditionador $(\underline{S}^1)^{-1} + (\underline{S}^2)^{-1}$. La matriz preconditionada es

$$\underline{FS} = \left((\underline{S}^1)^{-1} + (\underline{S}^2)^{-1} \right) \underline{S} = \left((\underline{S}^1)^{-1} + (\underline{S}^2)^{-1} \right) (\underline{S}^1 + \underline{S}^2). \quad (6.32)$$

La aplicación de este algoritmo implica la solución de dos problemas con condición de frontera tipo Dirichlet y dos problemas con condición de frontera tipo Neumann con datos sobre Γ .

El Algoritmo Dirichlet-Dirichlet Consideremos aquí el dual del algoritmo Neumann-Neumann este se le conoce como Finite Element Tearing and Interconnecting (FETI) o algoritmo Dirichlet-Dirichlet. Si consideramos el dominio Ω particionado en dos subdominios Ω_1 y Ω_2 , la formulación dada por la Ec. (6.1), puede ser expresada como un problema iterativo en el se inicia con un valor supuesto $\underline{\lambda}_{\Gamma_0}$ del flujo sobre Γ ver Ec. (6.21) - aquí $\underline{\lambda}_\Gamma$ es una aproximación a la derivada normal en la dirección \underline{n}_i^- , el primer paso en este algoritmo consiste en resolver un problema con condiciones de Frontera Neumann en cada Ω_i con datos $\underline{\lambda}_{\Gamma_0}$ sobre Γ , para después resolver un problema en cada subdominio con condiciones de frontera Dirichlet sobre Γ eligiendo como la derivada normal la diferencia de la solución de los dos problemas con condiciones de frontera Neumann. Los valores sobre Γ de la solución de dichos problemas Neumann es empleado para corregir el valor inicial $\underline{\lambda}_{\Gamma_0}$ y encontrar el nuevo valor de $\underline{\lambda}_{\Gamma_1}$, y así continuar de manera iterativa para

$n \geq 0$, y el algoritmo queda expresado como

$$(N_i) \left\{ \begin{array}{l} -\Delta \underline{u}_i^{n+1/2} = f \quad , \text{ en } \Omega_i \\ \underline{u}_i^{n+1/2} = 0 \quad , \text{ en } \partial\Omega_i \setminus \Gamma \\ \frac{\partial \underline{u}_i^{n+1/2}}{\partial n_i} = \underline{\lambda}_{\Gamma_i}^n \quad , \text{ en } \Gamma \end{array} \right\}, \text{ con } i = 1, 2 \quad (6.33)$$

$$(D_i) \left\{ \begin{array}{l} -\Delta \underline{v}_i^{n+1} = 0 \quad , \text{ en } \Omega_i \\ \underline{v}_i^{n+1} = 0 \quad , \text{ en } \partial\Omega_i \setminus \Gamma \\ \underline{v}^{n+1} = \underline{u}_1^{n+1/2} - \underline{u}_2^{n+1/2} \quad , \text{ en } \Gamma \end{array} \right\}, \text{ con } i = 1, 2$$

$$\underline{\lambda}_{\Gamma}^{n+1} = \underline{\lambda}_{\Gamma}^n - \theta \left(\frac{\partial \underline{v}_1^{n+1}}{\partial n_1} + \frac{\partial \underline{v}_2^{n+1}}{\partial n_2} \right) \quad \text{en } \Gamma$$

para algún adecuado $\theta \in (0, \theta_{m\acute{a}x})$. Usando una aproximación a la derivada normal, se derivará un método iterativo para el problema discreto. Definiendo los vectores $\underline{w}_i = \underline{u}_{I_i}$ y $\underline{r}_i = \underline{v}_{I_i}$ entonces se tiene que

$$(N_i) \left\{ \left[\begin{array}{cc} \underline{A}_{II}^i & \underline{A}_{I\Gamma}^i \\ \underline{A}_{\Gamma I}^i & \underline{A}_{\Gamma\Gamma}^i \end{array} \right] \left[\begin{array}{c} \underline{w}_i^{n+1} \\ \underline{r}_i^{n+1} \end{array} \right] = \left[\begin{array}{c} \underline{b}_{I_i} \\ \underline{b}_{\Gamma_i} + \underline{\lambda}_{\Gamma_i}^n \end{array} \right] \right\}, \text{ con } i = 1, 2 \quad (6.34)$$

$$(D_i) \left\{ \underline{A}_{II}^i \underline{r}_i^{n+1/2} + \underline{A}_{I\Gamma}^i \underline{t}_{\Gamma}^n = 0, \text{ con } i = 1, 2 \right\}$$

$$\underline{\lambda}_{\Gamma_i}^{n+1} = \underline{\lambda}_{\Gamma_i}^n - \theta \left(\underline{\eta}_1^{n+1} + \underline{\eta}_2^{n+1} \right)$$

donde el residual \underline{t}_{Γ} es definido como

$$\underline{t}_{\Gamma} = \underline{\gamma}_1^{n+1} - \underline{\gamma}_2^{n+1} \quad (6.35)$$

y el flujo $\underline{\eta}_i^{n+1}$ por

$$\underline{\eta}_i^{n+1} = \underline{A}_{I\Gamma}^i \underline{r}_i^{n+1/2} + \underline{A}_{\Gamma\Gamma}^i \underline{t}_{\Gamma} \quad (6.36)$$

conforme a la Ec (6.10).

Eliminando $\underline{w}_i^{n+1/2}$, $\underline{\gamma}_i^{n+1/2}$ y $\underline{r}_i^{n+1/2}$ de la Ec. (6.34), usando factorización por bloques de las matrices locales \underline{A}^i , los problemas (N_i) , quedan en términos de

$$\underline{t}_{\Gamma} = - \left(\underline{d}_{\Gamma} - \underline{F} \underline{\lambda}_{\Gamma}^n \right), \quad (6.37)$$

el cual muestra que la diferencia \underline{t}_{Γ} del flujo local es igual a menos el residual del sistema de la Ec. (6.21). Los problemas (D_i) quedan en términos de

$$\underline{\eta}_i^{n+1} = \underline{S}^i \underline{t}_{\Gamma} = - \underline{S}^i \left(\underline{d}_{\Gamma} - \underline{F} \underline{\lambda}_{\Gamma}^n \right). \quad (6.38)$$

Por lo tanto, encontramos

$$\underline{\lambda}_\Gamma^{n+1} - \underline{\lambda}_\Gamma^n = \theta \left(\underline{S}_1 + \underline{S}_2 \right) \left(\underline{d}_\Gamma - \underline{F}\underline{\lambda}_\Gamma^n \right) \quad (6.39)$$

lo cual muestra que el algoritmo Dirichlet-Dirichlet es también un sistema iterativo preconditionado de Richardson para el sistema del complemento de Schur, con preconditionador $\underline{S}_1 + \underline{S}_2$. La matriz preconditionada es

$$\underline{SF} = \underline{S} \left((\underline{S}^1)^{-1} + (\underline{S}^2)^{-1} \right) = (\underline{S}^1 + \underline{S}^2) \left((\underline{S}^1)^{-1} + (\underline{S}^2)^{-1} \right) \quad (6.40)$$

la aplicación de este algoritmo implica la solución de dos problemas con condición de frontera tipo Neumann y dos problemas con condición de frontera tipo Dirichlet con datos sobre Γ .

6.2.2 Algoritmos en Múltiples Subdominios

Sea $\Omega \subset \mathbb{R}^n$ un dominio, y $\Pi = \{\Omega_1, \dots, \Omega_E\}$ una partición o descomposición en subdominios del dominio Ω , i.e. se asume que:

- 1.- Ω_α , para $\alpha = 1, \dots, E$ es un subdominio de Ω ,
- 2.- $\Omega_\alpha \cap \Omega_\beta = \emptyset$, siempre que $\alpha \neq \beta$.
- 3.- $\Omega \subset \bigcup_{\alpha=1}^E \overline{\Omega_\alpha}$.

La notación $\partial\Omega$ y $\partial\Omega_\alpha$, $\alpha = 1, \dots, E$ es tomada de la frontera del dominio Ω y la frontera del subdominio Ω_α respectivamente, claramente

$$\partial\Omega \subset \bigcup_{\alpha=1}^E \partial\Omega_\alpha. \quad (6.41)$$

Adicionalmente definimos $\Gamma_i = \partial\Omega_i \cap \partial\Omega$, a la frontera interior como

$$\Gamma = \bigcup_i \Gamma_i. \quad (6.42)$$

Notemos que Γ y Γ_i son conjuntos abiertos. Un problema acoplado como en la Ec. (6.4) puede ser resuelto hallando las condiciones de transmisión

impuestas a lo largo de cada borde $\partial\Omega_i \cap \partial\Omega_j$. El sistema lineal dado por la Ec. (6.5) puede ser escrito como

$$\begin{bmatrix} \underline{\underline{A}}_{II} & \underline{\underline{A}}_{I\Gamma} \\ \underline{\underline{A}}_{\Gamma I} & \underline{\underline{A}}_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} \underline{u}_I \\ \underline{u}_\Gamma \end{bmatrix} = \begin{bmatrix} \underline{b}_I \\ \underline{b}_\Gamma \end{bmatrix} \quad (6.43)$$

el cuál ha sido particionado en los grados de libertad de los nodos interiores de los subdominios y los nodos de frontera sobre Γ . La matriz de carga y el lado derecho fueron obtenidos por subensamble de los correspondientes componentes relativos a los subdominios según la Ec. (6.7).

Las incógnitas en el interior de los subdominios \underline{u}_I pueden ser eliminadas por eliminación Gaussiana en bloques y el sistema lineal resultante es

$$\begin{bmatrix} \underline{\underline{A}}_{II} & \underline{\underline{A}}_{I\Gamma} \\ 0 & \underline{\underline{S}} \end{bmatrix} \begin{bmatrix} \underline{u}_I \\ \underline{u}_\Gamma \end{bmatrix} = \begin{bmatrix} \underline{b}_I \\ \underline{g}_\Gamma \end{bmatrix}. \quad (6.44)$$

Como antes, el complemento de Schur y el vector \underline{g}_Γ puede ser encontrado por subensamble de las contribuciones locales, ver sección (19.7). Primero definimos una familia de operadores de restricción, dado un vector de la cantidad de grados de libertad que el vector \underline{u}_Γ sobre la interfase Γ , definimos la restricción $\underline{\underline{R}}^i(\underline{u}_\Gamma)$ como el vector de grados de libertad de \underline{u}_Γ sobre Γ_i . Aquí $\underline{\underline{R}}^i$ es una matriz rectangular de ceros y unos. Y para cada subdominio Ω_i los grados de libertad de los nodos de la frontera interior Γ_i de Ω_i como en la Ec. (6.7), teniendo

$$\begin{aligned} \underline{\underline{S}} &= \sum_{i=1}^E (\underline{\underline{R}}^i)^T \underline{\underline{S}}^i \underline{\underline{R}}^i \\ \underline{g}_\Gamma &= \sum_{i=1}^E (\underline{\underline{R}}^i)^T \left(\underline{b}_{\Gamma_i} - \underline{\underline{A}}_{\Gamma I}^i \left(\underline{\underline{A}}_{II}^i \right)^{-1} \underline{b}_{I_i} \right) \end{aligned} \quad (6.45)$$

donde el complemento de Schur local es definido como en la Ec. (6.15) y $(\underline{\underline{R}}^i)^T$ es el transpuesto de $\underline{\underline{R}}^i$.

El Algoritmo Neumann-Neumann Examinando el método Neumann-Neumann para dos subdominios, en particular la Ec. (6.32), entonces la generalización al caso de múltiples subdominios queda dada en términos de

$$\underline{\underline{S}}_{NN} = \sum_{i=1}^E (\underline{\underline{R}}^i)^T (\underline{\underline{S}}^i)^{-1} \underline{\underline{R}}^i \underline{\underline{S}}. \quad (6.46)$$

Notemos que la aplicación de este operador a un vector implica la solución en cada subdominio Ω_i , de un problema con condiciones de frontera Dirichlet y un problema con condiciones de frontera Neumann sobre $\partial\Omega_i \cap \Gamma$. Así también, todos los subdominios que no tocan $\partial\Omega$, $\underline{\underline{S}}^i$ es no singular y $(\underline{\underline{S}}^i)^{-1}$ puede entenderse como una pseudo inversa o una inversa de un problema regularizado.

Adicionalmente, el algoritmo Neumann-Neumann también puede ser definido a nivel continuo usando la Ec. (6.26), con $i = 1, \dots, E$ con condiciones de frontera Neumann para el problema N_i sobre cada cara $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j$. La nueva iterada $\underline{u}_\Gamma^{n+1}$ en los nodos de la interfase, se construye por la corrección en todos los subdominios que tienen nodos sobre esas fronteras.

El Algoritmo Dirichlet-Dirichlet Aquí trataremos la versión más general del algoritmo Dirichlet-Dirichlet mejor conocido cómo método One-Level FETI.

Primeramente consideremos a $\Omega \subset \mathbb{R}^n$ un dominio, y $\Pi = \{\Omega_1, \dots, \Omega_E\}$ una partición o descomposición en subdominios del dominio Ω , además sea Γ_i la frontera de interior del subdominio Ω_i y Γ la frontera interior del dominio Ω .

Asumiremos que las discontinuidades en la ecuación diferencial parcial -si existen- estarán alineadas con las fronteras de los subdominios, tal que en cada subdominio Ω_i , el coeficiente $\rho(x)$ de la ecuación tenga un valor constante, sin pérdida de generalidad se asumirá $\rho_i > 0$. Además, denotaremos a W_i como el espacio de trazas de Ω_i , es decir

$$W_i = W^h(\partial\Omega_i \cap \Gamma), \text{ con } i = 1, \dots, E \quad (6.47)$$

también denotaremos W como el espacio producto del espacio de las trazas, es decir

$$W = \prod_{i=1}^E W_i \quad (6.48)$$

y la extensión armónica discreta Sec(6.1.2) por pedazos de \underline{u}_Γ por $\mathcal{H}(\underline{u}_\Gamma)$.

Así, en lo que resta de esta sección, se trabajara casi exclusivamente con funciones en el espacio de trazas W_i y cuando sea conveniente, se considerarán como un elemento representante de las funciones armónicas discretas en Ω_i , de tal forma que $\underline{w} \in W$, $\mathcal{H}(\underline{w})$ denotará la extensión por pedazos de la armónica discreta sobre todo el subdominio Ω_i , entenderemos $\mathcal{H}(\underline{w})$ como un elemento en el espacio producto W con componentes $\mathcal{H}_i(\underline{w}_i)$.

Reformulando el problema definido por la Ec. (6.1) a uno reducido a la interfase Γ por medio de elementos finitos, como un problema de minimización con restricciones impuestas por lo requerimientos de continuidad en Γ queda como:

Encontrar $\underline{u} \in W$ tal que

$$J(\underline{u}) = \frac{1}{2} \left\{ \begin{array}{l} \langle \underline{S}\underline{u}, \underline{u} \rangle - \langle \underline{f}, \underline{u} \rangle \rightarrow \min \\ \underline{B}\underline{u} = 0 \end{array} \right\} \quad (6.49)$$

donde la matriz por bloques \underline{S} es formada por las matrices \underline{S}_i Ec. (6.15) del complemento de Schur en el i -ésimo subdominio, el vector por bloques \underline{u} es formado por los vectores \underline{u}_i solución de la frontera interior en cada i -ésimo subdominio y el vector \underline{f} es formado por los vectores \underline{f}_i de la frontera interior en cada i -ésimo subdominio, i.e.

$$\underline{S} = \begin{bmatrix} \underline{S}^1 & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & \cdots & \underline{S}^E \end{bmatrix}, \underline{u} = \begin{bmatrix} \underline{u}_1 \\ \vdots \\ \vdots \\ \underline{u}_E \end{bmatrix}, \underline{f} = \begin{bmatrix} \underline{f}_1 \\ \vdots \\ \vdots \\ \underline{f}_E \end{bmatrix} \quad (6.50)$$

y la matriz \underline{B} es formada por las matrices \underline{B}_i en cada i -ésimo subdominio tal que la solución asociada a más de un subdominio coincida, i.e.

$$\underline{B} = [\underline{B}^1, \quad \cdots \quad \cdots, \quad \underline{B}^E] \quad (6.51)$$

es construida de $\{0, 1, -1\}$ tal que los valores de la solución \underline{u} asociada a más de un subdominio coincida cuando $\underline{B}\underline{u} = 0$, donde la elección de \underline{B} no es única.

Nótese que un mismo nodo en la frontera interior pertenece a dos o más subdominios, por ello es necesario algún mecanismo para asegurar que la solución asociada a más de un subdominio coincida.

El problema (6.49) es soluble de manera única ya que el

$$Kernel(\underline{S}) \cap Kernel(\underline{B}) = \{0\} \quad (6.52)$$

lo cual indica que \underline{S} es invertible sobre el $Kernel(\underline{B})$.

Pero introduciendo un vector de multiplicadores de Lagrange $\underline{\lambda}$ para imponer las restricciones $\underline{B}\underline{u} = 0$, obtenemos una formulación silla de la Ec. (6.49):

Encontrar $(\underline{u}, \underline{\lambda}) \in W \times U$ tal que

$$\begin{cases} \underline{S}\underline{u} + \underline{B}^T \underline{\lambda} = \underline{f} \\ \underline{B}\underline{u} = \underline{0} \end{cases} \quad (6.53)$$

la solución $\underline{\lambda}$ de la Ec. (6.53) es única salvo la adición de un elemento del $Kernel(\underline{B}^T)$. El espacio de multiplicadores de Lagrange U , es por lo tanto elegido como el $Rango(\underline{B})$. Este espacio puede ser entendido como el espacio de las funciones de salto en W .

Definición 51 *Decimos que un subdominio Ω_i es un subdominio flotante si la intersección de este con la frontera del dominio $\partial\Omega$ es vacía.*

También usamos a la matriz \underline{R} construida de todos los espacios nulos de los elementos de \underline{S} , cuyos elementos están asociados cada subdominio de manera individual

$$\underline{R} = \begin{bmatrix} \underline{R}^1 & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & \cdots & \underline{R}^E \end{bmatrix} \quad (6.54)$$

tal que el $Rango(\underline{R}) = Kernel(\underline{S})$. De hecho, sólo los subdominio flotantes contribuyen, i.e. el subdominio que intersecta a $\partial\Omega$ no contribuye al $Kernel(\underline{S})$, y por tanto esas columnas de \underline{R} son nulas.

Una solución \underline{u} a la primera ecuación de la Ec. (6.53) existe si y sólo si $\underline{f} - \underline{B}^T \underline{\lambda} \in Rango(\underline{S})$, esta restricción permite la introducción de un operador proyección P . Obteniendo

$$\underline{u} = \underline{S}^\dagger (\underline{f} - \underline{B}^T \underline{\lambda}) - \underline{R}\underline{\alpha} \text{ si } (\underline{f} - \underline{B}^T \underline{\lambda}) \perp Kernel(\underline{S}) \quad (6.55)$$

donde \underline{S}^\dagger es una pseudo-inversa de \underline{S} , notemos que $\underline{\alpha}$ puede ser fácilmente determinada una vez encontrada $\underline{\lambda}$.

Sustituyendo la expresión para \underline{u} dentro de la segunda ecuación de la Ec. (6.53) obtenemos

$$\underline{B}\underline{S}^\dagger \underline{B}^T \underline{\lambda} = \underline{B}\underline{S}^\dagger \underline{f} - \underline{B}\underline{R}\underline{\alpha} \quad (6.56)$$

así, se obtiene el sistema

$$\begin{cases} \underline{B}\underline{S}^\dagger \underline{B}^T \underline{\lambda} + \underline{B}\underline{R}\underline{\alpha} = \underline{B}\underline{S}^\dagger \underline{f} \\ \underline{B}\underline{R}^T \underline{\lambda} = \underline{R}^T \underline{f} \end{cases} \quad (6.57)$$

donde la primera ecuación se obtiene de la segunda ecuación de Ec. (6.53) al sustituir \underline{u} de la Ec. (6.55) y la segunda se obtiene de la primera Ec. (6.53) al sustituir también \underline{u} de la Ec. (6.55) y usando el hecho que $(\underline{f} - \underline{B}^T \underline{\lambda}) \perp \text{Kernel}(\underline{S})$. Esta última ecuación puede escribirse más compactamente como

$$\begin{cases} \underline{F}\underline{\lambda} + \underline{G}\underline{\alpha} = \underline{d} \\ \underline{G}^T \underline{\lambda} = \underline{e} \end{cases} \quad (6.58)$$

donde $\underline{F} = \underline{B}\underline{S}^\dagger \underline{B}^T$, $\underline{G} = \underline{B}\underline{R}$, $\underline{d} = \underline{B}\underline{S}^\dagger \underline{f}$ y $\underline{e} = \underline{R}^T \underline{f}$.

Introduciendo una matriz simétrica y positiva definida \underline{Q} y un producto interior $\langle \underline{\lambda}, \underline{\mu} \rangle = \langle \underline{\lambda}, \underline{Q}\underline{\mu} \rangle$ sobre $U = \text{Rango}(\underline{B})$, como antes $\langle \cdot, \cdot \rangle$ es el producto estándar del espacio L^2 . Sea

$$\underline{P}^T = \underline{I} - \underline{G} \left(\underline{G}^T \underline{Q} \underline{G} \right)^{-1} \underline{G}^T \underline{Q} \quad (6.59)$$

la proyección de U sobre el subespacio de multiplicadores de Lagrange que son Q -ortogonales al $\text{Rango}(\underline{G})$; también definimos

$$\underline{P} = \underline{I} - \underline{Q} \underline{G} \left(\underline{G}^T \underline{Q} \underline{G} \right)^{-1} \underline{G}^T \quad (6.60)$$

como la proyección de U sobre el $\text{Kernel}(\underline{G}^T)$, esta proyección es ortogonal en el Q^{-1} producto interior, es decir, el producto interior definido por $\langle \underline{\lambda}, \underline{Q}^{-1} \underline{\mu} \rangle$.

Notemos que si $\underline{Q} = \underline{I}$, entonces $\underline{P}^T = \underline{I} - \underline{B}\underline{R} \left((\underline{B}\underline{R})^T \underline{I} (\underline{B}\underline{R}) \right)^{-1} (\underline{B}\underline{R})^T \underline{I}$, desarrollando

$$\begin{aligned} \underline{P}^T &= \underline{I} - \underline{B}\underline{R} (\underline{B}\underline{R})^{-T} (\underline{B}\underline{R})^{-1} (\underline{B}\underline{R})^T \\ &= \underline{I} - \underline{B}\underline{R}\underline{R}^{-1} \underline{B}^{-1} \underline{B}^{-1} \underline{R}^{-1} \underline{R}\underline{B} \\ &= \underline{I} - \underline{I}_{\text{Kernel}(S)} \end{aligned} \quad (6.61)$$

así, $\underline{P}^T \underline{v} = \underline{v} - \text{Proy}_{\text{Kernel}(S)}(\underline{v})$.

Aplicando \underline{P}^T al sistema dado por la Ec. (6.58) obtenemos

$$\begin{cases} \underline{P}^T \underline{F}\underline{\lambda} = \underline{P}^T \underline{d} \\ \underline{P}^T \underline{G}^T \underline{\lambda} = \underline{e} \end{cases} \quad (6.62)$$

ya que $P^T \underline{G}\alpha = \underline{B}R\alpha - \text{Proy}_{\text{Kernel}(S)}(\underline{B}R\alpha) = 0$.

Multiplicando la Ec. (6.56) por $(\underline{G}^T \underline{Q} \underline{G})^{-1} \underline{G}^T \underline{Q}$ encontramos que

$$\alpha = \left(\underline{G}^T \underline{Q} \underline{G} \right)^{-1} \underline{G}^T \underline{Q} (\underline{d} - \underline{F}\lambda) \quad (6.63)$$

el cual queda totalmente determinado por los valores de λ . Notemos que los operadores P y P^T representa solamente la parte global del preconditionador.

Introduciendo ahora los subespacios

$$\begin{aligned} V &= \{ \lambda \in U \mid \langle \lambda, \underline{B}z \rangle = 0, \text{ con } z \in \text{Kernel}(\underline{S}) \} \\ &= \text{Kernel}(\underline{G}^T) = \text{Rango}(\underline{P}) \end{aligned} \quad (6.64)$$

y

$$\begin{aligned} V' &= \{ \mu \in U \mid \langle \mu, \underline{B}z \rangle_Q = 0, \text{ con } z \in \text{Kernel}(\underline{S}) \} \\ &= \text{Rango}(\underline{P}^T) \end{aligned} \quad (6.65)$$

donde el espacio V' es isomorfo al dual del espacio V .

El método One-Level FETI es un método de Gradiente Conjugado preconditionado en el espacio V , aplicado a

$$\underline{P}^T \underline{F}\lambda = \underline{P}^T \underline{d}, \lambda \in \lambda_0 + V \quad (6.66)$$

con condición inicial aproximada λ_0 escogido tal que $\underline{G}\lambda_0 = \underline{e}$.

El preconditionador más básico para FETI al tomar $\underline{Q} = \underline{I}$, es de la forma

$$\underline{M}^{-1} = \underline{B} \underline{S} \underline{B}^T = \sum_{i=1}^E \underline{B}^i \underline{S}^i (\underline{B}^i)^T \quad (6.67)$$

otra variante es

$$\underline{P} \underline{M}^{-1} \underline{P}^T \underline{F}\lambda = \underline{P} \underline{M}^{-1} \underline{P}^T \underline{d}, \lambda \in \lambda_0 + V \quad (6.68)$$

notemos que en esta variante, para $\lambda \in V$, $\underline{P} \underline{M}^{-1} \underline{P}^T \underline{F}\lambda = \underline{P} \underline{M}^{-1} \underline{P}^T \underline{P}^T \underline{F}\lambda$, y esto puede ser visto como el producto de dos matrices simétricas, nótese

que estrictamente $\underline{\underline{M}}^{-1}$ no tiene una inversa por eso usaremos $\hat{\underline{\underline{M}}}^{-1}$ que si la tiene y es definido como

$$\begin{aligned}\hat{\underline{\underline{M}}}^{-1} &= (\underline{\underline{BD}}^{-1} \underline{\underline{B}}^T)^{-1} \underline{\underline{BD}}^{-1} \underline{\underline{SD}}^{-1} \underline{\underline{B}}^T (\underline{\underline{BD}}^{-1} \underline{\underline{B}}^T)^{-1} \\ &= (\underline{\underline{BD}}^{-1} \underline{\underline{B}}^T)^{-1} \sum_i^E \underline{\underline{B}}^i (\underline{\underline{D}}^i)^{-1} \underline{\underline{S}}^i (\underline{\underline{D}}^i)^{-1} (\underline{\underline{B}}^i)^T (\underline{\underline{BD}}^{-1} \underline{\underline{B}}^T)^{-1}\end{aligned}\quad (6.69)$$

donde

$$\underline{\underline{D}} = \begin{bmatrix} \underline{\underline{D}}^1 & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & \cdots & \underline{\underline{D}}^E \end{bmatrix} \quad (6.70)$$

en la cual cada $\underline{\underline{D}}^i$ es la matriz diagonal con los elementos $\delta_i^\dagger(x)$ correspondientes a los puntos $x \in \partial\Omega_{i,h} \cap \Gamma_h$.

Entonces el método One-Level FETI queda en términos del método de Gradiente Conjugado preconditionado como a continuación se muestra:

1.- Inicializa

$$\begin{aligned}\underline{\underline{\lambda}}_0 &= \underline{\underline{QG}} \left(\underline{\underline{G}}^T \underline{\underline{QG}} \right)^{-1} \underline{\underline{e}} + \underline{\underline{\mu}}, \underline{\underline{\mu}} \in \text{Rango}(\underline{\underline{P}}) \\ \underline{\underline{r}}_0 &= \underline{\underline{d}} - \underline{\underline{F}} \underline{\underline{\lambda}}^0 \\ \underline{\underline{\beta}}^0 &= 0 \\ \underline{\underline{p}}^1 &= 0\end{aligned}$$

2.- Itera $k = 1, 2, \dots$ hasta converger

$$\begin{aligned}\underline{\underline{q}}^{k-1} &= \underline{\underline{P}}^T \underline{\underline{r}}^{k-1} \\ \underline{\underline{z}}^{k-1} &= \hat{\underline{\underline{M}}}^{-1} \underline{\underline{q}}^{k-1} \\ \underline{\underline{y}}^{k-1} &= \underline{\underline{P}} \underline{\underline{z}}^{k-1} \\ \underline{\underline{\beta}}^k &= \frac{\langle \underline{\underline{y}}^{k-1}, \underline{\underline{q}}^{k-1} \rangle}{\langle \underline{\underline{y}}^{k-2}, \underline{\underline{q}}^{k-2} \rangle} \\ \underline{\underline{p}}^k &= \underline{\underline{y}}^{k-1} + \underline{\underline{\beta}}^k \underline{\underline{p}}^{k-1} \\ \underline{\underline{\alpha}}^k &= \frac{\langle \underline{\underline{y}}^{k-1}, \underline{\underline{q}}^{k-1} \rangle}{\langle \underline{\underline{p}}^k, \underline{\underline{F}} \underline{\underline{q}}^k \rangle} \\ \underline{\underline{\lambda}}^k &= \underline{\underline{\lambda}}^{k-1} + \underline{\underline{\alpha}}^k \underline{\underline{p}}^k \\ \underline{\underline{r}}^k &= \underline{\underline{r}}^{k-1} - \underline{\underline{\alpha}}^k \underline{\underline{F}} \underline{\underline{p}}^k\end{aligned}$$

Así, una vez calculando el multiplicador de Lagrange $\underline{\lambda}$ obtenemos la solución en los nodos de la frontera interior \underline{u} mediante

$$\underline{u} = \underline{S}^\dagger (\underline{f} - \underline{B}^T \underline{\lambda}) \quad (6.71)$$

y para obtener la solución en los nodos interiores de cada subdominio se recurre a la aplicación de la Ec (19.73) del método de subestructuración o complemento de Schur, solucionando así el problema.

Como se ha mencionado a lo largo de esta sección, el algoritmo One-Level FETI es determinado por la elección de \underline{Q} y $\hat{\underline{M}}^{-1}$. Para la elección $\underline{Q} = \hat{\underline{M}}^{-1}$, cada paso correspondiente al método de Gradiente Conjugado supone una aplicación de \underline{P}^T y uno de \underline{P} , la solución de un problema con condiciones de frontera Dirichlet sobre los subdominios es necesario para la aplicación de $\hat{\underline{M}}^{-1}$ y la solución de un problema con condiciones de frontera Neumann sobre los subdominios es necesario para la aplicación de \underline{F} en el cálculo del nuevo residual. Entonces la aplicación de \underline{P}^T y \underline{P} implica dos adicionales aplicaciones de $\underline{Q} = \hat{\underline{M}}^{-1}$ y la solución de dos problemas sobre la partición gruesa, esto es un total de un problema con condiciones de frontera tipo Neumann y tres problemas con condiciones de frontera tipo Dirichlet sobre cada subdominio y dos problemas sobre la partición gruesa en cada paso de la iteración.

6.2.3 El Algoritmo One-Level FETI Simplificado

Aquí trataremos la versión particular del algoritmo Dirichlet-Dirichlet en el cual $\underline{R} = \underline{0}$, entonces reformulando el problema definido por la Ec. (6.1) a uno reducido a la interfase Γ por medio de elementos finitos, como un problema de minimización con restricciones impuestas por lo requerimientos de continuidad en Γ queda como:

Encontrar $\underline{u} \in W$ tal que

$$J(\underline{u}) = \frac{1}{2} \left. \begin{array}{l} \langle \underline{S}\underline{u}, \underline{u} \rangle - \langle \underline{f}, \underline{u} \rangle \rightarrow \min \\ \underline{B}\underline{u} = 0 \end{array} \right\} \quad (6.72)$$

donde la matriz por bloques \underline{S} es formada por las matrices \underline{S}^i Ec. (6.15) del complemento de Schur en el i -ésimo subdominio, el vector por bloques \underline{u} es formado por los vectores \underline{u}_i solución de la frontera interior en cada i -

ésimo subdominio y el vector \underline{f} es formado por los vectores \underline{f}_i de la frontera interior en cada i -ésimo subdominio, i.e.

$$\underline{S} = \begin{bmatrix} \underline{S}^1 & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & \cdots & \underline{S}^E \end{bmatrix}, \underline{u} = \begin{bmatrix} \underline{u}_1 \\ \vdots \\ \vdots \\ \underline{u}_E \end{bmatrix}, \underline{f} = \begin{bmatrix} \underline{f}_1 \\ \vdots \\ \vdots \\ \underline{f}_E \end{bmatrix} \quad (6.73)$$

y la matriz \underline{B} es formada por las matrices \underline{B}^i en cada i -ésimo subdominio tal que la solución asociada a más de un subdominio coincida, i.e.

$$\underline{B} = [\underline{B}^1, \quad \cdots \quad \cdots, \quad \underline{B}^E] \quad (6.74)$$

es construida de $\{0, 1, -1\}$ tal que los valores de la solución \underline{u} asociada a más de un subdominio coincida cuando $\underline{B}\underline{u} = 0$, donde la elección de \underline{B} no es única.

Nótese que un mismo nodo en la frontera interior pertenece a dos o más subdominios, por ello es necesario algún mecanismo para asegurar que la solución asociada a más de un subdominio coincida.

El problema (6.72) es soluble de manera única ya que el

$$\text{Kernel}(\underline{S}) \cap \text{Kernel}(\underline{B}) = \{0\} \quad (6.75)$$

lo cual indica que \underline{S} es invertible sobre el $\text{Kernel}(\underline{B})$.

Pero introduciendo un vector de multiplicadores de Lagrange $\underline{\lambda}$ para imponer las restricciones $\underline{B}\underline{u} = 0$, obtenemos una formulación silla de la Ec. (6.72):

Encontrar $(\underline{u}, \underline{\lambda}) \in W \times U$ tal que

$$\begin{cases} \underline{S}\underline{u} + \underline{B}^T \underline{\lambda} = \underline{f} \\ \underline{B}\underline{u} = 0 \end{cases} \quad (6.76)$$

la solución $\underline{\lambda}$ de la Ec. (6.76) es única salvo la adición de un elemento del $\text{Kernel}(\underline{B}^T)$. El espacio de multiplicadores de Lagrange U , es por lo tanto elegido como el $\text{Rango}(\underline{B})$. Este espacio puede ser entendido como el espacio de las funciones de salto en W .

Una solución \underline{u} a la primera ecuación de la Ec. (6.76) existe si y sólo si $\underline{f} - \underline{B}^T \underline{\lambda} \in \text{Rango}(\underline{S})$, sustituyendo la expresión para \underline{u} dentro de la segunda ecuación de la Ec. (6.76) obtenemos

$$\underline{B} \underline{S}^\dagger \underline{B}^T \underline{\lambda} = \underline{B} \underline{S}^\dagger \underline{f} \quad (6.77)$$

donde \underline{S}^\dagger es la inversa de \underline{S} y tomado como preconditionador -el más básico- para FETI

$$\underline{M}^{-1} = \underline{B} \underline{S} \underline{B}^T = \sum_{i=1}^E \underline{B}^i \underline{S}^i (\underline{B}^i)^T. \quad (6.78)$$

Entonces el método one-level FETI simplificado queda en términos del método de Gradiente Conjugado preconditionado como a continuación se muestra:

1.- Inicializa

$$\begin{aligned} \underline{\lambda}_0 &= \underline{0} \\ \underline{r}_0 &= (\underline{B} \underline{S}^\dagger \underline{f}) - (\underline{B} \underline{S}^\dagger \underline{B}^T) \underline{\lambda}^0 \\ \underline{\beta}^0 &= 0 \\ \underline{p}^1 &= 0 \end{aligned}$$

2.- Itera $k = 1, 2, \dots$ hasta converger

$$\begin{aligned} \underline{q}^{k-1} &= \underline{r}^{k-1} \\ \underline{z}^{k-1} &= (\underline{B} \underline{S} \underline{B}^T)^{-1} \underline{q}^{k-1} \\ \underline{y}^{k-1} &= \underline{z}^{k-1} \\ \underline{\beta}^k &= \frac{\langle \underline{y}^{k-1}, \underline{q}^{k-1} \rangle}{\langle \underline{y}^{k-1}, \underline{q}^{k-1} \rangle} \\ \underline{p}^k &= \underline{y}^{k-1} + \underline{\beta}^k \underline{p}^{k-1} \\ \underline{\alpha}^k &= \frac{\langle \underline{y}^{k-1}, \underline{q}^{k-1} \rangle}{\langle \underline{p}^k, (\underline{B} \underline{S}^\dagger \underline{B}^T) \underline{q}^k \rangle} \\ \underline{\lambda}^k &= \underline{\lambda}^{k-1} + \underline{\alpha}^k \underline{p}^k \\ \underline{r}^k &= \underline{r}^{k-1} - \underline{\alpha}^k (\underline{B} \underline{S}^\dagger \underline{B}^T) \underline{p}^k \end{aligned}$$

Así, una vez calculando el multiplicador de Lagrange $\underline{\lambda}$ obtenemos la solución en los nodos de la frontera interior \underline{u} mediante

$$\underline{u} = \underline{S}^\dagger (\underline{f} - \underline{B}^T \underline{\lambda}) \quad (6.79)$$

y para obtener la solución en los nodos interiores de cada subdominio se recurre a la aplicación de la Ec (19.73) del método de subestructuración o complemento de Schur, solucionando así el problema.

6.3 Dual-Primal FETI

El método dual-primal FETI (FETI-DP) fue introducido posteriormente que el método One-Level FETI, siendo esto una gran contribución a la teoría para la resolución de problemas elípticos de segundo y cuarto orden. Este método se basa en hacer cumplir un número relativamente pequeño de restricciones de continuidad a través de la frontera interior en cada paso de las iteraciones en comparación con el método de One-Level FETI.

Primeramente consideremos a $\Omega \subset \mathbb{R}^n$ un dominio, y $\Pi = \{\Omega_1, \dots, \Omega_E\}$ una partición o descomposición en subdominios del dominio Ω , además sea Γ_i la frontera de interior del subdominio Ω_i y Γ la frontera interior del dominio Ω_i .

Asumiremos que las discontinuidades en la ecuación diferencial parcial -si existen- estarán alineadas con las fronteras de los subdominios, tal que en cada subdominio Ω_i , el coeficiente $\rho(x)$ de la ecuación tenga un valor constante, sin pérdida de generalidad se asumirá $\rho_i > 0$. Además, denotaremos a W_i como el espacio de trazas de Ω_i , es decir

$$W_i = W^h(\partial\Omega_i \cap \Gamma), \text{ con } i = 1, \dots, E \quad (6.80)$$

también denotaremos W como el espacio producto del espacio de las trazas, es decir

$$W = \prod_{i=1}^E W_i \quad (6.81)$$

y la extensión armónica discreta (6.1.2) por pedazos de \underline{u}_Γ por $\mathcal{H}(\underline{u}_\Gamma)$.

Así, en lo que resta de esta sección, se trabajará casi exclusivamente con funciones en el espacio de trazas W_i y cuando sea conveniente, se considerarán como un elemento representante de las funciones armónicas discretas en Ω_i , de tal forma que $\underline{w} \in W$, $\mathcal{H}_i(\underline{w})$ denotará la extensión por pedazos de la armónica discreta sobre todo el subdominio Ω_i , entenderemos $\mathcal{H}(\underline{w})$ como un elemento en el espacio producto W con componentes $\mathcal{H}_i(\underline{w}_i)$.

La aproximación por medio de elementos finitos estándar al problema elíptico es continua a través de la frontera interior Γ y denotamos al correspondiente subespacio de W por \hat{W} . En este método se usan subespacios

intermedios \tilde{W} de W , de tal manera que el complemento de Schur usado en los cálculos será estrictamente positivo definido.

Denotamos $\tilde{W}^h(\Omega)$ como un subespacio de $\prod_{i=1}^E W_i^h(\Omega_i)$ el cual es igual a \tilde{W} cuando es restringido a la frontera interior Γ . Adicionalmente se introducen dos subespacios $\hat{W}_\Pi, \tilde{W}_\Delta \subset \hat{W}$ correspondiendo a la parte primal y dual del espacio \tilde{W} , además $\hat{W} = \hat{W}_\Pi \oplus \tilde{W}_\Delta$. Relacionamos al espacio dual \tilde{W}_Δ con los saltos en la frontera interior Γ y los multiplicadores de Lagrange son introducidos para eliminar tales saltos.

En el método FETI-DP se expresará al complemento de Schur $\underline{\tilde{S}}$ relacionado con el espacio dual \tilde{W}_Δ , así, en esta sección, \tilde{W} consiste de funciones en W que toman el mismo valor en los vértices del subdominio y puede escribirse como

$$\tilde{W} = \hat{W}_\Pi \oplus \tilde{W}_\Delta. \quad (6.82)$$

Aquí $\hat{W}_\Pi \subset \hat{W}$ es el espacio de funciones con la frontera interior Γ_h continua que se nulifican en todos los nodos sobre Γ excepto en los vértices del subdominio Ω_i , con $i = 1, \dots, E$, y \tilde{W}_Δ es la suma directa de los subespacios locales $\tilde{W}_{\Delta,i}$. i.e. $\tilde{W}_\Delta = \oplus \tilde{W}_{\Delta,i}$ donde $\tilde{W}_{\Delta,i} \subset W_i$ y consiste de las funciones locales sobre $\partial\Omega_i$ que se nulifican en los vértices de Ω_i .

El continuo de los grados de libertad asociados con los vértices de cada subdominio y con el subespacio \hat{W}_Π es llamado primal (Π), mientras aquellos -potenciales discontinuidades a través de Γ - asociados con los subespacios $\tilde{W}_{\Delta,i}$ y con el interior de la frontera de cada subdominio Ω_i es llamado dual (Δ).

Además consideramos la familia de funciones de peso $\delta_i \in W_i$, las cuales están asociadas con cada $\partial\Omega_i$ y definidas para $\gamma \in [\frac{1}{2}, \infty)$ por la suma de contribuciones de Ω_i con los vecinos pertinentes, así definimos

$$\delta_i(x) = \frac{\sum_{j \in N_x} \rho_j^\gamma}{\rho_i^\gamma}, \quad x \in \partial\Omega_{i,h} \cap \Gamma_h \quad (6.83)$$

donde N_x es el conjunto de índices j de las subregiones tal que $x \in \partial\Omega_{i,h}$, la pseudo-inversa δ_i^\dagger es definida por

$$\delta_i^\dagger(x) = (\delta_i(x))^{-1}, \quad x \in \partial\Omega_{i,h} \cap \Gamma_h. \quad (6.84)$$

Sea $\underline{\underline{A}}$ la matriz de carga, la cual es obtenida por la restricción

$$\underline{\underline{A}} = \text{diag} \{ \underline{\underline{A}}^1, \dots, \underline{\underline{A}}^E \} \quad (6.85)$$

de $\prod_i^E W^h(\Omega_i)$ a $\tilde{W}^h(\Omega)$, donde $\underline{\underline{A}}^i$ es la matriz de carga generada por el método de subestructuración en el subdominio i , notemos que $\underline{\underline{A}}$ no es una matriz diagonal en bloques ya que ahora están acoplados los distintos subdominios que tienen un vértice en común. Particionando $\underline{\underline{A}}$ como

$$\underline{\underline{A}} = \begin{bmatrix} \underline{\underline{A}}_{II} & \underline{\underline{A}}_{I\Pi} & \underline{\underline{A}}_{I\Delta} \\ \left(\underline{\underline{A}}_{I\Pi}\right)^T & \underline{\underline{A}}_{\Pi\Pi} & \underline{\underline{A}}_{\Pi\Delta} \\ \left(\underline{\underline{A}}_{I\Delta}\right)^T & \left(\underline{\underline{A}}_{\Pi\Delta}\right)^T & \underline{\underline{A}}_{\Delta\Delta} \end{bmatrix} \quad (6.86)$$

$$\underline{\underline{f}} = \begin{bmatrix} \underline{f}_I \\ \underline{f}_\Pi \\ \underline{f}_\Delta \end{bmatrix} \quad (6.87)$$

donde el superíndice I se refiere a los grados de libertad asociados a los nodos internos de los subdominios Ω_i , Π se refiere a los asociados con los vértices de los subdominios Ω_i y Δ a los asociados al interior de las caras de la frontera de los subdominios Ω_i .

Notemos que $\underline{\underline{A}}_{II}$ y $\underline{\underline{A}}_{\Delta\Delta}$ son matrices diagonales por bloques y cada bloque corresponde a un dominio individual Ω_i y cada no-cero de $\underline{\underline{A}}_{I\Delta}$ representa un acoplamiento entre los grados de libertad asociados a un subdominio. $\underline{\underline{A}}$ es obtenida por el ensamble parcial de las contribuciones locales asociadas con cada subdominio Ω_i .

Eliminado las variables I y Π , entonces el complemento de Schur asociado a los grados de libertad del conjunto Δ , del interior de las caras de las fronteras $\partial\Omega_i$, queda como

$$\underline{\underline{S}} = \underline{\underline{A}}_{\Delta\Delta} - \begin{bmatrix} \left(\underline{\underline{A}}_{I\Delta}\right)^T & \left(\underline{\underline{A}}_{\Pi\Delta}\right)^T \end{bmatrix} \begin{bmatrix} \underline{\underline{A}}_{II} & \underline{\underline{A}}_{I\Pi} \\ \left(\underline{\underline{A}}_{I\Pi}\right)^T & \underline{\underline{A}}_{\Pi\Pi} \end{bmatrix}^{-1} \begin{bmatrix} \underline{\underline{A}}_{I\Delta} \\ \underline{\underline{A}}_{\Pi\Delta} \end{bmatrix} \quad (6.88)$$

$$\underline{\underline{f}}_\Delta = \underline{f}_\Delta - \begin{bmatrix} \left(\underline{\underline{A}}_{I\Delta}\right)^T & \left(\underline{\underline{A}}_{\Pi\Delta}\right)^T \end{bmatrix} \begin{bmatrix} \underline{\underline{A}}_{II} & \underline{\underline{A}}_{I\Pi} \\ \left(\underline{\underline{A}}_{I\Pi}\right)^T & \underline{\underline{A}}_{\Pi\Pi} \end{bmatrix}^{-1} \begin{bmatrix} \underline{f}_I \\ \underline{f}_\Pi \end{bmatrix} \quad (6.89)$$

también obtenemos una reducción del lado derecho \tilde{f}_Δ del vector de carga asociado con los subdominios individuales. Denotamos por $\underline{u}_\Delta \in \tilde{W}_\Delta$ el vector de grados de libertad asociado a las caras de los subdominios.

Reformulando el problema definido por la Ec. (6.1) a uno reducido a un segundo subespacio \tilde{W}_Δ como un problema de minimización con restricciones impuestas por lo requerimientos de continuidad en Γ queda como:

Encontrar $\underline{u}_\Delta \in \tilde{W}$ tal que

$$J(\underline{u}_\Delta) = \frac{1}{2} \left\langle \underline{\tilde{S}} \underline{u}_\Delta, \underline{u}_\Delta \right\rangle - \left\langle \tilde{f}_\Delta, \underline{u}_\Delta \right\rangle \rightarrow \min \left. \vphantom{J(\underline{u}_\Delta)} \right\} \quad (6.90)$$

$$\underline{B}_\Delta \underline{u}_\Delta = 0$$

la matriz \underline{B}_Δ es construida de $\{0, 1, -1\}$ tal que los valores de la solución \underline{u}_Δ asociada a más de un subdominio coincida cuando $\underline{B}_\Delta \underline{u}_\Delta = 0$, donde la elección de \underline{B}_Δ no es única.

Pero introduciendo un vector de multiplicadores de Lagrange $\underline{\lambda} \in V = \text{Rango}(\underline{B}_\Delta)$ para imponer las restricciones $\underline{B}_\Delta \underline{u} = 0$, obtenemos una formulación silla de la Ec. (6.90). Eliminando el subvector \underline{u}_Δ , y obteniendo el siguiente sistema de multiplicadores de Lagrange

$$\underline{F} \underline{\lambda} = \underline{d} \quad (6.91)$$

con

$$\underline{F} = \underline{B}_\Delta \left(\underline{\tilde{S}} \right)^{-1} \left(\underline{B}_\Delta \right)^T, \quad \underline{d} = \underline{B}_\Delta \left(\underline{\tilde{S}} \right)^{-1} \tilde{f}_\Delta. \quad (6.92)$$

Una vez $\underline{\lambda}$ encontrada, podemos resolver hacia atrás y obtener

$$\underline{u}_\Delta = \left(\underline{\tilde{S}} \right)^{-1} \left(\tilde{f}_\Delta - \left(\underline{B}_\Delta \right)^T \underline{\lambda} \right) \in \tilde{W}_\Delta. \quad (6.93)$$

Los valores de la solución en el interior de los subdominios \underline{u}_I y en los vértices de los subdominios \underline{u}_{Π} son obtenidos como un subproducto cuando se resuelve el sistema lineal con la matriz por bloques dada por la Ec. (6.88).

Introduciendo una matriz de escalamiento diagonal \underline{D}_Δ^i , donde cada uno de los elementos de la diagonal corresponde a un multiplicador de Lagrange que fuerzan la continuidad entre los valores de los nodos de algunas $\underline{u}_i \in W^i$ y $\underline{u}_j \in W^j$ en algún punto $\underline{x} \in \Gamma_h$ y está dado por $\delta_i^\dagger(\underline{x})$. También se define un escalamiento del salto por medio del operador

$$\underline{B}_{D,\Delta} = \left[\underline{D}_\Delta^1 \underline{B}_\Delta^1, \dots, \underline{D}_\Delta^E \underline{B}_\Delta^E \right] \quad (6.94)$$

aquí, como antes, el bloque \underline{B}_{Δ}^i es obtenido por extracción de columnas de \underline{B}_{Δ} asociadas con el espacio local W_i .

Resolviendo el sistema dual dado por la Ec. (6.91) usando el método de Gradiente Conjugado con el preconditionador

$$\begin{aligned}\underline{M}^{-1} &= \underline{B}_{D,\Delta} \underline{S}_{\Delta} \left(\underline{B}_{D,\Delta} \right)^T \\ &= \sum_{i=1}^E \underline{D}_{\Delta}^i \underline{B}_{\Delta}^i \underline{S}_{\Delta}^i \left(\underline{B}_{\Delta}^i \right)^T \underline{D}_{\Delta}^i\end{aligned}$$

donde \underline{S}_{Δ}^i es la restricción del complemento de Schur local \underline{S}^i a $\tilde{W}_{\Delta,i} \subset W^i$.

El método FETI-DP es un método de Gradiente Conjugado preconditionado para resolver el sistema preconditionado

$$\underline{M}^{-1} \underline{F} \lambda = \underline{M}^{-1} \underline{d} \quad (6.95)$$

quedando en términos del método de Gradiente Conjugado preconditionado como a continuación se muestra:

1.- Inicializa

$$\begin{aligned}\underline{r}^0 &= \underline{d} - \underline{F} \lambda^0 \\ \beta^1 &= 0 \\ \underline{p}^1 &= \underline{z}^0\end{aligned}$$

2.- Itera $k = 1, 2, \dots$ hasta converger

$$\begin{aligned}\underline{z}^{k-1} &= \underline{M}^{-1} \underline{r}^{k-1} \\ \beta^k &= \frac{\langle \underline{z}^{k-1}, \underline{r}^{k-1} \rangle}{\langle \underline{z}^{k-2}, \underline{r}^{k-2} \rangle} \\ \underline{p}^k &= \underline{z}^{k-1} + \beta^k \underline{p}^{k-1} \\ \alpha^k &= \frac{\langle \underline{z}^{k-1}, \underline{r}^{k-1} \rangle}{\langle \underline{p}^k, \underline{F} \underline{p}^k \rangle} \\ \underline{\lambda}^k &= \underline{\lambda}^{k-1} + \alpha^k \underline{p}^k \\ \underline{r}^k &= \underline{r}^{k-1} - \alpha^k \underline{F} \underline{p}^k\end{aligned}$$

Así, una vez calculando el multiplicador de Lagrange $\underline{\lambda}$ obtenemos la solución en los nodos de la frontera interior \underline{u}_Δ mediante

$$\underline{u}_\Delta = \left(\underline{\tilde{S}} \right)^{-1} \left(\underline{\tilde{f}}_\Delta - \left(\underline{B}_\Delta \right)^T \underline{\lambda} \right) \quad (6.96)$$

y para obtener la solución en los nodos interiores de cada subdominio se recurre a la aplicación de la Ec (19.73) del método de subestructuración o complemento de Schur, solucionando así el problema.

El método FETI-DP presenta las siguientes ventajas:

- El algoritmo no requiere de la caracterización del kernel de los problemas locales con condiciones de frontera Neumann. Adicionalmente, la imposición de adicionales restricciones en cada iteración siempre crea problemas locales no singulares y al mismo tiempo proporciona un subyacente problema grueso global.
- El algoritmo no requiere la introducción de matrices de escalabilidad Q .
- El algoritmo en el método de Gradiente Conjugado puede usar un valor inicial arbitrario $\underline{\lambda}_0$.

6.4 Variantes para la Implementación Numérica

El método FETI-DP es un método de Gradiente Conjugado preconditionado para resolver el sistema preconditionado

$$\underline{M}^{-1} \underline{F} \underline{\lambda} = \underline{M}^{-1} \underline{d} \quad (6.97)$$

donde

$$\begin{aligned} \underline{M}^{-1} &= \underline{B}_{D,\Delta} \underline{S}_\Delta \left(\underline{B}_{D,\Delta} \right)^T \\ &= \sum_{i=1}^E \underline{D}_\Delta^i \underline{B}_\Delta^i \underline{S}_\Delta^i \left(\underline{B}_\Delta^i \right)^T \underline{D}_\Delta^i \\ \underline{F} &= \underline{B}_\Delta \left(\underline{\tilde{S}} \right)^{-1} \left(\underline{B}_\Delta \right)^T \quad \text{y} \quad \underline{d} = \underline{B}_\Delta \left(\underline{\tilde{S}} \right)^{-1} \underline{\tilde{f}}_\Delta. \end{aligned} \quad (6.98)$$

La implementación computacional queda como:

$$\begin{aligned} \underline{r} &= \underline{d} - \underline{F}\underline{u} \\ \underline{w} &= \underline{\underline{M}}^{-1}\underline{r} \\ \underline{v} &= \underline{\underline{M}}^{-1}\underline{w} \\ \alpha &= \sum_{j=1}^n w_j^2 \\ k &= 1 \\ \text{Mientras que } & k \leq N \end{aligned}$$

Si $\|\underline{v}\|_\infty < \varepsilon$ Salir

$$\begin{aligned} \underline{x} &= \underline{\underline{F}}\underline{v} \\ t &= \frac{\alpha}{\sum_{j=1}^n v_j x_j} \end{aligned}$$

$$\underline{u} = \underline{u} + t\underline{v}$$

$$\underline{r} = \underline{r} - t\underline{x}$$

$$\begin{aligned} \underline{w} &= \underline{\underline{M}}^{-1}\underline{r} \\ \beta &= \sum_{j=1}^n w_j^2 \end{aligned}$$

Si $\|\underline{r}\|_\infty < \varepsilon$ Salir

$$s = \frac{\beta}{\alpha}$$

$$\underline{v} = \underline{\underline{M}}^{-1}\underline{w} + s\underline{v}$$

$$\alpha = \beta$$

$$k = k + 1$$

Así, en esta sección describiremos cómo hacer el cálculo en donde estén involucradas las matrices $\underline{\underline{S}}$ y $\underline{\underline{S}}^{-1}$ (ya que estas matrices son virtuales, ya que todo queda en función de las matrices locales $\underline{\underline{S}}^\alpha$), el cálculo de los nodos interiores y otras variantes de la implementación numérica que ofrece mejoras computacionales al modelo.

6.4.1 Implementación de la Matriz $\underline{\underline{J}}$

Primeramente indicaremos una forma de construir la matriz $\underline{\underline{B}}$ y como construir a la matriz $\underline{\underline{J}}$ véase [62], la cual en nuestras pruebas presenta mejores resultados en la implementación computacional que la matriz $\underline{\underline{B}}$, para ello recordando, la matriz $\underline{\underline{B}}$ es formada por las matrices $\underline{\underline{B}}^i$ en cada i -ésimo

subdominio tal que la solución asociada a más de un subdominio coincida, i.e.

$$\underline{\underline{B}} = [\underline{\underline{B}}^1, \dots \dots, \underline{\underline{B}}^E] \quad (6.99)$$

es construida de $\{0, 1, -1\}$ tal que los valores de la solución \underline{u} asociada a más de un subdominio coincida cuando $\underline{\underline{B}}\underline{u} = 0$, donde la elección de $\underline{\underline{B}}$ no es única.

Estructura de $\underline{\underline{B}}^{(q)}$, donde q es un nodo de multiplicidad 2 queda como

$$\underline{\underline{B}}^{(q)} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \quad (6.100)$$

y la estructura de $\underline{\underline{B}}^{(q)}$, donde q es un nodo de multiplicidad 4 queda como

$$\underline{\underline{B}}^{(q)} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix}. \quad (6.101)$$

Por otro lado, la matriz $\underline{\underline{j}}$ es formada por las matrices $\underline{\underline{j}}_i$ en cada i -ésimo subdominio tal que la solución \underline{u} asociada a más de un subdominio coincida, i.e.

$$\underline{\underline{j}} = [\underline{\underline{j}}^1, \dots \dots, \underline{\underline{j}}^E] \text{ cy } \underline{\underline{j}}\underline{u} = 0. \quad (6.102)$$

Para ello, primeramente definimos dos matrices $\underline{\underline{a}}$ y $\underline{\underline{j}}$ con la propiedad de que $\underline{\underline{I}} = \underline{\underline{a}} + \underline{\underline{j}}$, donde $\underline{\underline{a}}$ y $\underline{\underline{j}}$ son ambas simétricas, no-negativas e idempotentes, donde además $\underline{\underline{a}}\underline{\underline{j}} = \underline{\underline{j}}\underline{\underline{a}} = \underline{\underline{0}}$, mediante la matriz local de promedio definida como $\underline{\underline{a}}^{(q)} = \frac{1}{|Z|} \sum_{q \in Z} \underline{u}(q)$ donde $|Z|$ es la multiplicidad de los nodos primales q , y la matriz local de salto definida como $\underline{\underline{j}}^{(q)} = \underline{\underline{I}} - \underline{\underline{a}}^{(q)}$.

Así, la estructura de $\underline{\underline{a}}^{(q)}$ y $\underline{\underline{j}}^{(q)}$, donde q es un nodo de multiplicidad 2 queda como

$$\underline{\underline{a}}^{(q)} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad \text{y} \quad \underline{\underline{j}}^{(q)} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad (6.103)$$

y la estructura de $\underline{\underline{a}}^{(q)}$ y $\underline{\underline{j}}^{(q)}$, donde q es un nodo de multiplicidad 4 queda como

$$\underline{\underline{a}}^{(q)} = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix} \quad y \quad \underline{\underline{j}}^{(q)} = \begin{bmatrix} \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} \end{bmatrix}. \quad (6.104)$$

Más concretamente supóngase que se tiene un dominio que es particionado en 2×2 y cada subdominio en 2×2 , usando FETI One-Level tenemos que hay 4 subdominios, en cada subdominio se tiene 4 elementos, 9 nodos, 1 nodo interior y 3 nodos de frontera interior. En total se tienen 4 nodos interiores, 5 nodos en la frontera interior y 12 multiplicadores de Lagrange.

Suponiendo que se construyera las matrices globales, entonces la estructura de las matrices $\underline{\underline{B}}$, $\underline{\underline{a}}$ y $\underline{\underline{j}}$ sería la siguiente:

$$\underline{\underline{B}} = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix},$$

$$\underline{\underline{\Omega}} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

y

$$\underline{\underline{j}} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{-1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{3}{4} & 0 & 0 & \frac{-1}{4} & 0 & \frac{-1}{4} & 0 & 0 & \frac{-1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{-1}{2} & 0 & 0 & 0 & 0 \\ \frac{-1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{-1}{4} & 0 & 0 & \frac{3}{4} & 0 & \frac{-1}{4} & 0 & 0 & \frac{-1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{-1}{2} & 0 \\ 0 & \frac{-1}{4} & 0 & 0 & \frac{-1}{4} & 0 & \frac{3}{4} & 0 & 0 & \frac{-1}{4} & 0 & 0 \\ 0 & 0 & \frac{-1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{-1}{2} \\ 0 & \frac{-1}{4} & 0 & 0 & \frac{-1}{4} & 0 & \frac{-1}{4} & 0 & 0 & \frac{3}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{-1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{2} & 0 & \frac{1}{2} \end{bmatrix}.$$

6.4.2 Cálculo de la Matriz $\underline{\underline{S}}$

La matriz $\underline{\underline{S}}$ definida por

$$\underline{\underline{S}} = \underline{\underline{A}}_{\Delta\Delta} - \underline{\underline{A}}_{\Delta\Pi} \left(\underline{\underline{A}}_{\Pi\Pi} \right)^{-1} \underline{\underline{A}}_{\Pi\Delta} \quad (6.105)$$

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

$$\underline{\underline{S}}^{\alpha} = \underline{\underline{A}}_{\Delta\Delta}^{\alpha} - \underline{\underline{A}}_{\Delta\Pi}^{\alpha} \left(\underline{\underline{A}}_{\Pi\Pi}^{\alpha} \right)^{-1} \underline{\underline{A}}_{\Pi\Delta}^{\alpha}. \quad (6.106)$$

Así que, las matrices locales $\underline{\underline{S}}^\alpha$ y $\left(\underline{\underline{A}}_{\text{III}}^\alpha\right)^{-1}$ no se construyen, ya que estas serían matrices densas y su construcción es computacionalmente muy costosa, y como sólo nos interesa el producto $\underline{\underline{S}}y_\Gamma$, o más precisamente $\left[\sum_{\alpha=1}^E \underline{\underline{S}}^\alpha\right] y_\Gamma$, entonces si llamamos $\underline{y}_\Gamma^\alpha$ al vector correspondiente al subdominio α , entonces tendremos

$$\tilde{\underline{u}}_\Gamma^\alpha = \left(\underline{\underline{A}}_{\Delta\Delta}^\alpha - \underline{\underline{A}}_{\Delta\Pi}^\alpha \left(\underline{\underline{A}}_{\text{III}}^\alpha \right)^{-1} \underline{\underline{A}}_{\Pi\Delta}^\alpha \right) \underline{y}_\Gamma^\alpha. \quad (6.107)$$

Para evaluar eficientemente esta expresión, realizamos las siguientes operaciones equivalentes

$$\begin{aligned} \underline{x1} &= \underline{\underline{A}}_{\Delta\Delta}^\alpha \underline{y}_\Gamma^\alpha & (6.108) \\ \underline{x2} &= \left(\underline{\underline{A}}_{\Delta\Pi}^\alpha \left(\underline{\underline{A}}_{\text{III}}^\alpha \right)^{-1} \underline{\underline{A}}_{\Pi\Delta}^\alpha \right) \underline{y}_\Gamma^\alpha \\ \tilde{\underline{u}}_\Gamma^\alpha &= \underline{x1} - \underline{x2} \end{aligned}$$

la primera y tercera expresión no tienen ningún problema en su evaluación, para la segunda expresión tendremos que hacer

$$\underline{x3} = \underline{\underline{A}}_{\Pi\Delta}^\alpha \underline{y}_\Gamma^\alpha \quad (6.109)$$

con este resultado intermedio deberíamos calcular

$$\underline{x4} = \left(\underline{\underline{A}}_{\text{III}}^\alpha \right)^{-1} \underline{x3} \quad (6.110)$$

pero como no contamos con $\left(\underline{\underline{A}}_{\text{III}}^\alpha \right)^{-1}$, entonces multiplicamos la expresión por $\underline{\underline{A}}_{\text{III}}^\alpha$ obteniendo

$$\underline{\underline{A}}_{\text{III}}^\alpha \underline{x4} = \underline{\underline{A}}_{\text{III}}^\alpha \left(\underline{\underline{A}}_{\text{III}}^\alpha \right)^{-1} \underline{x3} \quad (6.111)$$

al simplificar, tenemos

$$\underline{\underline{A}}_{\text{III}}^\alpha \underline{x4} = \underline{x3}. \quad (6.112)$$

Esta última expresión puede ser resuelta usando Factorización LU o Gradiente Conjugado (cada una de estas opciones tiene ventajas y desventajas que deben ser evaluadas al momento de implementar el código para un problema particular). Una vez obtenido $\underline{x4}$, podremos calcular

$$\underline{x2} = \underline{\underline{A}}_{\Delta\Pi}^\alpha \underline{x4} \quad (6.113)$$

así

$$\tilde{\underline{u}}_\Gamma^\alpha = \underline{x1} - \underline{x2} \quad (6.114)$$

completando la secuencia de operaciones necesaria para obtener $\underline{\underline{S}}_\alpha y_\Gamma^\alpha$.

6.4.3 Cálculo de la Matriz $\underline{\underline{S}}^{-1}$

En los algoritmos descritos anteriormente, interviene la evaluación de $\underline{\underline{S}}^{-1}$. Dado que la matriz $\underline{\underline{S}}$ no se construye, entonces la matriz $\underline{\underline{S}}^{-1}$ tampoco es necesaria construirla, en lugar de ello se procede de la siguiente manera. Se asume que en las operaciones anteriores al producto de $\underline{\underline{S}}^{-1}$, se ha obtenido un vector, supongamos que es \underline{v} , entonces para evaluar

$$\underline{u} = \underline{\underline{S}}^{-1}\underline{v} \quad (6.115)$$

se procede a multiplicar por $\underline{\underline{S}}$ a la ecuación anterior, obteniendo

$$\underline{\underline{S}}\underline{u} = \underline{\underline{S}}\underline{\underline{S}}^{-1}\underline{v} \quad (6.116)$$

simplificando, tenemos que

$$\underline{\underline{S}}\underline{u} = \underline{v}, \quad (6.117)$$

así, mediante algún procedimiento directo u iterativo (usando factorización LU o CGM) resolvemos el sistema anterior.

6.4.4 Cálculo de los Nodos Interiores

La evaluación de

$$\underline{u}_{\Pi} = - \left(\underline{\underline{A}}_{\Pi\Pi} \right)^{-1} \underline{\underline{A}}_{\Pi\Delta} \underline{u}_{\Delta} \quad (6.118)$$

involucra nuevamente cálculos locales de la expresión

$$\underline{u}_I^{\alpha} = - \left(\underline{\underline{A}}_{\Pi\Pi}^{\alpha} \right)^{-1} \underline{\underline{A}}_{\Pi\Delta}^{\alpha} \underline{u}_{\Gamma}^{\alpha} \quad (6.119)$$

en esta está nuevamente involucrado $\left(\underline{\underline{A}}_{\Pi\Pi}^{\alpha} \right)^{-1}$, por ello deberemos de usar el siguiente procedimiento para evaluar eficientemente esta expresión, realizando las operaciones equivalentes

$$\begin{aligned} \underline{x}_4 &= \underline{\underline{A}}_{\Pi\Delta}^{\alpha} \underline{u}_{\Gamma}^{\alpha} \\ \underline{u}_I^{\alpha} &= \left(\underline{\underline{A}}_{\Pi\Pi}^{\alpha} \right)^{-1} \underline{x}_4 \end{aligned} \quad (6.120)$$

multiplicando por $\underline{\underline{A}}_{\Pi\Pi}^{\alpha}$ a la última expresión, obtenemos

$$\underline{\underline{A}}_{\Pi\Pi}^{\alpha} \underline{u}_I^{\alpha} = \underline{\underline{A}}_{\Pi\Pi}^{\alpha} \left(\underline{\underline{A}}_{\Pi\Pi}^{\alpha} \right)^{-1} \underline{x}_4 \quad (6.121)$$

simplificando, tenemos

$$\underline{A}_{\text{III}}^\alpha \underline{u}_I^\alpha = \underline{x}_4 \quad (6.122)$$

esta última expresión puede ser resuelta usando Factorización LU o Gradiente Conjugado.

Como se observó, para resolver el sistema $\underline{A}_{\text{III}}^\alpha \underline{x} = \underline{b}$ podemos usar Factorización LU, Gradiente Conjugado o cualquier otro método para resolver sistemas lineales, pero deberá de usarse aquel que proporcione la mayor velocidad en el cálculo o que consuma la menor cantidad de memoria (ambas condicionantes son mutuamente excluyentes), por ello la decisión de qué método usar deberá de tomarse al momento de tener que resolver un problema particular en un equipo dado y básicamente el condicionante es el tamaño de la matriz $\underline{A}_{\text{III}}^\alpha$.

Para usar el método de Factorización LU, se deberá primeramente de factorizar la matriz bandada $\underline{A}_{\text{III}}^\alpha$ en una matriz \underline{LU} , la cual es bandada pero incrementa el tamaño de la banda a más del doble, pero esta operación sólo se deberá de realizar una vez por cada subdominio, y para solucionar los diversos sistemas lineales $\underline{A}_{\text{III}}^\alpha \underline{x} = \underline{b}$ sólo será necesario evaluar los sistemas

$$\begin{aligned} \underline{L}\underline{y} &= \underline{b} \\ \underline{U}\underline{x} &= \underline{y} \end{aligned} \quad (6.123)$$

en donde \underline{y} es un vector auxiliar. Esto proporciona una manera muy eficiente de evaluar el sistema lineal pero el consumo en memoria para un problema particular puede ser excesivo.

Por ello, si el problema involucra una gran cantidad de nodos interiores y el equipo en el que se implantará la ejecución del programa tiene una cantidad de memoria muy limitada, es recomendable usar el método de Gradiente Conjugado, este consume una cantidad de memoria adicional muy pequeña y el tiempo de ejecución se optimiza versus la Factorización LU.

De esta forma, es posible adaptar el código para tomar en cuenta la implementación de este en un equipo de cómputo en particular y poder sacar el máximo provecho al método de Subestructuración en la resolución de problemas elípticos de gran envergadura.

En lo que resta del presente trabajo, se asume que el método empleado para resolver $\underline{A}_{\text{III}}^\alpha \underline{x} = \underline{b}$ en sus respectivas variantes necesarias para evitar el cálculo de $\left(\underline{A}_{\text{III}}^\alpha\right)^{-1}$ es el método de Gradiente Conjugado, logrando así el máximo desempeño en velocidad en tiempo de ejecución.

FETI One-Level En el caso de FETI One-Level, al ser la matriz \underline{j} simétrica y positiva definida, la cual cumple también la restricción dada por la Ec. (6.49) ya que $\underline{j}\underline{u} = 0$, entonces la formulación silla de dicha ecuación, se reduce a:

Encontrar $(\underline{u}, \underline{\lambda}) \in W \times U$ tal que

$$\begin{cases} \underline{S}\underline{u} + \underline{j}\underline{\lambda} = \underline{f} \\ \underline{j}\underline{u} = \underline{0} \end{cases} \quad (6.124)$$

la solución $\underline{\lambda}$ de la Ec. (6.124) es única salvo la adición de un elemento del *Kernel* $\left(\underline{j}\right)$. El espacio de multiplicadores de Lagrange U , es por lo tanto elegido como el *Rango* $\left(\underline{j}\right)$. Este espacio puede ser entendido como el espacio de las funciones de salto en W .

El método One-Level FETI es un método de Gradiente Conjugado precondicionado en el espacio V , aplicado a

$$\underline{P}^T \underline{F}\underline{\lambda} = \underline{P}^T \underline{d}, \quad \underline{\lambda} \in \underline{\lambda}_0 + V \quad (6.125)$$

con condición inicial aproximada $\underline{\lambda}_0$ escogido tal que $\underline{G}\underline{\lambda}_0 = \underline{e}$. Con el precondicionador más básico para FETI al tomar $\underline{Q} = \underline{I}$, tiene la forma

$$\underline{M}^{-1} = \underline{j}\underline{S}\underline{j} = \sum_{i=1}^E \underline{j}^i \underline{S}^i \underline{j}^i \quad (6.126)$$

donde

$$\underline{F} = \underline{j}\underline{S}^\dagger \underline{j}^T, \quad \underline{G} = \underline{j}\underline{R}, \quad \underline{d} = \underline{j}\underline{S}^\dagger \underline{f}, \quad \underline{e} = \underline{R}^T \underline{f} \quad (6.127)$$

y

$$\underline{P}^T = \underline{I} - \underline{G} \left(\underline{G}^T \underline{Q}\underline{G} \right)^{-1} \underline{G}^T \underline{Q} \quad (6.128)$$

$$\underline{P} = \underline{I} - \underline{Q}\underline{G} \left(\underline{G}^T \underline{Q}\underline{G} \right)^{-1} \underline{G}^T. \quad (6.129)$$

Entonces el método One-Level FETI usando la matriz \underline{j} en vez de la matriz \underline{B} queda en términos del método de Gradiente Conjugado precondicionado como a continuación se muestra:

1.- Inicializa

$$\begin{aligned}\underline{\lambda}_0 &= \underline{Q}\underline{G} \left(\underline{G}^T \underline{Q}\underline{G} \right)^{-1} \underline{e} + \underline{\mu}, \underline{\mu} \in \text{Rango}(\underline{P}) \\ \underline{r}_0 &= \underline{d} - \underline{F}\underline{\lambda}^0 \\ \underline{\beta}^0 &= 0 \\ \underline{p}^1 &= 0\end{aligned}$$

2.- Itera $k = 1, 2, \dots$ hasta converger

$$\begin{aligned}\underline{q}^{k-1} &= \underline{P}^T \underline{r}^{k-1} \\ \underline{z}^{k-1} &= \underline{\hat{M}}^{-1} \underline{q}^{k-1} \\ \underline{y}^{k-1} &= \underline{P}\underline{z}^{k-1} \\ \underline{\beta}^k &= \frac{\langle \underline{y}^{k-1}, \underline{q}^{k-1} \rangle}{\langle \underline{y}^{k-2}, \underline{q}^{k-2} \rangle} \\ \underline{p}^k &= \underline{y}^{k-1} + \underline{\beta}^k \underline{p}^{k-1} \\ \underline{\alpha}^k &= \frac{\langle \underline{y}^{k-1}, \underline{q}^{k-1} \rangle}{\langle \underline{p}^k, \underline{F}\underline{p}^k \rangle} \\ \underline{\lambda}^k &= \underline{\lambda}^{k-1} + \underline{\alpha}^k \underline{p}^k \\ \underline{r}^k &= \underline{r}^{k-1} - \underline{\alpha}^k \underline{F}\underline{p}^k\end{aligned}$$

Así, una vez calculando el multiplicador de Lagrange $\underline{\lambda}$ obtenemos la solución en los nodos de la frontera interior \underline{u} mediante

$$\underline{u} = \underline{S}^\dagger \left(\underline{f} - \underline{j}\underline{\lambda} \right) \quad (6.130)$$

y para obtener la solución en los nodos interiores de cada subdominio se recurre a la aplicación de la Ec (19.73) del método de subestructuración o complemento de Schur, solucionando así el problema.

FETI One-Level Simplificado En el caso de FETI One-Level simplificado se toma a $\underline{R} = \underline{0}$, se reduce a:

El método One-Level FETI es un método de Gradiente Conjugado precondicionado en el espacio V , aplicado a

$$\underline{F}\underline{\lambda} = \underline{d}, \quad (6.131)$$

con el preconditionador

$$\underline{M}^{-1} = \underline{j}\underline{S}\underline{j} = \sum_{i=1}^E \underline{j}^i \underline{S}^i \underline{j}^i \quad (6.132)$$

donde

$$\underline{\underline{F}} = \underline{\underline{j}} \underline{\underline{S}}^\dagger \underline{\underline{j}}^T, \quad \underline{\underline{d}} = \underline{\underline{j}} \underline{\underline{S}}^\dagger \underline{\underline{f}}. \quad (6.133)$$

Entonces el método One-Level FETI simplificado queda en términos del método de Gradiente Conjugado preconditionado como a continuación se muestra:

1.- Inicializa

$$\begin{aligned} \underline{\lambda}_0 &= \underline{0} \\ \underline{r}_0 &= \left(\underline{\underline{j}} \underline{\underline{S}}^\dagger \underline{\underline{f}} \right) - \left(\underline{\underline{j}} \underline{\underline{S}}^\dagger \underline{\underline{j}} \right) \underline{\lambda}^0 \\ \underline{\beta}^0 &= 0 \\ \underline{p}^1 &= 0 \end{aligned}$$

2.- Itera $k = 1, 2, \dots$ hasta converger

$$\begin{aligned} \underline{q}^{k-1} &= \underline{r}^{k-1} \\ \underline{z}^{k-1} &= \left(\underline{\underline{j}} \underline{\underline{S}} \underline{\underline{j}} \right)^{-1} \underline{q}^{k-1} \\ \underline{y}^{k-1} &= \underline{z}^{k-1} \\ \underline{\beta}^k &= \frac{\langle \underline{y}^{k-1}, \underline{q}^{k-1} \rangle}{\langle \underline{y}^{k-2}, \underline{q}^{k-2} \rangle} \\ \underline{p}^k &= \underline{y}^{k-1} + \underline{\beta}^k \underline{p}^{k-1} \\ \underline{\alpha}^k &= \frac{\langle \underline{y}^{k-1}, \underline{q}^{k-1} \rangle}{\langle \underline{p}^k, \left(\underline{\underline{j}} \underline{\underline{S}}^\dagger \underline{\underline{j}} \right) \underline{q}^k \rangle} \\ \underline{\lambda}^k &= \underline{\lambda}^{k-1} + \underline{\alpha}^k \underline{p}^k \\ \underline{r}^k &= \underline{r}^{k-1} - \underline{\alpha}^k \left(\underline{\underline{j}} \underline{\underline{S}}^\dagger \underline{\underline{j}} \right) \underline{p}^k \end{aligned}$$

Así, una vez calculando el multiplicador de Lagrange $\underline{\lambda}$ obtenemos la solución en los nodos de la frontera interior \underline{u} mediante

$$\underline{u} = \underline{\underline{S}}^\dagger \left(\underline{f} - \underline{j} \underline{\lambda} \right) \quad (6.134)$$

y para obtener la solución en los nodos interiores de cada subdominio se recurre a la aplicación de la Ec (19.73) del método de subestructuración o complemento de Schur, solucionando así el problema.

FETI Dual-Primal En el caso de Dual-Primal, al ser la matriz \underline{j} simétrica y positiva definida, la cual cumple también la restricción dada por la Ec. (6.90) ya que $\underline{j}u = 0$, entonces la formulación silla de dicha ecuación, se reduce a:

Encontrar $\underline{u}_\Delta \in \tilde{W}$ tal que

$$J(u_\Delta) = \frac{1}{2} \left\langle \underline{\tilde{S}}u_\Delta, u_\Delta \right\rangle - \left\langle \underline{\tilde{f}}_\Delta, u_\Delta \right\rangle \rightarrow \min \left. \vphantom{J(u_\Delta)} \right\} \quad (6.135)$$

$$\underline{j}_\Delta u_\Delta = 0$$

la matriz \underline{j}_Δ es construida tal que los valores de la solución \underline{u}_Δ asociada a más de un subdominio coincida cuando $\underline{j}_\Delta u_\Delta = 0$.

Así, el método FETI-DP es un método de Gradiente Conjugado preconditionado para resolver el sistema preconditionado

$$\underline{M}^{-1} \underline{F} \lambda = \underline{M}^{-1} \underline{d} \quad (6.136)$$

con el preconditionador

$$\begin{aligned} \underline{M}^{-1} &= \underline{j}_{D,\Delta} \underline{S}_\Delta \underline{j}_{D,\Delta} \quad (6.137) \\ &= \sum_{i=1}^E \underline{D}_\Delta^i \underline{j}_\Delta^i \underline{S}_\Delta^i \underline{j}_\Delta^i \underline{D}_\Delta^i \end{aligned}$$

donde \underline{S}_Δ^i es la restricción del complemento de Schur local \underline{S}_Δ^i a $\tilde{W}_{\Delta,i} \subset W^i$ y

$$\underline{F} = \underline{j}_\Delta \left(\underline{\tilde{S}} \right)^{-1} \underline{j}_\Delta, \quad \underline{d} = \underline{j}_\Delta \left(\underline{\tilde{S}} \right)^{-1} \underline{\tilde{f}}_\Delta \quad (6.138)$$

$$\underline{j}_{D,\Delta} = \left[\underline{D}_\Delta^1 \underline{j}_{D,\Delta}^1, \dots, \underline{D}_\Delta^E \underline{j}_{D,\Delta}^E \right] \quad (6.139)$$

$$\underline{\tilde{S}} = \underline{A}_{\Delta\Delta} - \left[\begin{array}{cc} \left(\underline{A}_{I\Delta} \right)^T & \left(\underline{A}_{\Pi\Delta} \right)^T \end{array} \right] \left[\begin{array}{cc} \underline{A}_{II} & \underline{A}_{I\Pi} \\ \underline{A}_{\Pi I} & \underline{A}_{\Pi\Pi} \end{array} \right]^{-1} \left[\begin{array}{c} \underline{A}_{I\Delta} \\ \underline{A}_{\Pi\Delta} \end{array} \right] \quad (6.140)$$

$$\underline{\tilde{f}}_\Delta = \underline{f}_\Delta - \left[\begin{array}{cc} \left(\underline{A}_{I\Delta} \right)^T & \left(\underline{A}_{\Pi\Delta} \right)^T \end{array} \right] \left[\begin{array}{cc} \underline{A}_{II} & \underline{A}_{I\Pi} \\ \underline{A}_{\Pi I} & \underline{A}_{\Pi\Pi} \end{array} \right]^{-1} \left[\begin{array}{c} \underline{f}_I \\ \underline{f}_\Pi \end{array} \right] \quad (6.141)$$

El método FETI-DP queda en términos del método de Gradiente Conjugado preconditionado como a continuación se muestra:

1.- Inicializa

$$\begin{aligned}\underline{r}^0 &= \underline{d} - \underline{F}\underline{\lambda}^0 \\ \underline{\beta}^1 &= 0 \\ \underline{p}^1 &= \underline{z}^0\end{aligned}$$

2.- Itera $k = 1, 2, \dots$ hasta converger

$$\begin{aligned}\underline{z}^{k-1} &= \underline{M}^{-1}\underline{r}^{k-1} \\ \underline{\beta}^k &= \frac{\langle \underline{z}^{k-1}, \underline{r}^{k-1} \rangle}{\langle \underline{z}^{k-2}, \underline{r}^{k-2} \rangle} \\ \underline{p}^k &= \underline{z}^{k-1} + \underline{\beta}^k \underline{p}^{k-1} \\ \underline{\alpha}^k &= \frac{\langle \underline{z}^{k-1}, \underline{r}^{k-1} \rangle}{\langle \underline{p}^k, \underline{F}\underline{q}^k \rangle} \\ \underline{\lambda}^k &= \underline{\lambda}^{k-1} + \underline{\alpha}^k \underline{p}^k \\ \underline{r}^k &= \underline{r}^{k-1} - \underline{\alpha}^k \underline{F}\underline{p}^k\end{aligned}$$

Así, una vez calculando el multiplicador de Lagrange $\underline{\lambda}$ obtenemos la solución en los nodos de la frontera interior \underline{u}_Δ mediante

$$\underline{u}_\Delta = \left(\underline{\tilde{S}} \right)^{-1} \left(\underline{\tilde{f}}_\Delta - \left(\underline{B}_\Delta \right)^T \underline{\lambda} \right) \quad (6.142)$$

y para obtener la solución en los nodos interiores de cada subdominio se recurre a la aplicación de la Ec (19.73) del método de subestructuración o complemento de Schur, solucionando así el problema.

6.5 Implementación Computacional

A partir de los modelos matemáticos y los modelos numéricos en esta sección se describe el modelo computacional contenido en un programa de cómputo orientado a objetos en el lenguaje de programación C++ en su forma secuencial y en su forma paralela en C++ usando la interfaz de paso de mensajes (MPI) bajo el esquema maestro-esclavo.

Esto no sólo nos ayudará a demostrar que es factible la construcción del propio modelo computacional a partir del modelo matemático y numérico para la solución de problemas reales. Además, se mostrará los alcances y limitaciones en el consumo de los recursos computacionales, evaluando algunas de las variantes de los métodos numéricos con los que es posible implementar

el modelo computacional y haremos el análisis de rendimiento sin llegar a ser exhaustivo esté.

También exploraremos los alcances y limitaciones de cada uno de los métodos implementados y como es posible optimizar los recursos computacionales con los que se cuenta.

Primeramente hay que destacar que el paradigma de programación orientada a objetos es un método de implementación de programas, organizados como colecciones cooperativas de objetos. Cada objeto representa una instancia de alguna clase y cada clase es miembro de una jerarquía de clases unidas mediante relaciones de herencia, contención, agregación o uso.

Esto nos permite dividir en niveles la semántica de los sistemas complejos tratando así con las partes, que son más manejables que el todo, permitiendo su extensión y un mantenimiento más sencillo. Así, mediante la herencia, contención, agregación o uso nos permite generar clases especializadas que manejan eficientemente la complejidad del problema. La programación orientada a objetos organiza un programa entorno a sus datos (atributos) y a un conjunto de interfaces bien definidas para manipular estos datos (métodos dentro de clases reusables) esto en oposición a los demás paradigmas de programación.

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

Para empezar con la implementación computacional, primeramente definiremos el problema a trabajar. Este, pese a su sencillez, no pierde generalidad permitiendo que el modelo mostrado sea usado en muchos sistemas de la ingeniería y la ciencia.

La implementación de los métodos a priori, requieren de más trabajo tanto en la fase de construcción como en la parte de su aplicación, la gran ventaja de este tipo de preconditionadores es que pueden ser óptimos, es decir, para ese problema en particular el preconditionador encontrado será el mejor preconditionador existente, llegando a disminuir el número de iteraciones hasta en un orden de magnitud.

El Operador de Laplace y la Ecuación de Poisson Consideramos como modelo matemático el problema de valor en la frontera (BVP) asociado con el operador de Laplace en dos dimensiones, el cual en general es usualmente referido como la ecuación de Poisson, con condiciones de frontera Dirichlet, definido en Ω como:

$$\begin{aligned} -\nabla^2 u + k^2 u &= f_\Omega \text{ en } \Omega \\ u &= g_{\partial\Omega} \text{ en } \partial\Omega. \end{aligned} \tag{6.143}$$

Se toma está ecuación para facilitar la comprensión de las ideas básicas. Es un ejemplo muy sencillo, pero gobierna los modelos de muchos sistemas de la ingeniería y de la ciencia.

En particular consideramos el problema con Ω definido en:

$$\Omega = [0, 1] \times [0, 1] \tag{6.144}$$

donde

$$f_\Omega = \exp(xy) \quad \text{y} \quad g_{\partial\Omega} = 0. \tag{6.145}$$

En todos los cálculos de los métodos numéricos usados para resolver el sistema lineal algebraico asociado se usó una tolerancia mínima de 1×10^{-5} .

A partir de la formulación del método de elemento finito visto en la sección (5.1.2), la implementación computacional que se desarrolló tiene la jerarquía de clases siguiente:

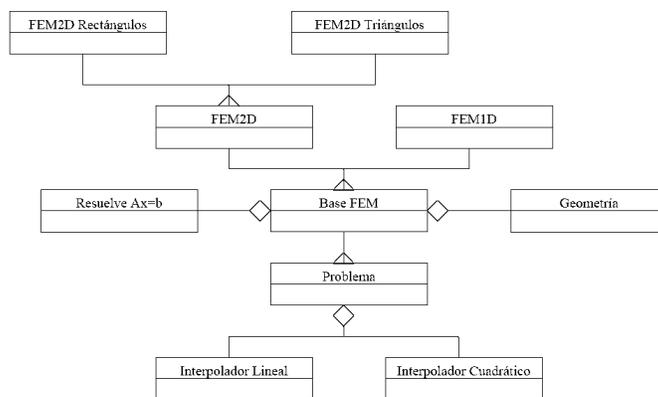


Figura 4: Jerarquía de clases para el método de elemento finito

Donde las clases participantes en *FEM2D Rectángulos* son:

La clase *Interpolador Lineal* define los interpoladores lineales usados por el método de elemento finito.

La clase *Problema* define el problema a tratar, es decir, la ecuación diferencial parcial, valores de frontera y dominio.

La clase *Base FEM* ayuda a definir los nodos al usar la clase *Geometría* y mantiene las matrices generadas por el método y a partir de la clase *Resuelve $Ax=B$* se dispone de diversas formas de resolver el sistema lineal asociado al método.

La clase *FEM2D* controla lo necesario para poder hacer uso de la geometría en 2D y conocer los nodos interiores y de frontera, con ellos poder montar la matriz de rigidez y ensamblar la solución.

La clase *FEM2D Rectángulos* permite calcular la matriz de rigidez para generar el sistema algebraico de ecuaciones asociado al método.

Notemos que esta misma jerarquía permite trabajar problemas en una y dos dimensiones, en el caso de dos dimensiones podemos discretizar usando rectángulos o triángulos, así como usar varias opciones para resolver el sistema lineal algebraico asociado a la solución de EDP.

Por otro lado, la computación en paralelo es una técnica que nos permite distribuir una gran carga computacional entre muchos procesadores. Y es bien sabido que una de las mayores dificultades del procesamiento en paralelo es la coordinación de las actividades de los diferentes procesadores y el intercambio de información entre los mismos.

Para hacer una adecuada coordinación de actividades entre los diferentes procesadores, el programa que soporta el método de subestructuración paralelo, usa la misma jerarquía de clases que el método de subestructuración secuencial. Este se desarrolló para usar el esquema maestro-esclavo, de forma tal que el nodo maestro mediante la agregación de un objeto de la clase de *Geometría* genere la descomposición gruesa del dominio y los nodos esclavos creen un conjunto de objetos *FEM2D Rectángulos* para que en estos objetos se genere la participación fina y mediante el paso de mensajes (vía MPI) puedan comunicarse los nodos esclavos con el nodo maestro.

La implementación computacional que se desarrolló tiene una jerarquía de clases en la cual se agregan las clases *FEM2D Rectángulos* y *Geometría*, además de heredar a la clase *Problema*. De esta forma se rehusó todo el código desarrollado para *FEM2D Rectángulos*, la jerarquía queda como:

La clase *DDM2D* realiza la partición gruesa del dominio mediante la clase *Geometría* y controla la partición de cada subdominio mediante un objeto de la clase de *FEM2D Rectángulos* generando la partición fina del dominio. La resolución de los nodos de la frontera interior se hace mediante el método de gradiente conjugado, necesaria para resolver los nodos internos de cada subdominio.

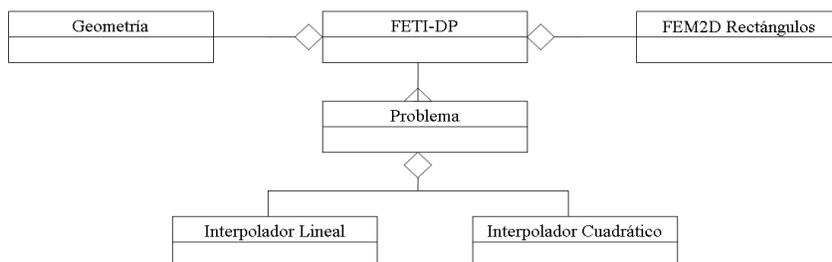


Figura 5: Jerarquía de clases para el método de subestructuración secuencial

Así, el dominio Ω es descompuesto en una descomposición gruesa de $n \times m$ subdominios y cada subdominio Ω_i se parte en $p \times q$ subdominios, generando la participación fina del dominio como se muestra en la figura:

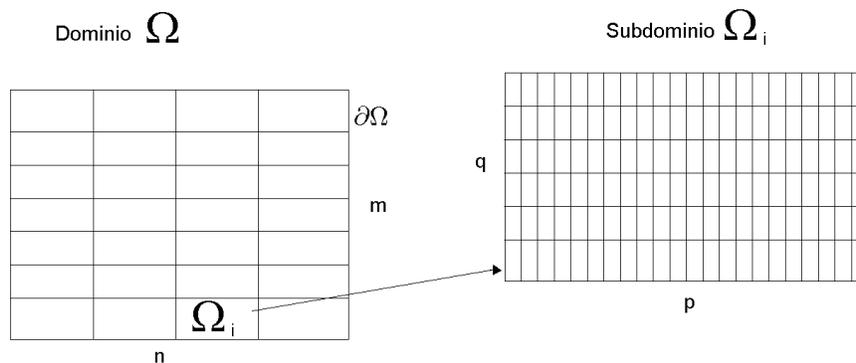


Figura 6: Descomposición del dominio Ω en $E = n \times m$ subdominios y cada subdominio Ω_i en $p \times q$ subdominios

Realizando las siguientes tareas:

A) El nodo maestro genera la descomposición gruesa del dominio (supongamos particionado en $n \times m$ subdominios) mediante la agregación de un objeto de la clase *Geometría*, esta geometría es pasada a los nodos esclavos.

B) Con esa geometría se construyen los objetos *FEM2D Rectángulos* (uno por cada subdominio), donde cada subdominio es particionado (supongamos en $p \times q$ subdominios). Cada objeto de *FEM2D Rectángulos* genera la geometría solicitada, regresando las coordenadas de los nodos de frontera del subdominio correspondiente al nodo maestro.

C) Con estas coordenadas, el nodo maestro conoce a los nodos de la frontera interior (son estos los que resuelve el método de descomposición de dominio). Las coordenadas de los nodos de la frontera interior se dan a conocer a los objetos *FEM2D Rectángulos* en los nodos esclavos, transmitiendo sólo aquellos que están en su subdominio.

D) Después de conocer los nodos de la frontera interior, cada objeto *FEM2D Rectángulos* calcula las matrices

$$\underline{A}_{II}^i, \underline{A}_{I\Pi}^i, \underline{A}_{I\Delta}^i, \underline{A}_{\Pi\Pi}^i, \underline{A}_{\Pi\Delta}^i \text{ y } \underline{A}_{\Delta\Delta}^i$$

necesarias para construir el complemento de Schur local

$$\tilde{\underline{S}}^i = \underline{A}_{\Delta\Delta}^i - \left[\begin{array}{cc} \left(\underline{A}_{I\Delta}^i\right)^T & \left(\underline{A}_{\Pi\Delta}^i\right)^T \end{array} \right] \left[\begin{array}{cc} \underline{A}_{II}^i & \underline{A}_{I\Pi}^i \\ \left(\underline{A}_{I\Pi}^i\right)^T & \underline{A}_{\Pi\Pi}^i \end{array} \right]^{-1} \left[\begin{array}{c} \underline{A}_{I\Delta}^i \\ \underline{A}_{\Pi\Delta}^i \end{array} \right]$$

$$\tilde{\underline{f}}_{\Delta}^i = \underline{f}_{\Delta}^i - \left[\begin{array}{cc} \left(\underline{A}_{I\Delta}^i\right)^T & \left(\underline{A}_{\Pi\Delta}^i\right)^T \end{array} \right] \left[\begin{array}{cc} \underline{A}_{II}^i & \underline{A}_{I\Pi}^i \\ \left(\underline{A}_{I\Pi}^i\right)^T & \underline{A}_{\Pi\Pi}^i \end{array} \right]^{-1} \left[\begin{array}{c} \underline{f}_{I}^i \\ \underline{f}_{\Pi}^i \end{array} \right]$$

sin realizar comunicación alguna. Al terminar de calcular las matrices se avisa al nodo maestro de la finalización de los cálculos.

E) Mediante la comunicación de vectores del tamaño del número de nodos de la frontera interior entre el nodo maestro y los objetos *FEM2D Rectángulos*, se prepara todo lo necesario para empezar el método de gradiente conjugado y resolver el sistema lineal virtual $\underline{M}^{-1}\underline{F}\lambda = \underline{M}^{-1}d$.

F) Para usar el método de gradiente conjugado, se transmite un vector del tamaño del número de nodos de la frontera interior para que en cada objeto se realicen las operaciones pertinentes y resolver así el sistema algebraico asociado, esta comunicación se realiza de ida y vuelta entre el nodo maestro y los objetos *FEM2D Rectángulos* tantas veces como iteraciones haga el método. Resolviendo con esto los nodos de la frontera interior \underline{u}_{Σ_i} .

G) Al término de las iteraciones se pasa la solución $\underline{\lambda}^i$ de los nodos de la frontera interior que pertenecen a cada subdominio dentro de cada objeto *FEM2D Rectángulos* para que se resuelvan los nodos interiores $\underline{u}_{\Delta}^i = \left(\underline{\tilde{S}}^i\right)^{-1} \left(\underline{\tilde{f}}_{\Delta}^i - \left(\underline{B}_{\Delta}^i\right)^T \underline{\lambda}^i\right)$, sin realizar comunicación alguna en el proceso, al concluir se avisa al nodo maestro de ello.

I) El nodo maestro mediante un último mensaje avisa que se concluya el programa, terminado así el esquema maestro-esclavo.

Del algoritmo descrito anteriormente hay que destacar la sincronía entre el nodo maestro y los objetos *FEM2D Rectángulos* contenidos en los nodos esclavos, esto es patente en las actividades realizadas en los incisos A, B y C, estas consumen una parte no significativa del tiempo de cálculo.

Una parte importante del tiempo de cálculo es consumida en la generación de las matrices locales descritas en el inciso D que se realizan de forma independiente en cada nodo esclavo, esta es muy sensible a la discretización particular del dominio usado en el problema.

Los incisos E y F del algoritmo consumen la mayor parte del tiempo total de ejecución al resolver el sistema lineal que dará la solución a los nodos de la frontera interior. La resolución de los nodos interiores planteada en el inciso G consume muy poco tiempo de ejecución, ya que sólo se realiza una serie de cálculos locales previa transmisión del vector que contiene la solución a los nodos de la frontera interior.

Este algoritmo es altamente paralelizable ya que los nodos esclavos están la mayor parte del tiempo ocupados y la fracción serial del algoritmo está principalmente en las actividades que realiza el nodo maestro, estas nunca podrán ser eliminadas del todo pero consumirá menos tiempo del algoritmo conforme se haga más fina la malla en la descomposición del dominio.

Por ejemplo, para resolver la Ec. (6.143), usando 3072×3072 nodos podemos tomar alguna de las siguientes descomposiciones:

Introducción a los Métodos de Descomposición de Dominio

Descomposición	Subdominios	Nodos Interiores	Elementos Subdominio	Total Nodos Subdominio	Nodos Desconocidos Subdominio
8×8 y 384×384	64	147456	148225	146689	9388096
16×16 y 192×192	96	36864	37249	36481	9339136
32×32 y 96×96	1024	9216	9409	9025	9241600
64×64 y 48×48	4096	2304	2401	2409	9048064
128×128 y 24×24	16384	576	625	529	8667136

Cada una de las descomposiciones genera un problema distinto. Usando el equipo secuencial a 2.8 GHz y evaluando el desempeño del método de subestructuración secuencial se obtuvieron los siguientes resultados:

Partición	Nodos Frontera Interior	Iteraciones	Tiempo
		Subestructuración	Subestructuración
8×8 y 384×384	42945	4	18071 seg.
16×16 y 192×192	91905	3	4751 seg.
32×32 y 96×96	189441	2	911 seg.
64×64 y 48×48	382977	1	781 seg.
128×128 y 24×24	76395	1	3130 seg.

y para el método FETI-DP secuencial se obtuvieron los siguientes resultados:

Partición	Nodos Frontera Interior	Iteraciones	Tiempo
		FETI-DP	FETI-DP
8×8 y 384×384	42945	2	14685 seg.
16×16 y 192×192	91905	2	3985 seg.
32×32 y 96×96	189441	1	777 seg.
64×64 y 48×48	382977	1	673 seg.
128×128 y 24×24	76395	1	2977 seg.

Nótese que aún en un solo procesador es posible encontrar una descomposición que disminuya los tiempos de ejecución (la descomposición de 64×64 y 48×48 concluye en 673 seg. versus los 781 seg. en el caso del algoritmo de subestructuración).

Notemos también que en la última descomposición, en lugar de disminuir el tiempo de ejecución este aumenta, esto se debe a que se construyen muchos objetos *FEM2D Rectángulos* (76395 en este caso), con los cuales hay que hacer comunicación resultando muy costoso computacionalmente.

Por otro lado, para la implementación paralela, la descomposición adecuada del dominio para tener un buen balanceo de cargas se logra cuando se descompone en $n \times m$ nodos en la partición gruesa, generándose $n * m$ subdominios y si se trabaja con P procesadores (1 para el nodo maestro y $P - 1$ para los nodos esclavos), entonces el balance de cargas adecuado será cuando $(P - 1) \mid (n * m)$. Así, los siguientes tiempos fueron obtenidos al usar 1,2,3,5,9 y 17 procesadores.

Usando para los cálculos en un procesador el equipo secuencial y para la parte paralela el cluster homogéneo a 2.8 GHz resolviendo por el método de gradiente conjugado, la solución para una partición 64×64 y 48×48 se encontró la solución en 1 iteración en los siguientes tiempos:

Partición	CPUs	Tiempo Total
64×64 y 48×48	1	673 seg.
64×64 y 48×48	2	820 seg.
64×64 y 48×48	3	415 seg.
64×64 y 48×48	5	286 seg.
64×64 y 48×48	9	222 seg.
64×64 y 48×48	17	190 seg.

Las métricas de desempeño son las siguientes

Procesadores	Tiempo	Factor de Aceleración	Eficiencia	Fracción Serial
1	673 seg.			
2	820 seg.	0.8207	0.41036	1.43684
3	409 seg.	1.6216	0.54056	0.42496
5	399 seg.	2.3531	0.47062	0.28120
9	353 seg.	3.0315	0.33683	0.24609
17	330 seg.	3.5421	0.20835	0.23746

Estos resultados pueden ser apreciados mejor de manera gráfica como se muestra a continuación:

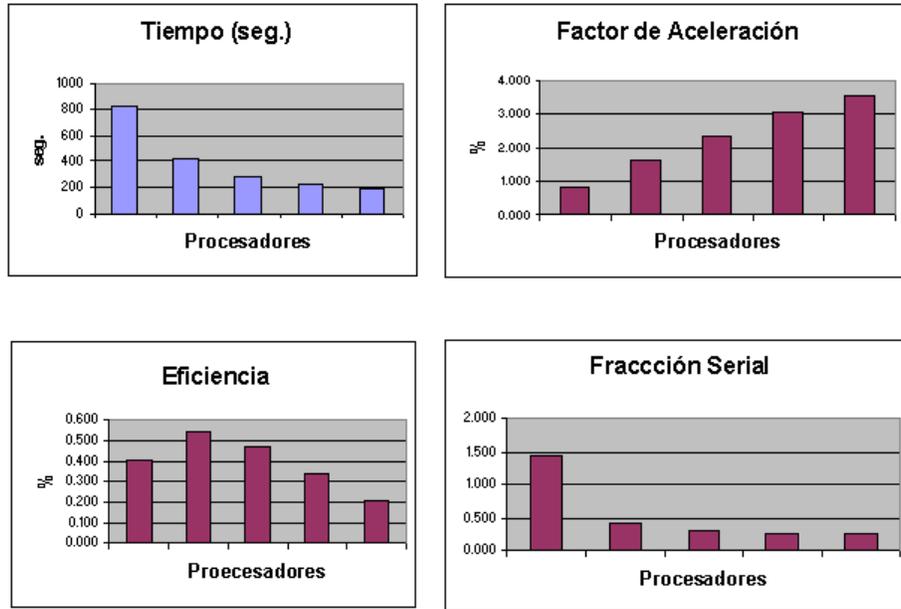


Figura 7: Métricas de desempeño mostrando sólo cuando las cargas están bien balanceadas (2, 3, 5, 9 y 17 procesadores).

En cuanto a las métricas de desempeño, obtenemos que el factor de aceleración en el caso ideal debería de aumentar de forma lineal al aumento del número de procesadores, que en nuestro caso no es lineal pero cumple bien este hecho si están balanceadas las cargas de trabajo.

El valor de la eficiencia deberá ser cercano a uno cuando el Hardware es usado de manera eficiente, como es en nuestro caso cuando se tiene un procesador por cada subdominio.

Y en la fracción serial su valor debería de tender a cero en el caso ideal, siendo este nuestro caso si están balanceadas las cargas de trabajo, de aquí se puede concluir que la granularidad del problema es gruesa, es decir, no existe una sobrecarga en los procesos de comunicación siendo el cluster una buena herramienta de trabajo para este tipo de problemas.

Finalmente las posibles mejoras de eficiencia para el método de subestructuración en paralelo para disminuir el tiempo de ejecución pueden ser:

- Balance de cargas de trabajo homogéneo.
- Al compilar los códigos usar directivas de optimización.

- Usar bibliotecas que optimizan las operaciones en el manejo de los elementos de la matriz usando punteros en las matrices densas o bandadas.
- El cálculo de las matrices que participan en el complemento de Schur pueden ser obtenidas en paralelo.

Un comentario que considero pertinente hacer, es con respecto a la comparación del número de iteraciones que se requiere para concluir un problema dado por ejemplo entre los métodos FETI-DP y por el Complemento de Schur, ya que por ejemplo, tomando una partición de 10×10 y cada subdominio en 10×10 tenemos:

Para el método de Complemento de Schur:

El CGM resuelve para 1701 nodos de frontera interior (1701 grados de libertad) en 38 iteraciones.

Para el método de FETI-DP

El CGM resuelve para 1701 nodos de frontera interior, pero con 3564 multiplicadores de Lagrange (3564 grados de libertad) en 79 iteraciones.

Por ello hay que tomar en cuenta que el número de iteraciones pese a que resuelven el mismo problema, no son comparables por los distintos grados de libertad que se resuelven en el sistema lineal, pero el tiempo de ejecución si deberá de usarse para comparar los métodos FETI-DP y Schur para una partición dada de un problema determinado.

7 Funciones Definidas por Tramos

Sea $\Omega \subset \mathbb{R}^n$ un dominio, y $\Pi = \{\Omega_1, \dots, \Omega_E\}$ una partición o descomposición en subdominios Ω_i sin traslape del dominio Ω -también conocida como malla gruesa \mathcal{T}_H . Un ejemplo de un dominio Ω y su descomposición en subdominios Ω_i y cada Ω_i a su vez descompuesto en Ω_e subdominios se muestra en la figura:

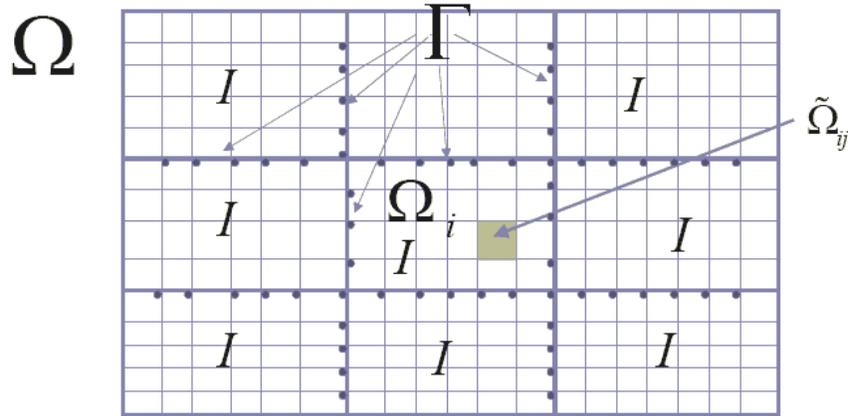


Figura 8: El dominio Ω , su frontera externa $\partial\Omega$ y la frontera interna Γ .

Se asume que:

- 1.- Ω_i , para $i = 1, \dots, E$ es un subdominio de Ω ,
- 2.- $\Omega_i \cap \Omega_j = \emptyset$, siempre que $i \neq j$.
- 3.- $\Omega \subset \bigcup_{i=1}^E \overline{\Omega}_i$.

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

$$\partial\Omega \subset \bigcup_{i=1}^E \partial\Omega_i. \quad (7.1)$$

Adicionalmente definimos a la frontera interior como

$$\Gamma = \bigcup_{i \neq j} \partial\Omega_i \cap \partial\Omega_j \quad (7.2)$$

y a $\partial\Omega$ como la frontera exterior del dominio Ω , denotamos por H al diámetro $H_i = \text{Diam}(\Omega_i)$ de cada Ω_i que satisface $\text{Diam}(\Omega_i) \leq H$ para cada $i = 1, 2, \dots, E$, además, cada subdominio Ω_i es descompuesto en un mallado fino \mathcal{T}_h de K subdominios mediante una triangulación Ω_e de modo que esta sea conforme, denotamos por h al diámetro $h_i = \text{Diam}(\Omega_e)$ de cada Ω_e que satisface $\text{Diam}(\Omega_e) \leq h$ para cada $e = 1, 2, \dots, K$ de cada $i = 1, 2, \dots, E$.

También asumimos que salvo en un conjunto de medida cero sobre Γ se define un único vector normal denotado por \underline{n} cuyo sentido se elige arbitrariamente y el lado positivo de Γ es definido hacia el sentido positivo del vector normal.

Sea $D(\Omega)$ un espacio lineal de funciones definidas en Ω , entonces

Definición 52 *Identificamos como una sola función, a dos funciones $u, w \in D(\Omega)$ cuyo dominio de definición está contenido en Ω , cuando se satisface la condición de que el conjunto de puntos en los cuales $u \neq w$ tiene medida de Lebesgue cero.*

Dada una partición $\Pi = \{\Omega_1, \dots, \Omega_E\}$ del dominio Ω , entonces

Definición 53 *Una función definida por pedazos es una sucesión de funciones $\{w_1, \dots, w_E\}$, tal que para cada $i = 1, \dots, E$, la función w_i está definida en Ω_i . Dada una función w definida en Ω esta tiene una única función definida por pedazos $\{w_1, \dots, w_E\}$, tal que*

$$w_i = w|_{\Omega_i} \quad \text{con } i = 1, \dots, E \quad (7.3)$$

donde $w|_{\Omega_i}$ es la restricción de w a Ω_i .

Esto establece una correspondencia biunívoca entre las funciones definidas en Ω y las funciones definidas por pedazos.

Definición 54 *Identificaremos a la función w definida en casi todos los lados salvo un conjunto de medida cero en Ω y la correspondiente sucesión $\{w_1, \dots, w_E\}$.*

Así, dada una función w contenida en Ω , la sucesión $\{w_1, \dots, w_E\}$ será referida como la representación definida por pedazos de w y las funciones $w_i, i = 1, \dots, E$ como componentes locales de w . También, establecemos una correspondencia biunívoca entre los espacios $\{D(\Omega_1), \dots, D(\Omega_E)\}$ y $\hat{D}(\Omega)$ en Ω .

Definición 55 Dada una familia $\{D(\Omega_1), \dots, D(\Omega_E)\}$ de espacios lineales definidos en $\Omega_1, \dots, \Omega_E$ respectivamente, definimos el espacio lineal $\hat{D}(\Omega)$ contenido en Ω por

$$\hat{D}(\Omega) \equiv D(\Omega_1) \oplus \dots \oplus D(\Omega_E). \quad (7.4)$$

Teorema 56 Sea $\{w_1, \dots, w_E\}$ la representación en pedazos de cualquier función w , entonces $w \in \hat{D}(\Omega)$ si y sólo si $w_i \in D(\Omega_i)$ para todo $i = 1, \dots, E$.

Definición 57 El espacio lineal de funciones definidas por pedazos $\hat{D}(\Omega)$ por el espacio cuyos elementos son la restricción a Ω_i de las funciones pertenecientes a $D(\Omega_i)$.

En cuyo caso la función de $D(\Omega)$ a $\hat{D}(\Omega)$ la cual asocia a cada $w \in D(\Omega)$ una representación en pedazos de

$$\{w_1, \dots, w_E\} \in D(\Omega_1) \oplus \dots \oplus D(\Omega_E) \quad (7.5)$$

es una biyección la cual es referida como la inmersión natural de $D(\Omega)$ sobre $D(\Omega_1) \oplus \dots \oplus D(\Omega_E)$. En lo sucesivo identificaremos los dos espacios lineales y escribiremos

$$D(\Omega) \subset D(\Omega_1) \oplus \dots \oplus D(\Omega_E). \quad (7.6)$$

En vista de las definiciones anteriores, para cada función $v \in \hat{D}(\Omega)$, existe una sucesión de funciones $\{v_1, \dots, v_E\}$ tal que $v_i = v|_{\Omega_i}; i = 1, \dots, E$, donde $v|_{\Omega_i}$ es la restricción de v a Ω_i .

Puesto que una función $v \in \hat{D}(\Omega)$ se forma por las restricciones de funciones definidas de manera independiente en cada subdominio Ω_i , esta es en general totalmente discontinua en la frontera interior Γ , así, las funciones que pertenecen a este espacio pueden tener discontinuidades de salto finitas tanto en el valor de la función como en el valor de sus derivadas normales en Γ .

Cuando se califica a una función como continua, se entiende que la función es continua en su valor, sin asumir nada acerca de la continuidad en sus derivadas. Aunque es claro que la función es continua en todo Ω , esto es, en cada subdominio Ω_i y en la frontera interior Γ , se pondrá énfasis en la continuidad a través de Γ . Cuando se califica a una función como totalmente continua, se entiende que la función es continua tanto en su valor como en sus derivadas normales a través de Γ . Cuando se califica a una función

como totalmente discontinua, se entiende que la función puede presentar discontinuidades tanto en su valor como en sus derivadas normales a través de Γ .

Sea $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j$ donde $\partial\Omega_i$ y $\partial\Omega_j$ son las fronteras de dos subregiones adyacentes, entonces definimos como la traza a la restricción de v^i a Γ_{ij} .

Pero como Γ_{ij} , para dos subregiones vecinas hay dos trazas definidas una que corresponde a v^i y otra a v^j , entonces se requiere introducir la siguiente notación para poderlas distinguir entre sí:

$$v_+ \equiv Tr(v^i) \quad (7.7)$$

cuando Ω_i cae del lado positivo de Γ_{ij} y

$$v_- \equiv Tr(v^j) \quad (7.8)$$

en caso contrario. Aquí $Tr(v)$ designa al operador traza de la función v . En general $v_+ \neq v_-$ ya que se trabaja con espacios de funciones definidas por tramos.

Observación 3 *Notemos que al considerar una función w en Ω , su definición en Γ es innecesaria, ya que la medida de Lebesgue de Γ es cero. Si la traza de w_α es definida en casi todos lados salvo un conjunto de medida cero sobre $\partial\Omega_\alpha$ para $\alpha = 1, \dots, E$, entonces tal traza es también definida en Γ . En particular, si la traza de w_α esta definida sobre $\partial\Omega_\alpha$ para cada $\alpha = 1, \dots, E$, entonces ellas definen dos funciones definidas en casi todos lados salvo un conjunto de medida cero sobre Γ , denotadas por (w_+, w_-) correspondientes a los lados de trazas positivas y negativas de Γ respectivamente.*

Definición 58 *El salto de v sobre Γ de funciones definidas por pedazos como*

$$[[w]] \equiv w_+ - w_- \quad (7.9)$$

y el promedio como

$$\dot{w} \equiv \frac{1}{2}(w_+ + w_-) \quad (7.10)$$

respectivamente.

Observemos que tanto el promedio de una función \dot{w} , cómo el producto $[w] \cdot \underline{n}$ no depende de como se elija el sentido del vector normal unitario \underline{n} en Γ , además las siguientes identidades se satisfacen

$$w_+ = \dot{w} + \frac{1}{2} [[w]] \quad \text{y} \quad w_- = \dot{w} - \frac{1}{2} [[w]]. \quad (7.11)$$

La discontinuidad de una función en la frontera interior Γ se puede expresar ya sea especificando los valores de sus trazas en Γ , o bien, especificando los valores de su promedio y de su salto en Γ . Además, si la función v es continua a través de Γ se tiene que

$$v = v_+ = v_- = \dot{v} \quad \text{y que} \quad [[v]] = 0.$$

Por otro lado tenemos que si u y v son funciones definidas en Ω , entonces tenemos que

$$[[uv]] = \dot{u} [[v]] + \dot{v} [[u]] \quad \text{en } \Gamma.$$

7.1 Espacios de Sobolev de Funciones Definidas por Tramos

Dada una familia de espacios lineales $\{D(\Omega_1), \dots, D(\Omega_E)\}$, tal que $D(\Omega_i)$, para cada $i = 1, 2, \dots, E$, es un espacio lineal de funciones definido en casi todos lados salvo un conjunto de medida cero en Ω_i , se puede considerar el espacio

$$D(\Omega) = D(\Omega_1) \oplus \dots \oplus D(\Omega_E) \quad (7.12)$$

entonces, los elementos de $D(\Omega)$ son funciones definidas por tramos, (w_1, \dots, w_E) , con $w_i \in D(\Omega_i)$, $i = 1, 2, \dots, E$.

Definición 59 *El espacio de Sobolev de orden $p \geq 0$ para funciones definidas por tramos está dado por*

$$\hat{H}^p(\Omega, \Pi) = H^p(\Omega_1) \oplus \dots \oplus H^p(\Omega_E) \quad (7.13)$$

aquí, $H^p(\Omega_i)$ es el espacio de Sobolev de orden p , de funciones definidas en Ω_i . Cada función $w \in \hat{H}^p(\Omega)$ es una sucesión, $w \equiv (w_1, \dots, w_E)$, con $w_i \in H^p(\Omega_i)$, $i = 1, 2, \dots, E$.

Observemos que cuando $w \in H^p(\Omega)$, entonces la restricción, w_i de w a Ω_i tiene la propiedad que $w_i \in H^p(\Omega_i)$. Por lo tanto

$$H^p(\Omega) \subset \hat{H}^p(\Omega) \quad (7.14)$$

Para $p > 0$, esta es una inclusión propia. Sin embargo para $p = 0$, $H^0(\Omega) \equiv \hat{H}^0(\Omega) \equiv L^2(\Omega)$. Además

$$H^0(\Omega) \equiv \hat{H}^0(\Omega) \supset \hat{H}^p(\Omega) \quad \forall p \geq 0 \quad (7.15)$$

Aquí las funciones definidas en Ω han sido identificadas con sus representaciones por partes. Todos los espacios $\hat{H}^p(\Omega)$, para $p = 0, 1, 2, \dots$, están hechos de funciones las cuales pertenecen a $H^0(\Omega) \equiv L^2(\Omega)$.

Teorema 60 Para cada $p \geq 0$, una función $\hat{u} = (u_1, \dots, u_E) \in H^0(\Omega)$ pertenecen a $\hat{H}^p(\Omega)$ si y sólo si la norma

$$\|\hat{v}\|_{p,\Omega,\Pi} = \left(\sum_{i=1}^E \|v_i\|_{p,\Omega_i}^2 \right)^{\frac{1}{2}} \quad (7.16)$$

está bien definida.

Cuando $\hat{H}^p(\Omega)$ es equipada con esta norma el correspondiente producto interior (\cdot, \cdot) , se convierte en un espacio de Hilbert.

Observación 4 Las siguientes propiedades se satisfacen:

1.- Cuando $w \in H^p(\Omega)$, entonces la restricción de w a Ω_α , w_α tiene la propiedad de que $w_\alpha \in H^p(\Omega_p)$. Por lo tanto

$$H^p(\Omega) \subset \hat{H}^p(\Omega) \quad (7.17)$$

2.- Cuando $u \in \hat{H}^1(\Omega)$ entonces

$$[[u]] = 0 \text{ sobre } \Gamma \Leftrightarrow u \in H^1(\Omega) \quad (7.18)$$

3.- Cuando $u \in \hat{H}^2(\Omega)$ entonces

$$[[u]] = 0 \text{ y } \left[\left[\frac{\partial u}{\partial n} \right] \right] = 0 \text{ sobre } \Gamma \Leftrightarrow u \in H^2(\Omega). \quad (7.19)$$

La identidad

$$\begin{aligned} \sum_{\alpha=1}^E \int_{\partial\Omega_\alpha} u_\alpha w_\alpha n_i dx &= \int_{\partial\Omega} u w n_i dx - \int_{\Gamma} (u w) n_i dx \\ &= \int_{\partial\Omega} u w n_i dx - \int_{\Gamma} \left(\dot{u} [[w]] + \dot{w} [[u]] \right) n_i dx \end{aligned} \quad (7.20)$$

puede ser fácilmente verificada. Aquí n_i es cualquier componente del vector normal unitario.

7.2 Fórmulas Green-Herrera

Sea Ω un dominio y $\Pi = \{\Omega_1, \dots, \Omega_E\}$ una partición o descomposición en subdominios del dominio Ω . Sea una ecuación diferencial en forma general

$$\mathcal{L}u = \mathcal{L}u_\Omega \equiv f_\Omega, \text{ en } \Omega_i, i = 1, \dots, E \quad (7.21)$$

con condiciones de frontera

$$B_j u = B_j u_\partial \equiv g_\partial, \text{ en } \partial\Omega \quad (7.22)$$

y saltos prescritos

$$J_k u = J_k u_\Gamma \equiv j_\Gamma, \text{ en } \Gamma \quad (7.23)$$

donde B_j y J_k son k operadores diferenciales. Aquí, $u_\Omega = (u_\Omega^1, \dots, u_\Omega^E)$, u_∂ , y u_Γ son funciones dadas en $\widehat{D}_1(\Omega)$, que definen los datos del problema. De manera tal que tenemos un problema bien planteado, es decir, se garantiza la existencia y la unicidad de la solución. El problema enunciado se denomina *Problema con Valores en la Frontera con Saltos Prescritos*.

Si $u \in \widehat{D}_1(\Omega)$, entonces la ecuación diferencial $\mathcal{L}u$ está definida en el interior de cada Ω_i para $i = 1, \dots, E$. De igual forma, si $w \in \widehat{D}_2(\Omega)$ entonces \mathcal{L}^*w está definida en el interior de cada Ω_i , para $i = 1, \dots, E$. Ambos operadores diferenciales podrían no estar definidas en $\Gamma \cup \partial\Omega$.

Y como por definición del operador diferencial \mathcal{L} y su operador diferencial adjunto formal \mathcal{L}^* satisfacen la condición

$$w \mathcal{L}u - u \mathcal{L}^*w = \nabla \cdot \underline{\mathcal{D}}(u, w) \quad (7.24)$$

donde $\underline{\mathcal{D}}(u, w)$ es una función bilineal definida en $\widehat{D}_1(\Omega) \times \widehat{D}_2(\Omega)$ apropiada para el operador \mathcal{L} . Además asumimos que existen funcionales bilineales

$\mathcal{B}(u, w)$, $\mathcal{C}(w, u)$, $\mathcal{J}(u, w)$ y $\mathcal{K}(w, u)$ donde las primeras dos están definidas en $\partial\Omega$ y las dos últimas sobre Γ , tal que

$$\underline{\mathfrak{D}}(u, w) \cdot \underline{n} = \mathcal{B}(u, w) - \mathcal{C}(w, u) \quad \text{en } \partial\Omega \quad (7.25)$$

y

$$- [[\underline{\mathfrak{D}}(u, w)]] \cdot \underline{n} = \mathcal{J}(u, w) - \mathcal{K}(w, u) \quad \text{en } \Gamma \quad (7.26)$$

generalmente, las definiciones de \mathcal{B} y \mathcal{J} depende de las condiciones de frontera y de criterios de suavidad del problema en particular de que se trate, si los coeficientes del operador diferencial son continuos, las fórmulas de Herrera para \mathcal{J} y \mathcal{K} satisfacen

$$[[\underline{\mathfrak{D}}(u, w)]] = \underline{\mathfrak{D}}(\dot{u}, [[w]]) + \underline{\mathfrak{D}}([[u]], \dot{w}) \quad (7.27)$$

i.e.

$$\mathcal{J}(u, w) \equiv -\underline{\mathfrak{D}}([[u]], \dot{w}) \cdot \underline{n} \text{ y } \mathcal{K}(w, u) \equiv \underline{\mathfrak{D}}(\dot{u}, [[w]]) \cdot \underline{n}. \quad (7.28)$$

Si integramos la ecuación (7.24) cada Ω_i para $i = 1, 2, \dots, E$, y se considera $\bar{\Omega} = \bigcup_{i=1}^E \bar{\Omega}_i$, se tiene que

$$\sum_{i=1}^E \int_{\Omega_i} (w\mathcal{L}u - u\mathcal{L}^*w) d\bar{x} = \sum_{i=1}^E \int_{\Omega_i} \nabla \cdot \underline{\mathfrak{D}}(u, w) d\bar{x} \quad (7.29)$$

aplicando el teorema generalizado de la divergencia

$$\int_{\Omega} \nabla \cdot \underline{\mathfrak{D}}(u, w) d\bar{x} = \int_{\partial\Omega} \underline{\mathfrak{D}}(u, w) \cdot \underline{n}_{\partial\Omega} d\bar{x} - \int_{\Gamma} [[\underline{\mathfrak{D}}(u, w)]] \cdot \underline{n}_{\Gamma} d\bar{x} \quad (7.30)$$

en el lado derecho de la Ec. (7.29) obtenemos

$$\sum_{i=1}^E \int_{\Omega_i} w\mathcal{L}u d\bar{x} - \sum_{i=1}^E \int_{\Omega_i} u\mathcal{L}^*w d\bar{x} = \int_{\Omega} \underline{\mathfrak{D}}(u, w) \cdot \underline{n} d\bar{x} - \int_{\Gamma} [[\underline{\mathfrak{D}}(u, w)]] \cdot \underline{n} d\bar{x} \quad (7.31)$$

desarrollando el álgebra de saltos en el segundo sumando del lado derecho de la ecuación anterior se tiene

$$\begin{aligned} & \sum_{i=1}^E \int_{\Omega_i} w \mathcal{L} u d\underline{x} - \sum_{i=1}^E \int_{\Omega_i} u \mathcal{L}^* w d\underline{x} \\ &= \int_{\Omega} \underline{\mathcal{D}}(u, w) \cdot \underline{n} d\underline{x} - \int_{\Gamma} [[\underline{\mathcal{D}}(\dot{u}, [w])]] \cdot \underline{n} d\underline{x} - \int_{\Gamma} [[\underline{\mathcal{D}}([u], \dot{w})]] \cdot \underline{n} d\underline{x}. \end{aligned} \quad (7.32)$$

Ahora, para poder usar las fórmulas de Green se introducen las siguientes funcionales bilineales reales definidas en $\widehat{D}_1(\Omega) \times \widehat{D}_2(\Omega)$. Sean las funcionales bilineales $\mathcal{B}(u, w)$, $\mathcal{C}^*(u, w)$, $\mathcal{J}(u, w)$ y $\mathcal{K}^*(u, w)$ tales que producen las siguientes descomposiciones

$$\underline{\mathcal{D}}(u, w) \cdot \underline{n} = \mathcal{B}(u, w) - \mathcal{C}^*(u, w) \quad \text{en } \partial\Omega \quad (7.33)$$

$$\underline{\mathcal{D}}(\dot{u}, [[w]]) \cdot \underline{n} \equiv \mathcal{K}^*(u, w) \quad \text{en } \Gamma \quad (7.34)$$

$$- \underline{\mathcal{D}}([u], \dot{w}) \cdot \underline{n} \equiv \mathcal{J}(u, w) \quad \text{en } \Gamma \quad (7.35)$$

donde $\mathcal{C}^*(u, w)$ es el transpuesto de la funcional bilineal $\mathcal{C}(w, u)$ y se define como

$$\mathcal{C}^*(u, w) \equiv \mathcal{C}(w, u) \quad (7.36)$$

de igual manera para

$$\mathcal{K}^*(u, w) \equiv \mathcal{K}(w, u). \quad (7.37)$$

La funcional bilineal $\mathcal{B}(u, w)$ está en función de los valores de frontera (condiciones de frontera y condiciones iniciales), mientras que la funcional $\mathcal{C}^*(u, w)$ involucra los valores desconocidos en $\partial\Omega$ (información desconocida).

Por otra parte, la funcional $\mathcal{K}^*(u, w)$ involucra los valores relacionados con los promedios \dot{u} en Γ mientras que la funcional $\mathcal{J}(u, w)$ involucra los valores relacionados con los saltos $[[u]]$ en Γ .

Adicionalmente, consideramos las funcionales bilineales reales definidas en $\widehat{D}_1(\Omega) \times \widehat{D}_2(\Omega)$:

$$\mathcal{P}(u, w) \equiv w \mathcal{L} u, \quad \text{en } \Omega_i \text{ para } i = 1, \dots, E \quad (7.38)$$

$$\mathcal{J}^*(u, w) \equiv w \mathcal{L}^* u, \quad \text{en } \Omega_i \text{ para } i = 1, \dots, E \quad (7.39)$$

Sustituyendo las Ecs. (7.33) a (7.39) en Ec. (7.32) y reordenando los términos, se obtiene la fórmula de Green-Herrera

$$\begin{aligned} & \sum_{i=1}^E \int_{\Omega_i} \mathcal{P}(u, w) d\underline{x} - \int_{\partial\Omega} \mathcal{B}(u, w) d\underline{x} - \int_{\Gamma} \mathcal{J}(u, w) d\underline{x} \quad (7.40) \\ &= \sum_{i=1}^E \int_{\Omega_i} \mathcal{J}^*(u, w) d\underline{x} - \int_{\partial\Omega} \mathcal{C}^*(u, w) d\underline{x} - \int_{\Gamma} \mathcal{K}^*(u, w) d\underline{x}. \end{aligned}$$

Ahora, sean las funcionales bilineales reales

$$\langle \mathcal{P}u, w \rangle, \langle \mathcal{B}u, w \rangle, \langle \mathcal{J}u, w \rangle, \langle \mathcal{Q}^*u, w \rangle, \langle \mathcal{C}^*u, w \rangle \text{ y } \langle \mathcal{K}^*u, w \rangle \quad (7.41)$$

definidas en $\widehat{D}_1(\Omega) \times \widehat{D}_2(\Omega)$ tales que

$$\langle \mathcal{P}u, w \rangle \equiv \sum_{i=1}^E \int_{\Omega_i} \mathcal{P}(u, w) d\underline{x} = \sum_{i=1}^E \int_{\Omega_i} w \mathcal{L}u d\underline{x} \quad \text{en } \Omega_i \text{ para } i = 1, \dots, E \quad (7.42)$$

$$\langle \mathcal{B}u, w \rangle \equiv \int_{\partial\Omega} \mathcal{B}(u, w) d\underline{x} \quad \text{en } \partial\Omega \quad (7.43)$$

$$\langle \mathcal{J}u, w \rangle \equiv \int_{\Gamma} \mathcal{J}(u, w) d\underline{x} \quad \text{en } \Gamma \quad (7.44)$$

$$\langle \mathcal{Q}^*u, w \rangle \equiv \sum_{i=1}^E \int_{\Omega_i} \mathcal{J}^*(u, w) d\underline{x} = \sum_{i=1}^E \int_{\Omega_i} w \mathcal{L}^*u d\underline{x} \quad \text{en } \Omega_i \text{ para } i = 1, \dots, E \quad (7.45)$$

$$\langle \mathcal{C}^*u, w \rangle \equiv \int_{\partial\Omega} \mathcal{C}^*(u, w) d\underline{x} \quad \text{en } \partial\Omega \quad (7.46)$$

$$\langle \mathcal{K}^*u, w \rangle \equiv \int_{\Gamma} \mathcal{K}^*(u, w) d\underline{x} \quad \text{en } \Gamma. \quad (7.47)$$

Si se sustituye las Ecs. (7.42) a (7.47) en (7.40), se tiene

$$\langle \mathcal{P}u, w \rangle - \langle \mathcal{B}u, w \rangle - \langle \mathcal{J}u, w \rangle = \langle \mathcal{Q}^*u, w \rangle - \langle \mathcal{C}^*u, w \rangle - \langle \mathcal{K}^*u, w \rangle \quad (7.48)$$

que se satisface para todo $(u, w) \in \widehat{D}_1(\Omega) \times \widehat{D}_2(\Omega)$, o bien, por la propiedad de linealidad

$$\langle (\mathcal{P} - \mathcal{B} - \mathcal{J})u, w \rangle = \langle (Q^* - \mathcal{C}^* - \mathcal{K}^*)u, w \rangle \quad (7.49)$$

o de manera compacta

$$\mathcal{P} - \mathcal{B} - \mathcal{J} = Q^* - \mathcal{C}^* - \mathcal{K}^* \quad (7.50)$$

esta expresión representa la fórmula Green-Herrera para operadores en campos discontinuos.

Las funcionales bilineales

$$\langle \mathcal{P}u, w \rangle, \langle \mathcal{B}u, w \rangle, \langle \mathcal{J}u, w \rangle, \langle Q^*u, w \rangle, \langle \mathcal{C}^*u, w \rangle \text{ y } \langle \mathcal{K}^*u, w \rangle \quad (7.51)$$

también se pueden considerar como operadores funcionales lineales definidos en $\widehat{D}_1(\Omega)$ y valuadas en $\widehat{D}_2^*(\Omega)$, i.e., valuados en el dual algebraico de $\widehat{D}_2(\Omega)$. Por ejemplo, $\mathcal{P} : \widehat{D}_1(\Omega) \rightarrow \widehat{D}_2^*(\Omega)$ dónde $\mathcal{P}u \in \widehat{D}_2^*(\Omega)$ y $u \in \widehat{D}_1(\Omega)$. De igual modo, los transpuestos de las funcionales bilineales

$$\langle \mathcal{P}^*u, w \rangle, \langle \mathcal{B}^*u, w \rangle, \langle \mathcal{J}^*u, w \rangle, \langle Qu, w \rangle, \langle \mathcal{C}u, w \rangle \text{ y } \langle \mathcal{K}u, w \rangle \quad (7.52)$$

se pueden considerar como operadores funcionales lineales definidos en $\widehat{D}_2(\Omega)$ y valuados en $\widehat{D}_1^*(\Omega)$, i.e., valuados en el espacio dual algebraico de $\widehat{D}_1(\Omega)$, por ejemplo $\mathcal{P}^* : \widehat{D}_2(\Omega) \rightarrow \widehat{D}_1^*(\Omega)$ dónde $\mathcal{P}w \in \widehat{D}_1^*(\Omega)$ y $w \in \widehat{D}_2(\Omega)$.

Finalmente, sean las funciones $u_\partial \in \widehat{D}_1(\Omega)$ y $u_\Gamma \in \widehat{D}_1(\Omega)$. Entonces las funcionales lineales $g_\partial \in \widehat{D}_2^*(\Omega)$ y $j_\Gamma \in \widehat{D}_2^*(\Omega)$ se define como

$$g_\partial(w) \equiv \langle \mathcal{B}u_\partial, w \rangle \text{ para todo } w \in \widehat{D}_2(\Omega) \quad (7.53)$$

$$j_\Gamma(w) \equiv \langle \mathcal{J}u_\Gamma, w \rangle \text{ para todo } w \in \widehat{D}_2(\Omega) \quad (7.54)$$

o brevemente:

$$g_\partial j_\Gamma \mathcal{B}u_\partial \quad (7.55)$$

$$j_\Gamma \equiv \mathcal{J}u_\Gamma. \quad (7.56)$$

Fórmula de Green La fórmula de Green es un caso particular de las fórmulas de Green-Herrera cuando $\langle \mathcal{J}u, w \rangle = 0$ y $\langle \mathcal{K}^*u, w \rangle = 0$, lo cual implica la continuidad de la función u y de sus derivadas normales a través de Γ .

Supóngase que las condiciones de salto en Γ son nulas, entonces al aplicar el teorema de la divergencia (7.30) en el lado derecho de Ec. (7.29) se obtienen

$$\sum_{i=1}^E \int_{\Omega_i} w \mathcal{L}u - \sum_{i=1}^E \int_{\Omega_i} u \mathcal{L}^*w d\underline{x} = \int_{\partial\Omega} \underline{\mathcal{D}}(u, w) \cdot \underline{n} d\underline{x}. \quad (7.57)$$

Si se compara con la Ec. (7.31) se observa que $[\underline{\mathcal{D}}(u, w)] = 0$. Ahora, si se introduce las funcionales bilineales previamente definidas se tiene

$$\sum_{i=1}^E \int_{\Omega_i} \mathcal{P}(u, w) d\underline{x} - \sum_{i=1}^E \int_{\Omega_i} \mathcal{J}^*(u, w) d\underline{x} = \int_{\partial\Omega} (\mathcal{B}(u, w) - \mathcal{C}^*(u, w)) d\underline{x} \quad (7.58)$$

de esta forma se obtiene

$$\langle \mathcal{P}u, w \rangle - \langle \mathcal{B}u, w \rangle = \langle \mathcal{Q}^*u, w \rangle - \langle \mathcal{C}^*u, w \rangle \quad (7.59)$$

y más compactamente

$$\mathcal{P} - \mathcal{B} = \mathcal{Q}^* - \mathcal{C}^* \quad (7.60)$$

para todo $(u, w) \in \widehat{D}_1(\Omega) \times \widehat{D}_2(\Omega)$. Esta última fórmula, es la llamada fórmula de Green convencional.

7.3 Formulaciones Variacionales con Valor en la Frontera con Saltos Prescritos

Definición 61 Se dice que las condiciones de frontera $g_\partial \in \widehat{D}_2^*(\Omega)$ en $\partial\Omega$ y las condiciones de salto prescrito $j_\Gamma \in \widehat{D}_2^*(\Omega)$ son condiciones compatibles cuando existe una función $u_{\partial\Gamma} \in \widehat{D}_1(\Omega)$ tal que

$$\begin{aligned} g_\partial &= \mathcal{B}u_{\partial\Gamma}, \text{ en } \partial\Omega \\ j_\Gamma &= \mathcal{J}u_{\partial\Gamma}, \text{ en } \Gamma. \end{aligned} \quad (7.61)$$

Es decir, que ambas condiciones se pueden derivar de una única función, de modo que realmente no imponen condiciones contradictorias.

Definición 62 *El problema de contorno con valores en la frontera con saltos prescritos (BVPJ) consiste en buscar una función $u \in \widehat{D}_1(\Omega)$, tal que satisfaga:*

1) *El operador*

$$\mathcal{L}u = \mathcal{L}u_\Omega = f_\Omega, \text{ en } \Omega \quad (7.62)$$

2) *Condiciones de frontera*

$$\mathcal{B}(u, \cdot) = \mathcal{B}(u_\partial, \cdot) = g_\partial, \text{ en } \partial\Omega \quad (7.63)$$

3) *Saltos prescritos*

$$\mathcal{J}(u, \cdot) = \mathcal{J}(u_\Gamma, \cdot) = j_\Gamma, \text{ en } \Gamma \quad (7.64)$$

donde u_Ω , u_∂ , y u_Γ son funciones dadas en $\widehat{D}_1(\Omega)$, que definen los datos del problema, además las condiciones de frontera en $\partial\Omega$ y las condiciones de salto en Γ deben de ser compatibles.

Además se usa la convención de que la Ec. (7.62) que se satisface solamente en los puntos interiores de cada una de las subregiones $\Omega_i, i = 1, \dots, E$.

Si definimos las siguientes funcionales f, g y j que pertenecen a $\widehat{D}_2^*(\Omega)$ cómo

$$\langle f, w \rangle \equiv \langle \mathcal{P}u_\Omega, w \rangle, \text{ para todo } w \in \widehat{D}_2(\Omega) \quad (7.65)$$

$$\langle g, w \rangle \equiv \langle \mathcal{B}u_\partial, w \rangle, \text{ para todo } w \in \widehat{D}_2(\Omega) \quad (7.66)$$

$$\langle j, w \rangle \equiv \langle \mathcal{J}u_\Gamma, w \rangle, \text{ para todo } w \in \widehat{D}_2(\Omega) \quad (7.67)$$

entonces

Definición 63 *Una formulación débil del problema con valores en la frontera con saltos prescritos se puede escribir como*

$$\mathcal{P}u = f; \quad \mathcal{B}u = g; \quad \mathcal{J}u = j. \quad (7.68)$$

Definición 64 *Llamaremos solución de un problema con valores en la frontera con saltos prescritos, a una función $u \in \widehat{D}_1(\Omega)$ que satisfaga la formulación débil del problema con valores en la frontera con saltos prescritos.*

En lo sucesivo se supondrá que existe al menos una solución del problema con valores en la frontera con saltos prescritos además la Ec. (7.68) es equivalente a la siguiente ecuación simple:

$$(\mathcal{P} - \mathcal{B} - \mathcal{J})u = f - g - j. \quad (7.69)$$

Una condición suficiente para que la Ec. (7.69) sea equivalente a la Ec. (7.68) es que \mathcal{B} y \mathcal{J} sean operadores de frontera para $\mathcal{P} : \widehat{D}_1(\Omega) \rightarrow \widehat{D}_2^*(\Omega)$. Si escribimos la Ec. (7.69) de manera más explícita resulta

$$\langle (\mathcal{P} - \mathcal{B} - \mathcal{J})u, w \rangle = \langle f - g - j, w \rangle \quad \text{para todo } w \in \widehat{D}_2^*(\Omega) \quad (7.70)$$

la cual representa la formulación variacional del problema con valores en la frontera con saltos prescritos y nos referiremos a ella como la ecuación variacional en términos de los datos del problema.

Haciendo uso de la fórmula Green-Herrera Ec. (7.50) se obtiene la siguiente formulación variacional equivalente en términos de la información complementaria:

$$\langle (Q - \mathcal{C} - \mathcal{K})^*u, w \rangle = \langle f - g - j, w \rangle \quad \text{para todo } w \in \widehat{D}_2(\Omega) \quad (7.71)$$

cuando se aplica el método de residuos pesados, la solución aproximada $u \in \widehat{D}_1(\Omega)$ satisface

$$\langle (Q - \mathcal{C} - \mathcal{K})^*\tilde{u}, w^\alpha \rangle = \langle f - g - j, w^\alpha \rangle \quad \text{con } \alpha = 1, \dots, E \quad (7.72)$$

donde $\{w^1, \dots, w^E\} \subset \widehat{D}_2(\Omega)$ es un sistema de funciones de peso.

Como la solución exacta satisface la Ec. (7.71) entonces se cumple que

$$\langle (Q - \mathcal{C} - \mathcal{K})^*(u - \tilde{u}), w^\alpha \rangle = 0 \quad \text{con } \alpha = 1, \dots, E \quad (7.73)$$

este resultado puede ser usado para analizar la información acerca de la solución exacta que está contenida en la solución aproximada.

8 Método de Trefftz

En este capítulo se considerarán problemas con valor en la frontera (VBVP) de la forma

$$\begin{aligned} \mathcal{L}u &= f \quad \text{en } \Omega \\ u &= g \quad \text{en } \partial\Omega \end{aligned} \quad (8.1)$$

donde

$$\mathcal{L}u = -\Delta u \quad (8.2)$$

como un caso particular del operador elíptico definido por la Ec. (4.43) de orden dos. Consideremos el problema dado por la Ec. (8.1) en el dominio $\Omega \subset \mathbb{R}^n$ y $\Pi = \{\Omega_1, \dots, \Omega_E\}$ una partición o descomposición en subdominios del dominio Ω , i.e. se asume que:

- 1.- Ω_i , para $i = 1, \dots, E$ es un subdominio de Ω ,
- 2.- $\Omega_i \cap \Omega_j = \emptyset$, siempre que $i \neq j$.
- 3.- $\Omega \subset \bigcup_{i=1}^E \overline{\Omega}_i$.

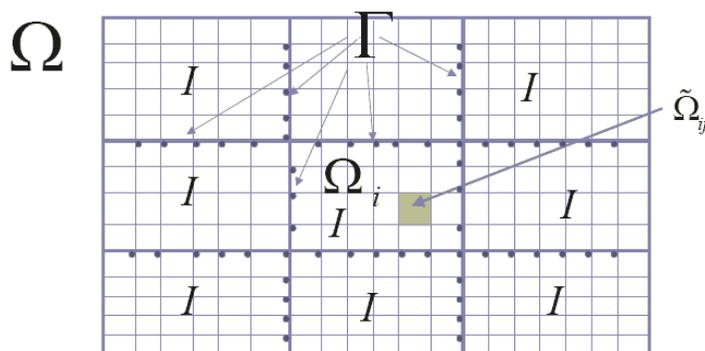


Figura 9: El dominio Ω , su frontera externa $\partial\Omega$ y la frontera interna Γ .

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

$$\partial\Omega \subset \bigcup_{i=1}^E \partial\Omega_i. \quad (8.3)$$

Adicionalmente definimos a la frontera interior cómo

$$\Gamma = \bigcup_{i \neq j} \partial\Omega_i \cap \partial\Omega_j \quad (8.4)$$

y a $\partial\Omega$ como la frontera exterior del dominio Ω .

También asumimos que salvo en un conjunto de medida cero sobre Γ se define un único vector normal denotado por \underline{n} cuyo sentido se elige arbitrariamente y el lado positivo de Γ es definido hacia el sentido positivo del vector normal.

El método Trefftz-Herrera es un método -al igual que en los métodos de Descomposición de Dominio-, permite construir la solución global definida en todo el dominio, resolviendo exclusivamente problemas de contorno locales formulados en cada uno de los subdominios de la partición.

La estrategia general para alcanzar dicho propósito consiste en recabar cierta información de la solución pero únicamente en la frontera interior Γ de la partición, que sea la suficiente para definir problemas de contorno independientes y bien planteados en cada uno de los subdominios, cuyas soluciones individuales sean precisamente las restricciones correspondientes de la solución global. Para esto, se elige de antemano cierta información objetivo de la solución en que posea esta propiedad, la cual se denota como información buscada.

Existen dos grandes categorías de métodos para recabar dicha información buscada: los métodos directos y los métodos indirectos. Los métodos directos utilizan soluciones locales del operador diferencial original para establecer las condiciones de compatibilidad que debe aportar la información buscada; mientras que los métodos indirectos utilizan para tal fin el operador diferencial adjunto. Entonces, a partir de estas condiciones de compatibilidad se deriva una matriz del sistema global asociada con el problema.

Los métodos directos se derivan de una forma muy general de las condiciones de continuidad de Poincaré-Steklov, mientras que los métodos indirectos se derivan de una teoría desarrollada por Herrera, la cual se relaciona con la metodología Trefftz, llamada con frecuencia teoría Trefftz-Herrera. Existen diversas maneras de implementar cada uno de estos métodos. Una de ellas, se basa en el uso de una clase especial de funciones de base y de peso, llamadas genéricamente funciones óptimas.

En los métodos de localización indirectos se desarrolla y se aplica un sistema de funciones de peso especializadas que tiene la propiedad de capturar

la información buscada de la solución en la frontera interior exclusivamente. La idea de construir tales funciones óptimas de peso surge del hecho de que en el método de residuos pesados, la información acerca de la solución exacta que contiene la solución aproximada, depende del sistema de funciones de peso que se aplica.

Para utilizar esta dependencia en la construcción de las funciones óptimas de peso, se requiere de un procedimiento de análisis. Las herramientas básicas de este análisis son las fórmulas de Green-Herrera, las cuales se pueden aplicar aún cuando las funciones de base y de peso son completamente discontinuas, cosa que no se puede hacer en los espacios de Sobolev estándares ni con los operadores definidos en ellos.

Entonces, las fórmulas de Green-Herrera se aplican para diseñar las funciones óptimas de peso adecuadamente, las cuales tienen entre otras propiedades las de ser soluciones locales a la ecuación diferencial homogénea asociada con el operador diferencial adjunto. Las funciones óptimas de peso se utilizan para derivar las condiciones de compatibilidad de las cuales se obtiene la información buscada.

Por otro lado, en los métodos de localización directos se introduce un espacio lineal cuyos elementos, llamados funciones óptimas de base, tiene la siguiente propiedad: una función óptima de base satisface las condiciones de continuidad de Poincaré-Steklov si y sólo si contienen la información buscada. Cuando se utiliza la aproximación directa, la obtención de la información buscada se transforma en la construcción de la función óptima de base que cumple con las condiciones de Poincaré-Steklov, de la cual se deriva la matriz del sistema global. Las funciones óptimas de base son soluciones locales de la ecuación diferencial homogénea asociada con el operador diferencial original del problema considerado.

Como se mencionó anteriormente, el término de funciones óptimas denota un conjunto de funciones que incluye tanto a las funciones óptimas de base como a las funciones óptimas de peso, de modo que el primero se compone de la suma directa de los dos últimos.

8.1 Conceptos Básicos

Trabajaremos con los espacios de Sobolev para definir los espacios $\widehat{D}_\alpha(\Omega)$, ya que estos son apropiados para manejar funciones discontinuas definidas por tramos, en donde las trazas de las funciones y de sus derivadas normales en las fronteras de los subdominios $\partial\Omega_i$ siempre están definidas y en conse-

cuencia, los saltos y los promedios de éstas. De esta forma, se garantiza la existencia y la unicidad de la solución para los problemas de contorno con saltos prescritos y con coeficientes discontinuos, en donde la solución también puede ser discontinua.

La ubicación dentro del dominio Ω tanto de los saltos prescritos como de las discontinuidades de los coeficientes del operador diferencial determinan la partición Π , de forma que solamente haya éstos sobre la frontera interior Γ . De esta forma, la solución solamente puede presentar discontinuidades en Γ .

Usando las definiciones del capítulo de Funciones Definidas por Tramos, entonces los espacios $\widehat{D}_1(\Omega)$ y $\widehat{D}_2(\Omega)$ corresponden a los espacios de las funciones de base y de las funciones de peso respectivamente. Las funciones de base construyen la solución de la ecuación diferencial; mientras que las funciones de peso o funciones de prueba ponderan a la ecuación diferencial.

En esta sección se trabaja con ecuaciones diferenciales parciales elípticas de segundo orden, en consecuencia, se utilizarán para definir los espacios de las funciones base y funciones de peso los espacios de Sobolev por tramos $\widehat{H}^s(\Omega)$ de orden $s = 2$, es decir

$$\widehat{D}_1(\Omega) \subset \widehat{H}^2(\Omega) \text{ y } \widehat{D}_2(\Omega) \subset \widehat{H}^2(\Omega). \quad (8.5)$$

Definición 65 Decimos que un conjunto de funciones $\mathcal{E} \subset \widehat{D}_2(\Omega)$ es un conjunto de funciones TH-Completo para $\mathcal{P} : \widehat{D}_1(\Omega) \rightarrow \widehat{D}_2^*(\Omega)$ cuando

$$\langle \mathcal{P}u, w \rangle = 0, \text{ para todo } w \in \mathcal{E} \quad (8.6)$$

es decir

$$\mathcal{P}u = 0. \quad (8.7)$$

Definición 66 Decimos que un operador $\mathcal{B} : \widehat{D}_1(\Omega) \rightarrow \widehat{D}_2^*(\Omega)$ es un operador de frontera para $\mathcal{P} : \widehat{D}_1(\Omega) \rightarrow \widehat{D}_2^*(\Omega)$, cuando $\mathcal{N}_{\mathcal{B}^*} \subset \widehat{D}_2(\Omega)$ es un conjunto de funciones TH-Completo para \mathcal{P} , es decir

$$\langle \mathcal{P}u, w \rangle = 0, \text{ para todo } w \in \mathcal{N}_{\mathcal{B}^*}. \quad (8.8)$$

Teorema 67 Sean los operadores \mathcal{B} y \mathcal{J} operadores de frontera para el operador \mathcal{P} y además, sean los espacios $\mathcal{N}_{\mathcal{P}^*}$, $\mathcal{N}_{\mathcal{B}^*}$ y $\mathcal{N}_{\mathcal{J}^*}$ espacios TH-Completos para los operadores \mathcal{P} , \mathcal{B} y \mathcal{J} respectivamente. Entonces, las ecuaciones

$$\mathcal{P}u = f; \quad \mathcal{B}u = g; \quad \mathcal{J}u = j \quad (8.9)$$

son equivalentes a la ecuación

$$\langle (\mathcal{P} - \mathcal{B} - \mathcal{J})u, w \rangle = \langle f - g - j, w \rangle. \quad (8.10)$$

En virtud de las fórmulas de Green-Herrera, se tienen dos formulaciones débiles. La primera, la ecuación

$$(\mathcal{P} - \mathcal{B} - \mathcal{J})u = f - g - j \quad (8.11)$$

referida como formulación variacional en términos de los datos del problema y la segunda llamada formulación variacional en términos de la información complementaria

$$(Q - \mathcal{C} - \mathcal{K})^*u = f - g - j \quad (8.12)$$

de hecho, si la función $u \in \widehat{D}_1(\Omega)$ es la solución del BVPJ se dice que los términos $\mathcal{P}u$, $\mathcal{B}u$ y $\mathcal{J}u$ son los datos del problema, mientras que los términos Q^*u , \mathcal{C}^*u y \mathcal{K}^*u son la información complementaria.

Si se tiene un BVPJ homogéneo con condiciones homogéneas, las siguientes observaciones son útiles.

Corolario 68 Sean los operadores \mathcal{B} y \mathcal{J} operadores de frontera para el operador \mathcal{P} y además, sean los espacios $\mathcal{N}_{\mathcal{P}^*}$, $\mathcal{N}_{\mathcal{B}^*}$ y $\mathcal{N}_{\mathcal{J}^*}$ espacios TH-Completo para los operadores \mathcal{P} , \mathcal{B} y \mathcal{J} respectivamente. Entonces se tiene que

$$\mathcal{P}u = 0; \quad \mathcal{B}u = 0; \quad \mathcal{J}u = 0 \Leftrightarrow \langle (\mathcal{P} - \mathcal{B} - \mathcal{J})u, w \rangle = 0. \quad (8.13)$$

Teorema 69 Sean los operadores \mathcal{C} y \mathcal{K} operadores de frontera para el operador Q y además, sean los espacios \mathcal{N}_Q , $\mathcal{N}_{\mathcal{C}}$ y $\mathcal{N}_{\mathcal{K}}$ espacios TH-Completo para los operadores Q , \mathcal{C} y \mathcal{K} respectivamente. Entonces se tiene que

$$Q^*u = 0; \quad \mathcal{C}^*u = 0; \quad \mathcal{K}^*u = 0 \Leftrightarrow \langle (Q - \mathcal{C} - \mathcal{K})^*u, w \rangle = 0. \quad (8.14)$$

8.1.1 Condiciones de Poincaré-Steklov

Sea el siguiente problema de Poisson con solución única $u \in C^1(\Omega)$ sujeto a condiciones de frontera homogéneas tipo Dirichlet

$$\begin{aligned} -\Delta u &= f_{\Omega} \quad \text{en } \Omega \\ u &= 0 \quad \text{en } \partial\Omega \end{aligned} \quad (8.15)$$

introduciendo la siguiente partición $\Pi = \{\Omega_1, \Omega_2\}$. Entonces, este problema se puede formular de manera equivalente en múltiples subdominios como

$$\begin{aligned} -\Delta u_1 &= f_\Omega \quad \text{en } \Omega_1 \\ u_1 &= 0 \quad \text{en } \partial\Omega \cap \partial\Omega_1 \end{aligned} \quad (8.16)$$

$$\begin{aligned} -\Delta u_2 &= f_\Omega \quad \text{en } \Omega_2 \\ u_2 &= 0 \quad \text{en } \partial\Omega \cap \partial\Omega_2 \end{aligned} \quad (8.17)$$

$$\begin{aligned} u_1 &= u_2 \quad \text{en } \Gamma \\ \frac{\partial u_1}{\partial n} &= \frac{\partial u_2}{\partial n} \quad \text{en } \Gamma \end{aligned} \quad (8.18)$$

donde u_i es la restricción de la solución u en Ω_i y \underline{n} es un vector normal a $\partial\Omega_i$. Las ecuaciones (8.18) representan las condiciones de transmisión en Γ .

Cuando se aplica algún método de descomposición de dominio a una ecuación diferencial, en general se tiene que resolver un problema de condiciones de transmisión en Γ . En particular, esta ecuación de transmisión se puede representar por medio del operador de Steklov-Poincaré. Para tal efecto, considerese los siguientes problemas tipo Dirichlet

$$\begin{aligned} -\Delta w_i &= f_\Omega \quad \text{en } \Omega_i \\ w_i &= 0 \quad \text{en } \partial\Omega \cap \partial\Omega_i \\ w_i &= \lambda \quad \text{en } \Gamma \end{aligned} \quad (8.19)$$

para $i = 1, 2$, donde λ es el valor desconocido de u en Γ . La solución w_i se puede escribir como

$$w_i = u_i^H + u_i^P \quad (8.20)$$

donde u_i y u_p son la solución de los siguientes problemas

$$\begin{aligned} -\Delta u_i^H &= 0 \quad \text{en } \Omega_i \\ u_i^H &= 0 \quad \text{en } \partial\Omega \cap \partial\Omega_i \\ u_i^H &= \lambda \quad \text{en } \Gamma \end{aligned} \quad (8.21)$$

y

$$\begin{aligned} -\Delta u_i^P &= f_\Omega \quad \text{en } \Omega_i \\ u_i^P &= 0 \quad \text{en } \partial\Omega \cap \partial\Omega_i \\ u_i^P &= 0 \quad \text{en } \Gamma \end{aligned} \quad (8.22)$$

de aquí que la función u_i^H sea una extensión armónica de λ en Ω_i , la cual se denotará como $\mathcal{H}_i\lambda$; además $\mathcal{H} = \mathcal{H}_1 + \mathcal{H}_2$. Por otro lado, la función u_i^P se denotará como $\mathcal{G}_i\lambda$; además $\mathcal{G} = \mathcal{G}_1 + \mathcal{G}_2$.

En consecuencia, comparando las ecuaciones del problema (8.16) a (8.18) con el problema (8.19) se tiene que

$$w_i = u_i, i = 1, 2 \text{ si y sólo si } \frac{\partial w_1}{\partial n} = \frac{\partial w_2}{\partial n} \text{ en } \Gamma \quad (8.23)$$

esta última condición equivale a que λ satisfaga la ecuación de transmisión de Steklov-Poincaré

$$\mathcal{S}\lambda = \mathcal{X} \text{ en } \Gamma \quad (8.24)$$

donde \mathcal{S} es el operador de Steklov-Poincaré y se define como

$$\begin{aligned} \mathcal{S}\lambda &\equiv \frac{\partial}{\partial n}\mathcal{H}_1\lambda - \frac{\partial}{\partial n}\mathcal{H}_2\lambda \equiv \left[\left[\frac{\partial}{\partial n}\mathcal{H}\lambda \right] \right] \\ \mathcal{X} &\equiv \frac{\partial}{\partial n}\mathcal{G}_1f_\Omega - \frac{\partial}{\partial n}\mathcal{G}_2f_\Omega \equiv \left[\left[\frac{\partial}{\partial n}\mathcal{G}f_\Omega \right] \right] \end{aligned} \quad (8.25)$$

finalmente, el operador inverso del operador Steklov-Poincaré \mathcal{S}^{-1} , se le llama operador de Poincaré-Steklov.

8.2 Método Indirecto de Trefftz-Herrera

Partiendo de la formulación variacional en términos de la información complementaria de un BVPJ

$$\langle (Q - \mathcal{C} - \mathcal{K})^*u, w \rangle = \langle f - g - j, w \rangle, \text{ para todo } w \in \widehat{D}_2(\Omega) \quad (8.26)$$

de la información relacionada con la solución $u \in \widehat{D}_1(\Omega)$ en el interior de los subdominios Ω_i de la partición Π está dada por el término Q^*u , en la frontera exterior $\partial\Omega$ por el término \mathcal{C}^*u y en la frontera interior Γ por el término \mathcal{K}^*u .

El método indirecto de Trefftz-Herrera se caracteriza por construir un espacio de funciones de peso especializado para capturar cierta información de la solución en las fronteras de los subdominios, dicho espacio se llama espacio de funciones óptimas de peso $\widehat{O}_T \in \widehat{D}_2(\Omega)$. La información que se busca de la solución en la frontera interior puede ser el promedio de la función, el promedio de sus derivadas o una combinación de éstos. De hecho, esa información buscada determina los espacios nulos relacionados con los

operadores Q, \mathcal{C} y \mathcal{K} cuando se aplica el método de residuos pesados. El análisis para construir las funciones óptimas de peso se detalla a continuación.

Primero notemos que la información buscada de la solución sólo está en $\Gamma \cup \partial\Omega$ y no en el interior de los subdominios Ω_i . Entonces, para obtener esa información se requiere eliminar el término Q^*u de la Ec. (8.26) aplicando el método de residuos pesados, la condición anterior se logra construyendo funciones de peso tales que satisfaga la condición $Qw = 0$ en el interior de cada subdominio Ω_i por separado, y de este modo, se introduce el espacio nulo de Q el cual es $N_Q = \left\{ w \in \widehat{D}_2(\Omega) \mid Qw = 0 \text{ en cada } \Omega_i \right\}$.

Definición 70 Definimos al núcleo de Q , como $w \in \widehat{D}_2(\Omega)$ tales que $Qw = 0$ en cada Ω_i , i.e.

$$N_Q \equiv \left\{ w \in \widehat{D}_2(\Omega) \mid Qw = 0 \text{ en cada } \Omega_i \right\}. \quad (8.27)$$

En este método se pretende recabar suficiente información de la solución en $\Gamma \cup \partial\Omega$ para proponer problemas de contorno locales bien planteados en cada subdominio Ω_i , i.e., no es necesario recabar toda la información posible de la solución en $\Gamma \cup \partial\Omega$. Esto induce la descomposición del operador \mathcal{K} en dos partes. Una parte se refiere a la información buscada de la solución en Γ y otra parte se refiere a la información redundante o no buscada de la solución en Γ .

Definición 71 Sea la descomposición $\{S_k, R_k\}$ de la funcional bilineal \mathcal{K} tal que

$$\mathcal{K} \equiv S_k + R_k \quad (8.28)$$

donde el operador S_k se toma de tal forma que S_k^*u sea precisamente la información buscada de la solución en Γ .

Con el objetivo de formalizar la descomposición del operador \mathcal{K} se introduce lo siguiente: Sean $S_k^*(u, w)$ y $R_k^*(u, w)$ funcionales bilineales reales definidas en $\widehat{D}_1(\Omega) \times \widehat{D}_2(\Omega)$ tales que producen la siguiente descomposición

$$\mathcal{K}^*(u, w) \equiv S_k^*(u, w) + R_k^*(u, w) \text{ para todo } w \in \Gamma \quad (8.29)$$

además sean $\langle S_k^* u, w \rangle$ y $\langle R_k^* u, w \rangle$ funcionales bilineales reales definidas en $\widehat{D}_1(\Omega) \times \widehat{D}_2(\Omega)$ tales que

$$\begin{aligned} \langle S_k^* u, w \rangle &\equiv \int_{\Gamma} S_k^*(u, w) d\underline{x} \text{ en } \Gamma & (8.30) \\ \langle R_k^* u, w \rangle &\equiv \int_{\Gamma} R_k^*(u, w) d\underline{x} \text{ en } \Gamma. \end{aligned}$$

El operador R_k se asocia con la información redundante o no buscada de la solución en Γ . Así, una vez considerada la descomposición del operador \mathcal{K} , cuando se requiere eliminar el término $R_k^* u$ de la Ec. (8.26). Aplicando el método de residuos pesados, la condición anterior se logra construyendo funciones de peso tales que satisfagan la condición $R_k^* u = 0$ en Γ y de este modo, se introduce el espacio nulo de R_k en cual es $N_{R_k} = \left\{ w \in \widehat{D}_2(\Omega) \mid R_k w = 0 \text{ en } \Gamma \right\}$.

Definición 72 Definimos al núcleo de R_k como $w \in \widehat{D}_2(\Omega)$ tales que $R_k w = 0$ en Γ , i.e

$$N_{R_k} \equiv \left\{ w \in \widehat{D}_2(\Omega) \mid R_k w = 0 \text{ en } \Gamma \right\}. \quad (8.31)$$

Definición 73 Sea un BVPJ con solución única $u \in \widehat{D}_1(\Omega)$, sean las funcionales bilineales S_k y R_k tales que $\mathcal{K} = S_k + R_k$, además sea $\hat{u} \in \widehat{D}_1(\Omega)$ tal que

$$S_k^* \hat{u} = S_k^* u \text{ en } \Gamma \quad (8.32)$$

entonces decimos que \hat{u} contiene la información buscada de la solución en Γ .

Por otro lado, notemos que para una ecuación diferencial elíptica de segundo orden se necesita condiciones de frontera en todo $\partial\Omega$ para que el problema de contorno esté bien planteado.

Lo anterior significa que, en general, no se busca información en $\partial\Omega$. Entonces se requiere eliminar el término $\mathcal{C}^* w$ de (8.26). Aplicando el método de residuos pesados, la condición anterior se logra construyendo funciones de peso tales que satisfagan la condición $\mathcal{C} w = 0$ en $\partial\Omega$ y de este modo, se introduce el espacio nulo de \mathcal{C} el cual es $N_{\mathcal{C}} = \left\{ w \in \widehat{D}_2(\Omega) \mid \mathcal{C} w = 0 \text{ en } \partial\Omega \right\}$.

Definición 74 Definimos el nulo de \mathcal{C} como $w \in \widehat{D}_2(\Omega)$ tales que $\mathcal{C}w = 0$ en $\partial\Omega$, i.e.

$$N_{\mathcal{C}} \equiv \left\{ w \in \widehat{D}_2(\Omega) \mid \mathcal{C}w = 0 \text{ en } \partial\Omega \right\}. \quad (8.33)$$

Definición 75 Definimos al espacio de funciones óptimas de peso \widehat{O}_T como

$$\widehat{O}_T \equiv N_Q \cap N_{\mathcal{C}} \cap N_{R_k} \subset \widehat{D}_2(\Omega) \quad (8.34)$$

i.e.

$$\langle (Q - \mathcal{C} - R_k)^*u, w \rangle = 0, \text{ para todo } w \in N_Q \cap N_{\mathcal{C}} \cap N_{R_k}. \quad (8.35)$$

Teorema 76 Método indirecto de Trefftz-Herrera

Sea el BVPJ $(\mathcal{P} - \mathcal{B} - \mathcal{J})u = f - g - j$, con los datos $f = \mathcal{P}u_{\Omega}$, $g = \mathcal{B}u_{\partial}$ y $j = \mathcal{J}u_{\Gamma}$ donde $u_{\Omega} \in \widehat{D}_1(\Omega)$, $u_{\partial} \in \widehat{D}_1(\Omega)$ y $u_{\Gamma} \in \widehat{D}_1(\Omega)$. Además sea \widehat{O}_T el espacio de funciones óptimas de peso TH-Completo para $S_k^* : \widehat{D}_1(\Omega) \times \widehat{D}_2^*(\Omega)$ y supongase que el BVPJ tiene una única solución $u \in \widehat{D}_1(\Omega)$. Entonces $\hat{u} \in \widehat{D}_1(\Omega)$ contiene la información buscada, i.e. $S_k^*\hat{u} = S_k^*u$, si y sólo si

$$-\langle S_k^*\hat{u}, w \rangle = \langle f - g - j, w \rangle, \text{ para toda } w \in \widehat{O}_T. \quad (8.36)$$

Por último, puesto que el operador Q se relaciona con el operador diferencial adjunto y el espacio de funciones base se toma igual que el espacio de funciones óptimas de peso, el método Trefftz-Herrera solamente es capaz de obtener información de la solución en la frontera interior Γ y no es capaz de obtener información de la solución en el interior de los subdominios Ω_i . Para encontrar la solución en el interior de los subdominios Ω_i se necesita un procedimiento llamado interpolación óptima.

El procedimiento llamado interpolación óptima consiste en extender la información buscada de la solución en Γ al interior de cada subdominio Ω_i de la siguiente manera. Una vez encontrada la información buscada de la solución en Γ , junto con las condiciones de frontera $\mathcal{B}u = \mathcal{B}u_{\partial}$ en $\partial\Omega$ y las condiciones de salto $\mathcal{J}u = \mathcal{J}u_{\Gamma}$ en Γ , se obtienen problemas de contorno locales bien planteados e independientes, en cada Ω_i . La solución de estos problemas de contorno locales extiende la información buscada de la solución en Γ al interior de los subdominios Ω_i .

8.3 Método directo de Steklov-Poincaré

Partiendo de la formulación variacional en términos de los datos de un BVPJ

$$\langle (\mathcal{P} - \mathcal{B} - \mathcal{J})u, w \rangle = \langle f - g - j, w \rangle, \text{ para todo } w \in \widehat{D}_2(\Omega) \quad (8.37)$$

donde la información relacionada con la solución $u \in \widehat{D}_1(\Omega)$ en el interior de los subdominios Ω_i de la partición Π está dada por el término $\mathcal{P}u$, en la frontera exterior $\partial\Omega$ por el término $\mathcal{B}u$ y en la frontera interior Γ por el término $\mathcal{J}u$.

El método directo de Steklov-Poincaré se caracteriza por construir un espacio de funciones de base especializada para contener cierta información de la solución en las fronteras de los subdominios, dicho espacio se llamará funciones óptimas de base, donde la información que se busca de la solución en la frontera interior es el promedio de la función. Una propiedad relevante es que las funciones óptimas de base contienen la información buscada si y sólo si satisfacen las condiciones de continuidad de Poincaré-Steklov en la frontera interior Γ . De hecho, la condición anterior determina los espacios nulos relacionados con los operadores \mathcal{P} , \mathcal{B} y \mathcal{J} . El análisis para construir las funciones óptimas de base se detalla a continuación.

Primero, se requiere que las funciones óptimas de base satisfagan las condiciones de continuidad de Poincaré-Steklov en la frontera interior Γ . Esto induce la descomposición del operador \mathcal{J} en dos partes. Una parte se refiere precisamente a dichas condiciones de continuidad en Γ , mientras que la otra parte se refiere a condiciones de continuidad redundantes en Γ .

Definición 77 *Sea la descomposición $\{S_j, R_j\}$ de la funcional bilineal \mathcal{J} tal que*

$$\mathcal{J} \equiv S_j + R_j \quad (8.38)$$

donde el operador S_j se toma de tal forma que $S_j v$ sea precisamente las condiciones de continuidad de Poincaré-Steklov en Γ .

Con el objetivo de formalizar la descomposición del operador \mathcal{J} se introduce lo siguiente. Sean $S_j(u, w)$ y $R_j(u, w)$ funciones bilineales reales definidas en $\widehat{D}_1(\Omega) \times \widehat{D}_2(\Omega)$ tales que producen la siguiente descomposición

$$\mathcal{J} \equiv S_j(u, w) + R_j(u, w) \text{ para toda } \underline{x} \in \Gamma, \quad (8.39)$$

luego, sean $\langle S_j u, w \rangle$ y $\langle R_j u, w \rangle$ funcionales bilineales reales definidas en $\widehat{D}_1(\Omega) \times \widehat{D}_2(\Omega)$ tales que

$$\begin{aligned} \langle S_j u, w \rangle &= \int_{\Gamma} S_j(u, w) d\underline{x} \text{ en } \Gamma \\ \langle R_j u, w \rangle &= \int_{\Gamma} R_j(u, w) d\underline{x} \text{ en } \Gamma \end{aligned} \quad (8.40)$$

El operador R_j se asocia con las condiciones de continuidad redundantes en Γ . También la descomposición del operador \mathcal{J} induce una descomposición correspondiente pero en los datos del problema, específicamente en $j = \mathcal{J}u_{\Gamma}$, de modo que

$$\begin{aligned} j_s + j_r &= j = \mathcal{J}u_{\Gamma} \\ j_s &= S_j u_{\Gamma} \\ j_r &= R_j u_{\Gamma}. \end{aligned} \quad (8.41)$$

La estrategia general del método directo de Steklov-Poincaré consiste en lo siguiente. La solución $u \in \widehat{D}_1(\Omega)$ se conforma de la suma de dos funciones. Una función óptima de base $v \in \widehat{O}_B$ la cual cumple con las condiciones de continuidad de Poincaré-Steklov en Γ y que contiene la información buscada de la solución en Γ ; y una función auxiliar $u_p \in \widehat{D}_1(\Omega)$ la cual cumple las condiciones de continuidad redundantes en Γ y que no contiene en absoluto la información buscada. Además, la función auxiliar u_p se construye con las soluciones particulares locales de cada uno de los subdominios de la partición, satisfaciendo las condiciones de frontera y las condiciones de saltos prescritos asociadas con las condiciones redundantes de continuidad en Γ . Ciertamente la función auxiliar u_p no es una función óptima de base.

De este modo, la solución $u \in \widehat{D}_1(\Omega)$ se puede escribir como $u = v + u_p$, donde $u_p \in \widehat{D}_1(\Omega)$ y $v \in \widehat{O}_B$, lo cual implica que la Ec. (8.37) se transforma en

$$\langle (\mathcal{P} - \mathcal{B} - \mathcal{J})v, w \rangle = \langle f - g - j, w \rangle - \langle (\mathcal{P} - \mathcal{B} - \mathcal{J})u_p, w \rangle \quad (8.42)$$

para toda $w \in \widehat{D}_2(\Omega)$. En cierto modo se puede decir que la función auxiliar u_p es una solución particular, construida resolviendo problemas de contorno locales, mientras que la función óptima de base v es una solución homogénea,

construida resolviendo un problema de contorno global y cuya utilidad es la de acoplar las mencionadas soluciones locales al contrarrestar sus discontinuidades que presentan en Γ introducidas por la forma en que se construyen.

Definición 78 *Sea la función auxiliar $u_p \in \widehat{D}_1(\Omega)$ tal que*

$$\langle (\mathcal{P} - \mathcal{B} - R_j)u_p, w \rangle = \langle f - g - j_R, w \rangle \text{ para toda } w \in \widehat{D}_2(\Omega) \quad (8.43)$$

y

$$S_k^* u_p = 0 \quad (8.44)$$

donde el operador S_k se utiliza para establecer la información buscada y se definió en la Ec.(8.28).

Nótese que si $S_k^* u_p = 0$ entonces $S_k^* u = S_k^*(v + u_p) = S_k^* v$, lo cual significa que efectivamente la función óptima de base contiene completamente la información buscada de la solución en Γ .

Para el caso de una ecuación diferencial elíptica de segundo orden la solución particular $u_p \in \widehat{D}_1(\Omega)$ se construye como una función que satisface por separado los problemas de contorno no homogéneos locales en cada subdominio Ω_i . Esta función también satisface las condiciones de frontera en $\partial\Omega$ y las condiciones de saltos prescritos pero únicamente de la función en Γ , imponiéndose que su promedio sea cero en Γ .

Sin embargo, puesto que dicha función se obtiene a partir de problemas locales independientes, aunque sí es posible construirla satisfaciendo condiciones de continuidad de la función en las fronteras de los subdominios, no lo es satisfaciendo simultáneamente condiciones de continuidad en sus derivadas normales. De aquí que la función óptima de base $v \in \widehat{O}_B$, con el objetivo de acoplar soluciones locales particulares, debe contribuir a satisfacer estas últimas condiciones de continuidad. Lo anterior se logra gracias a que la función óptima de base contrarresta el salto en las derivadas normales en Γ que representa la función auxiliar u_p . Esto último constituye precisamente las condiciones de continuidad de Poincaré-Steklov.

Además, para el caso elíptico, la información buscada de la solución en Γ es el promedio de la función. Nótese que la función auxiliar u_p satisface las condiciones de salto de la función en Γ y su promedio es cero en Γ . En consecuencia, la función óptima de base v es continua en Γ y es precisamente el promedio de la solución en Γ , i.e.

$$\dot{u}|_{\Gamma} = (\dot{u}_p + \dot{v})|_{\Gamma} = \dot{v}|_{\Gamma} = v|_{\Gamma}. \quad (8.45)$$

A continuación, se plantea el espacio de funciones óptimas de base. Puesto que la función auxiliar u_p ya satisface la ecuación diferencial homogénea, entonces las funciones óptimas de base se construyen de tal modo que satisfagan la condición $\mathcal{P}v = 0$ en el interior de cada subdominio Ω_i por separado y de este modo, se introduce el espacio nulo de \mathcal{P} el cual es

$$N_{\mathcal{P}} = \left\{ v \in \widehat{D}_1(\Omega) \mid \mathcal{P}v = 0 \text{ en cada } \Omega_i \right\}$$

y puesto que la función auxiliar u_p ya satisfizo las condiciones de frontera, entonces las funciones óptimas de base se construyen de tal modo que satisfagan la condición $\mathcal{B}v = 0$ en $\partial\Omega$ y de este modo se introduce el espacio nulo de \mathcal{B} el cual es $N_{\mathcal{B}} = \left\{ v \in \widehat{D}_1(\Omega) \mid \mathcal{B}v = 0 \text{ en cada } \partial\Omega \right\}$. Por último, la descomposición del operador \mathcal{J} introduce su espacio nulo de R_j , el cual es $N_{R_j} = \left\{ v \in \widehat{D}_1(\Omega) \mid R_j v = 0 \text{ en cada } \Gamma \right\}$.

Definición 79 Definimos a los espacios nulos de \mathcal{P} , \mathcal{B} y R_j como

$$\begin{aligned} N_{\mathcal{P}} &= \left\{ v \in \widehat{D}_1(\Omega) \mid \mathcal{P}v = 0 \text{ en cada } \Omega_i \right\} \\ N_{\mathcal{B}} &= \left\{ v \in \widehat{D}_1(\Omega) \mid \mathcal{B}v = 0 \text{ en cada } \partial\Omega \right\} \\ N_{R_j} &= \left\{ v \in \widehat{D}_1(\Omega) \mid R_j v = 0 \text{ en cada } \Gamma \right\} \end{aligned} \quad (8.46)$$

respectivamente.

Definición 80 Sea el espacio de las funciones óptimas de base \widehat{O}_B definido como

$$\widehat{O}_B \equiv N_{\mathcal{P}} \cap N_{\mathcal{B}} \cap N_{R_j} \subset \widehat{D}_1(\Omega) \quad (8.47)$$

lo cual implica que

$$\langle (\mathcal{P} - \mathcal{B} - R_j)u_p, w \rangle = 0 \text{ para toda } v \in N_{\mathcal{P}} \cap N_{\mathcal{B}} \cap N_{R_j}. \quad (8.48)$$

Así, el espacio de funciones de peso $\widehat{D}_2(\Omega)$ lo tomamos igual que el espacio de funciones de base $\widehat{D}_1(\Omega)$ y denotaremos a $\hat{v} \in \widehat{O}_B$ como una función óptima de base que contiene la información buscada de la solución $u \in \widehat{D}_1(\Omega)$, i.e. $S_k^* \hat{v} = S_k^* u$.

Como conclusión, el siguiente teorema establece las condiciones que se debe cumplir para que la función de base \hat{v} contenga la información buscada de la solución u en Γ .

Teorema 81 *Sea el BVPJ $(\mathcal{P} - \mathcal{B} - \mathcal{J})u = f - g - j$ con los datos $f = \mathcal{P}u_\Omega, g = \mathcal{B}u_\partial$ y $j = \mathcal{J}u_\Gamma$ donde $u_\Omega \in \widehat{D}_1(\Omega), u_\partial \in \widehat{D}_1(\Omega)$ y $u_\Gamma \in \widehat{D}_1(\Omega)$. Además sea \widehat{O}_B el espacio de funciones óptimas de base TH-Completo para $S_j : \widehat{D}_1(\Omega) \times \widehat{D}_2^*(\Omega)$ y supongase que el BVPJ tiene una única solución $u \in \widehat{D}_1(\Omega)$ tal que $u = \hat{v} + u_p$, donde $\hat{v} \in \widehat{O}_B$ y $u_p \in \widehat{D}_1(\Omega)$ cumple con la Ec.(8.42) y Ec.(8.43). Entonces $\hat{v} \in \widehat{O}_B$ contiene la información buscada, i.e. $S_k^* \hat{v} = S_k^* u$, si y sólo si*

$$- \langle S_j \hat{v}, w \rangle = \langle S_j u_p, w \rangle - \langle J_s, w \rangle, \text{ para toda } w \in \widehat{O}_B. \quad (8.49)$$

Finalmente, a diferencia del método indirecto de Trefftz-Herrera, el método directo de Steklov-Poincaré no requiere del procedimiento de interpolación óptima para extender la información buscada de la solución en Γ al interior de los subdominios Ω_i de la partición Π , ya que el operador \mathcal{P} se relaciona con el operador diferencial original.

9 Métodos de Funciones Discontinuas Definidas por Tramos

Consideremos la ecuación de Poisson en un dominio Ω y $\Pi = \{\Omega_1, \dots, \Omega_E\}$ una descomposición en subdominios del dominio Ω y asumiendo condiciones de frontera tipo Dirichlet igual a cero, i.e.

$$\begin{aligned} -\Delta \bar{u} &= f_\Omega \text{ en } \Omega \\ \bar{u} &= 0 \text{ sobre } \partial\Omega \end{aligned} \quad (9.1)$$

el espacio \bar{D} , donde la solución \bar{u} es buscada, se define como

$$\bar{D} = \{v \in H^2(\Omega) \mid \text{traza } v = 0 \text{ sobre } \partial\Omega\} \quad (9.2)$$

otro espacio que usaremos en lo sucesivo es

$$\tilde{D} = \left\{ v \in \hat{H}^2(\Omega, \Pi) \mid \text{traza } v = 0 \text{ sobre } \partial\Omega \right\}. \quad (9.3)$$

Cuando el dato f_Ω es tal que la solución \bar{u} pertenece a \bar{D} , entonces este problema es equivalente a el siguiente problema con valores en la frontera con saltos prescritos (BVPJ):

Encontrar $\tilde{u} \in \tilde{D}$ tal que

$$\begin{aligned} -\Delta \tilde{u} &= f_\Omega \text{ en } \Omega_\alpha, \alpha = 1, \dots, E \\ [[\tilde{u}]] &= \left[\left[\frac{\partial \tilde{u}}{\partial n} \right] \right] = 0 \text{ en } \Gamma. \end{aligned} \quad (9.4)$$

aquí las condiciones de frontera no aparecen ya que estas fueron incorporadas en la definición del espacio \tilde{D} , más precisamente, $\tilde{u} \in \tilde{D}$ satisface la ecuación (9.5) si y sólo si $\tilde{u} = u$.

Se desarrollarán dos maneras de aproximar el problema dado por la Ec. (9.4):

A.- Una manera es introducir una función auxiliar $\tilde{u}_p \in \tilde{D}$ que satisfaga

$$\begin{aligned} -\Delta \tilde{u}_p &= f_\Omega \text{ en } \Omega_\alpha, \alpha = 1, \dots, E \\ [[\tilde{u}_p]] &= 0 \text{ y } \hat{\tilde{u}}_p = 0 \text{ sobre } \Gamma \end{aligned} \quad (9.5)$$

entonces, si $u \equiv \tilde{u} - \tilde{u}_p$ obtenemos para $u \in \tilde{D}$ las ecuaciones

$$\begin{aligned} -\Delta u &= 0 \text{ en } \Omega_\alpha, \alpha = 1, \dots, E & (9.6) \\ [[u]] &= 0 \text{ y } \left[\left[\frac{\partial u}{\partial n} \right] \right] = - \left[\left[\frac{\partial \tilde{u}_p}{\partial n} \right] \right] \text{ en } \Gamma. \end{aligned}$$

B.- Otra opción es reemplazar la ecuación (9.5) por

$$\begin{aligned} -\Delta \tilde{u}_p &= f_\Omega \text{ en } \Omega_\alpha, \alpha = 1, \dots, E & (9.7) \\ \left[\left[\frac{\partial \tilde{u}_p}{\partial n} \right] \right] &= 0 \text{ y } \widehat{\frac{\partial \tilde{u}_p}{\partial n}} = 0 \text{ sobre } \Gamma \end{aligned}$$

en cuyo caso

$$\begin{aligned} -\Delta u &= 0 \text{ en } \Omega_\alpha, \alpha = 1, \dots, E & (9.8) \\ [[u]] &= -[[\tilde{u}_p]] \text{ y } \left[\left[\frac{\partial u}{\partial n} \right] \right] = 0 \text{ en } \Gamma. \end{aligned}$$

La primera aproximación da lugar al algoritmo Neumann-Neumann y la segunda aproximación da lugar al algoritmo Dirichlet-Dirichlet. Independientemente de la aproximación elegida, se están buscando funciones en el espacio lineal

$$D \equiv \left\{ u \in \tilde{D} \mid -\Delta u = 0 \text{ en } \Omega_\alpha; \alpha = 1, 2, \dots, E \right\}. \quad (9.9)$$

9.1 Algoritmos a Nivel Continuo

En esta sección, los algoritmos mostrados en la sección anterior, serán presentados a nivel discreto en una manera en que pueden ser aplicados a cualquier número de dimensiones y a cualquier número de particiones del subdominio Ω , incluyendo particiones con vértices y operadores diferenciales que sean positivos pero no positivos definidos. Aprovechando la ventaja del tipo de sistema discreto que se obtiene se usará en el algoritmo el método para resolución de sistemas lineales Gradiente Conjugado, ver sección (21.2.5).

9.1.1 Algoritmo Neumann-Neumann

Construir:

1.- $\tilde{u}_p \in \tilde{D}$ que satisfaga

$$\begin{aligned} -\Delta \tilde{u}_p &= f_\Omega \text{ en } \Omega_\alpha; \alpha = 1, \dots, E & (9.10) \\ [[\tilde{u}_p]] &= 0 \text{ y } \hat{\tilde{u}}_p = 0 \text{ en } \Gamma \end{aligned}$$

$\tilde{u} \in \tilde{D}$ que satisfaga

$$\begin{aligned} -\Delta \tilde{u} &= f_\Omega \text{ en } \Omega_\alpha; \alpha = 1, \dots, E & (9.11) \\ [[\tilde{u}]] &= \left[\left[\frac{\partial \tilde{u}}{\partial n} \right] \right] = 0 \text{ en } \Gamma \end{aligned}$$

y definimos $u = \tilde{u} - \tilde{u}_p$, donde $u = u_{21} \oplus u_{22}$.

2.- Construir $u_{21} \in D$, tal que

$$\left[\left[\frac{\partial u_{21}}{\partial n} \right] \right] = - \left[\left[\frac{\partial \tilde{u}_p}{\partial n} \right] \right] \text{ y } \frac{\hat{\partial u_{21}}}{\partial n} = 0 \text{ en } \Gamma \quad (9.12)$$

3.- Definiendo $r^0 \in D$ tal que

$$[[r^0]] = 0 \text{ y } \hat{r}^0 = \hat{u}_{21} \text{ en } \Gamma \quad (9.13)$$

sea $p^0 = r^0$ y $u^0 = 0$.

Usando el método de Gradiente Conjugado, entonces para $n = 0, 1, 2, \dots$

4.- Construir $\psi^n \in D$ tal que

$$\left[\left[\frac{\partial \psi^n}{\partial n} \right] \right] = 0 \text{ y } \frac{\hat{\partial \psi^n}}{\partial n} = \frac{\hat{\partial p^n}}{\partial n} \text{ en } \Gamma \quad (9.14)$$

5.-

$$\alpha^n = \frac{p^n \cdot p^n}{p^n \cdot p^n + \psi^n \cdot \psi^n} \quad (9.15)$$

6.-

$$u^{n+1} = u^n + \alpha^n p^n \quad (9.16)$$

7.- Además, construir $q^n \in D$ tal que

$$[[q^n]] = 0 \text{ y } \hat{q}^n = \hat{\psi}^n \text{ en } \Gamma \quad (9.17)$$

8.-

$$r^{n+1} = r^n - \alpha^n q^n \quad (9.18)$$

9.-

$$\beta^n = \frac{r^{n+1} \cdot r^{n+1}}{r^n \cdot r^n} \quad (9.19)$$

10.-

$$p^{n+1} = r^{n+1} + \beta^n p^n \quad (9.20)$$

11.-

$$n = n + 1, \quad (9.21)$$

y regresar a 4.

9.1.2 Algoritmo Dirichlet-Dirichlet

Construir:

1.- $\tilde{u}_p \in \tilde{D}$ que satisfaga

$$-\Delta \tilde{u}_p = f_\Omega \text{ en } \Omega_\alpha; \alpha = 1, \dots, E \quad (9.22)$$

$$\left[\left[\frac{\tilde{u}_p}{\partial n} \right] \right] = 0 \text{ y } \hat{\tilde{u}}_p = 0 \text{ en } \Gamma$$

$\tilde{u} \in \tilde{D}$ que satisfaga

$$-\Delta \tilde{u} = f_\Omega \text{ en } \Omega_\alpha; \alpha = 1, \dots, E \quad (9.23)$$

$$[[u]] = -[[\tilde{u}_p]] \text{ y } \left[\left[\frac{\partial u}{\partial n} \right] \right] = 0 \text{ en } \Gamma.$$

y definimos $u = \tilde{u} - \tilde{u}_p$, donde $u = u_{11} \oplus u_{12}$.

2.- Construir $u_{11} \in D$, tal que

$$[[u_{11}]] = -[[\tilde{u}_p]] \text{ y } \hat{u}_{11} = 0 \text{ en } \Gamma \quad (9.24)$$

3.- Definiendo $r^0 \in D$ tal que

$$\left[\left[\frac{\partial r^0}{\partial n} \right] \right] = 0 \text{ y } \widehat{\frac{\partial r^0}{\partial n}} = \widehat{\frac{\partial u_{11}}{\partial n}} \text{ en } \Gamma \quad (9.25)$$

sea $p^0 = r^0$ y $u^0 = 0$.

Usando el método de Gradiente Conjugado, entonces para $n = 0, 1, 2, \dots$

4.- Construir $\psi^n \in D$ tal que

$$[[\psi^n]] = 0 \text{ y } \widehat{\psi^n} = \widehat{p^n} \text{ en } \Gamma \quad (9.26)$$

5.-

$$\alpha^n = \frac{p^n \cdot p^n}{p^n \cdot p^n + \psi^n \cdot \psi^n} \quad (9.27)$$

6.-

$$u^{n+1} = u^n + \alpha^n p^n \quad (9.28)$$

7.- Además, construir $q^n \in D$ tal que

$$\left[\left[\frac{\partial q^n}{\partial n} \right] \right] = 0 \text{ y } \widehat{\frac{\partial q^n}{\partial n}} = \widehat{\frac{\partial \psi^n}{\partial n}} \text{ en } \Gamma \quad (9.29)$$

8.-

$$r^{n+1} = r^n - \alpha^n q^n \quad (9.30)$$

9.-

$$\beta^n = \frac{r^{n+1} \cdot r^{n+1}}{r^n \cdot r^n} \quad (9.31)$$

10.-

$$p^{n+1} = r^{n+1} + \beta^n p^n \quad (9.32)$$

11.-

$$n = n + 1, \quad (9.33)$$

y regresar a 4.

Observación 5 *Notemos que en estas formulaciones son tales que de manera directa se obtienen transformaciones positivo definidas sin recurrir a los multiplicadores de Lagrange.*

9.2 Discretización Axiomática

Sea \overline{D} un espacio de Hilbert de funciones de dimensión finita definido en Ω de dimensión \overline{N} , sea $\Pi = \{\Omega_1, \dots, \Omega_E\}$ una partición, definiendo, para cada $\alpha = 1, \dots, E$,

$$D(\Omega_\alpha) = \{v \mid v = u|_{\Omega_\alpha} \text{ y } u \in \overline{D}\} \quad (9.34)$$

entonces, escribimos

$$\tilde{D} = D(\Omega_1) \oplus \dots \oplus D(\Omega_E) \quad (9.35)$$

por lo tanto, \tilde{D} es un espacio de funciones definidas por pedazos y bajo la inmersión natural de \overline{D} dentro de \tilde{D} , tenemos que $\overline{D} \subset \tilde{D}$. Una función $\tilde{w} \in \tilde{D}(\Omega)$ se dice que tiene soporte local cuando existe un $\alpha \in \{1, \dots, E\}$ tal que el soporte de \tilde{w} está contenido en la clausura de Ω_α .

Dada cualquier función $\overline{w} \in \overline{D}$, decimos que una función $\tilde{w} \in \tilde{D}$ es hija de \overline{w} , cuando \tilde{w} es la restricción de \overline{w} a un subdominio de la partición, claramente, todas las hijas de una función $\overline{w} \in \overline{D}$ tienen soporte local. En cuanto al producto interior en estos espacios, asumiremos que ellos satisfacen

$$u \cdot w = \sum_{\alpha=1}^E u_\alpha \cdot w_\alpha \quad (9.36)$$

donde, $u = \{u^1, \dots, u^E\}$ y $w = \{w^1, \dots, w^E\}$.

Sea $\overline{\mathcal{B}} \subset \overline{D}$ una base de \overline{D}

$$\overline{\mathcal{B}} = \{\overline{w}^1, \dots, \overline{w}^{\overline{N}}\} \quad (9.37)$$

entonces para cada $i = 1, \dots, \overline{N}$, $\mathcal{B}^i \subset \overline{D}$ es una colección de hijas de \overline{w}^i . Además escribimos

$$\mathcal{B} = \bigcup_{i=1}^{\overline{N}} \mathcal{B}^i \subset \tilde{D} \quad (9.38)$$

claramente, los elementos de \mathcal{B} tienen soporte local, y también asumimos que \mathcal{B} , como fue definida es una base linealmente independiente de \tilde{D} .

La colección de conjuntos $\{\mathcal{B}^1, \dots, \mathcal{B}^{\overline{N}}\}$ es clasificado en dos subfamilias $\{\mathcal{B}_I^1, \dots, \mathcal{B}_I^{\overline{N}_I}\} \subset \{\mathcal{B}^1, \dots, \mathcal{B}^{\overline{N}}\}$ y $\{\mathcal{B}_\Gamma^1, \dots, \mathcal{B}_\Gamma^{\overline{N}_\Gamma}\} \subset \{\mathcal{B}^1, \dots, \mathcal{B}^{\overline{N}}\}$; ellos han sido definidos por las siguientes condiciones $\mathcal{B}^i \in \{\mathcal{B}_I^1, \dots, \mathcal{B}_I^{\overline{N}_I}\}$ si y sólo si la

cardinalidad \mathcal{B}^i es uno, $\mathcal{B}^i \in \{\mathcal{B}_\Gamma^1, \dots, \mathcal{B}_\Gamma^{N_\Gamma}\}$ si y sólo si la cardinalidad de \mathcal{B}^i es más grande que uno. Entonces

$$\{\mathcal{B}^1, \dots, \mathcal{B}^{\overline{N}}\} = \{\mathcal{B}_I^1, \dots, \mathcal{B}_I^{N_I}\} \cup \{\mathcal{B}_\Gamma^1, \dots, \mathcal{B}_\Gamma^{N_\Gamma}\} \quad (9.39)$$

definiendo

$$\mathcal{B}_I = \bigcup_{i=1}^{N_I} \mathcal{B}_I^i \quad \text{y} \quad \mathcal{B}_\Gamma = \bigcup_{i=1}^{N_\Gamma} \mathcal{B}_\Gamma^i \quad (9.40)$$

tal que

$$\mathcal{B} = \mathcal{B}_I \cup \mathcal{B}_\Gamma. \quad (9.41)$$

Ahora definimos una familia de conjuntos \mathcal{B}_Γ^i , $i = 1, \dots, \overline{N}_\Gamma$, donde cada conjunto $\overline{\mathcal{B}}_\Gamma^i$ es definido al remplazar el conjunto \mathcal{B}_Γ^i por un conjunto equivalente linealmente independiente (en el sentido que cada uno de \mathcal{B}_Γ^i y $\overline{\mathcal{B}}_\Gamma^i$ generan el mismo espacio lineal). La notación siguiente es adoptada

$$\mathcal{B}_\Gamma^i = \left\{ w_M^i, w_{J_1}^i, \dots, w_{J_{m(i)}}^i \right\} \quad (9.42)$$

donde w_M^i es definida como la función madre de la función $\overline{w}_M^i \in \overline{\mathcal{B}} \subset \overline{\mathcal{D}}$, i.e.

$$w_M^i \equiv \overline{w}^i \in \overline{\mathcal{B}} \subset \overline{\mathcal{D}} \quad (9.43)$$

además, el conjunto

$$\mathcal{B}_J^i = \left\{ w_{J_1}^i, \dots, w_{J_{m(i)}}^i \right\} \quad (9.44)$$

es un complemento algebraico del conjunto $\{\overline{w}^i\}$, con la propiedad que $\overline{\mathcal{B}}_\Gamma^i$ cuando es definida por la Ec. (9.42), genera el mismo espacio lineal que $\overline{\mathcal{B}}_\Gamma^i$, otras definiciones son

$$\mathcal{B}_{\Gamma M} = \left\{ w_M^1, \dots, w_M^{\overline{N}_\Gamma} \right\} \quad (9.45)$$

$$\mathcal{B}_{\Gamma J} = \bigcup_{i=1}^{\overline{N}_\Gamma} \mathcal{B}_J^i$$

$$\mathcal{B}_\Gamma = \mathcal{B}_{\Gamma M} \cup \mathcal{B}_{\Gamma J}$$

notemos que $\mathcal{B}_{\Gamma M} \subset \overline{\mathcal{B}} \subset \overline{\mathcal{D}}$. Con esta definición $\overline{\mathcal{B}}_\Gamma$ y \mathcal{B}_Γ generan el mismo espacio lineal, sin embargo la diferencia significativa entre $\overline{\mathcal{B}}_\Gamma$ y \mathcal{B}_Γ es que

todos los elementos de \mathcal{B}_Γ tienen soporte local, lo cual no es cierto para $\overline{\mathcal{B}}_\Gamma$. Además una propiedad adicional es

$$\overline{\mathcal{B}} = \mathcal{B}_{\Gamma M} + \mathcal{B}_I. \quad (9.46)$$

Los subespacios generados por las funciones $\mathcal{B}_I, \mathcal{B}_\Gamma, \mathcal{B}_{\Gamma J}$ y $\mathcal{B}_{\Gamma M}$ serán denotados por $\tilde{D}_I, \tilde{D}_\Gamma, \tilde{D}_{\Gamma_1}$, y \tilde{D}_{Γ_2} respectivamente y cuyas dimensiones de los espacios $\tilde{D}_I, \tilde{D}_\Gamma$ y \tilde{D} son N_I, N_Γ y \tilde{N} respectivamente y satisfacen $\tilde{N} = N_I + N_\Gamma$.

Además

$$\left. \begin{aligned} \tilde{D} &= \tilde{D}_I + \tilde{D}_\Gamma & \text{con } \tilde{D}_I \cap \tilde{D}_\Gamma &= \{0\} \\ \tilde{D}_\Gamma &= \tilde{D}_{\Gamma_1} + \tilde{D}_{\Gamma_2} & \text{con } \tilde{D}_{\Gamma_1} \cap \tilde{D}_{\Gamma_2} &= \{0\} \end{aligned} \right\} \quad (9.47)$$

y

$$\overline{D} = \tilde{D}_I + \tilde{D}_{\Gamma_2}, \quad (9.48)$$

la Ec. (9.47) implica que cualquier función $\tilde{v} \in \tilde{D}$ y cualquier función $\tilde{v}_\Gamma \in \tilde{D}_\Gamma$ puede ser escrita de forma única de las siguientes maneras

$$\tilde{v} = \tilde{v}_\Gamma + \tilde{v}_I, \quad \text{con } \tilde{v}_\Gamma \in \tilde{D}_\Gamma \text{ y } \tilde{v}_I \in \tilde{D}_I \quad (9.49)$$

$$\tilde{v} = \tilde{v}_J + \tilde{v}_M, \quad \text{con } \tilde{v}_J \in \tilde{D}_{\Gamma_1} \text{ y } \tilde{v}_M \in \tilde{D}_{\Gamma_2} \quad (9.50)$$

$$\tilde{v} = \tilde{v}_J + \tilde{v}_M + \tilde{v}_I, \quad \text{con } \tilde{v}_J \in \tilde{D}_{\Gamma_1}, \tilde{v}_M \in \tilde{D}_{\Gamma_2} \text{ y } \tilde{v}_I \in \tilde{D}_I. \quad (9.51)$$

Por otro lado, si el espacio $D \subset \tilde{D}$ es definido como el complemento ortogonal, con respecto a \tilde{D} , de $\tilde{D}_I \subset \tilde{D}$, i.e. $D = \left\{ v \in \tilde{D} \mid v \cdot w = 0, \forall w \in \tilde{D}_I \right\}$, entonces

$$\tilde{D} = D + \tilde{D}_I \text{ y } D \cap \tilde{D}_I = \{0\} \quad (9.52)$$

introduciendo la notación $Proy_D : \tilde{D} \rightarrow D$ es introducida por el operador proyección de vectores de \tilde{D} sobre D , recordando que $\tilde{D} = \tilde{D}_\Gamma + \tilde{D}_I$ y $\tilde{D}_\Gamma \cap \tilde{D}_I = \{0\}$, entonces la

$$Proy_D \tilde{D}_\Gamma = D \quad (9.53)$$

además la función $Proy_D : \tilde{D} \rightarrow D$ es una biyección.

En lo que sigue de este capítulo, el complemento ortogonal de los subespacios de D serán tomados con respecto a D . Usando tal notación, adicionalmente definimos

$$D_{11} \equiv Proj_D \tilde{D}_{\Gamma_1} \quad \text{y} \quad D_{12} \equiv Proj_D \tilde{D}_{\Gamma_2} \quad (9.54)$$

junto con

$$D_{21} \equiv (D_{11})^\perp \quad y \quad D_{22} \equiv (D_{12})^\perp. \quad (9.55)$$

Entonces

$$D = D_{11} + D_{12} \quad y \quad D_{11} \cap D_{12} = \{0\} \quad (9.56)$$

ya que

$$D = \text{Proy}_D \tilde{D}_{\Gamma_1} + \text{Proy}_D \tilde{D}_{\Gamma_2} \quad (9.57)$$

por la Ec. (9.47).

9.3 Esquema General

En esta sección, se establecerá el esquema general en términos de los distintos métodos de subestructuración pueden ser formulados.

Axiom 82 *La única suposición de este esquema es que existe un espacio de Hilbert D y un par de subespacios cerrados de D , $\{D_{11}, D_{12}\}$ con la propiedad de que*

$$D = D_{11} + D_{12} \quad y \quad D_{11} \cap D_{12} = \{0\}. \quad (9.58)$$

Definición 83 *Sea*

$$D_{21} \equiv (D_{11})^\perp \quad y \quad D_{22} \equiv (D_{12})^\perp. \quad (9.59)$$

Teorema 84 *Asumiendo el axioma y definición anteriores se tiene*

$$\left. \begin{aligned} D &= D_{11} + D_{21} & y & & D_{11} \cap D_{21} &= \{0\} \\ D &= D_{12} + D_{22} & y & & D_{12} \cap D_{22} &= \{0\} \\ D &= D_{21} + D_{22} & y & & D_{21} \cap D_{22} &= \{0\} \end{aligned} \right\} \quad (9.60)$$

De las formulaciones dadas por las Ecs(9.58 y 9.60) implican que cualquier función $u \in D$ puede ser escrita de forma única como

$$u = u_{11} + u_{12} = u_{21} + u_{22} \quad (9.61)$$

con

$$u_{\alpha\beta} \in D_{\alpha\beta} \quad \text{con } \alpha, \beta = 1, 2. \quad (9.62)$$

Múltiples métodos iterativos de subestructuración pueden ser formulados en términos de los siguientes dos problemas abstractos que se formulan a continuación:

Problema 1: En este problema $u_{21} \in D_{21}$ es un dato: Entonces, dado $u_{21} \in D_{21}$, encontrar $u \in D_{12}$ tal que $u = u_{21} + u_{22}$, para alguna $u_{22} \in D_{22}$.

Problema 2: En este problema $u_{11} \in D_{11}$ es un dato: Entonces, dado $u_{11} \in D_{11}$, encontrar $u \in D_{22}$ tal que $u = u_{11} + u_{12}$, para alguna $u_{12} \in D_{12}$.

Dependiendo de la forma en que los subespacios de D sean escogidos, entonces estos problemas llevan a una generalización de versiones de aproximaciones tipo Neumann-Neumann y Dirichlet-Dirichlet.

De la Ec.(9.61), se deduce lo siguiente

$$u_{2\alpha} = \sum_{\beta=1}^2 (u_{1\beta})_{2\alpha} \quad \text{y} \quad u_{1\alpha} = \sum_{\beta=1}^2 (u_{2\beta})_{1\alpha} \quad (9.63)$$

con $\alpha = 1, 2$. Definiendo para cada $\alpha, \beta = 1, 2$ y para cada $u \in D$, la función $\tau_{\alpha\beta} : D_{1\alpha} \rightarrow D_{2\beta}$ y $\mu_{\alpha\beta} : D_{2\alpha} \rightarrow D_{1\beta}$ por

$$\tau_{\alpha\beta}u \equiv (u_{1\alpha})_{2\beta} \quad \text{y} \quad \mu_{\alpha\beta}u \equiv (u_{2\alpha})_{1\beta}. \quad (9.64)$$

Lema 85 Cuando $u \in D_{12}$ y $w \in D_{22}$ se tiene que

$$w \cdot \tau_{22}u = -u \cdot \mu_{22}w \quad (9.65)$$

Corolario 86 Definiendo la transformación $T_D : D_{12} \rightarrow D_{12}$, para toda $u \in D_{12}$ por

$$T_D u \equiv -\mu_{22}\tau_{22}u \quad (9.66)$$

y la transformación $T_N : D_{22} \rightarrow D_{22}$, para toda $u \in D_{22}$ por

$$T_N u \equiv -\tau_{22}\mu_{22}u \quad (9.67)$$

entonces, cada una de estas transformaciones es no-negativa definida.

Teorema 87 Dadas las formulaciones de los problemas 1 y 2, sea I la transformación identidad, entonces:

A) Una función $u \in D_{12}$ es la solución del Problema 1, si y sólo si

$$(I + T_D)u = \mu_{12}u_{21} \quad (9.68)$$

B) Una función $u \in D_{22}$ es la solución del Problema 2, si y sólo si

$$(I + T_N)u = \tau_{12}u_{11}. \quad (9.69)$$

Aquí las transformaciones $(I + T_D) : D_{12} \rightarrow D_{12}$ y $(I + T_N) : D_{22} \rightarrow D_{22}$ son positiva definida. Ya que ambas $I + T_D$ y $I + T_N$ son transformaciones positiva definida, el método de Gradiente Conjugado (21.2.5) es aplicable a este tipo de problemas. Para obtener los nuevos algoritmos se usa la siguiente secuencia de pasos:

Algoritmo 1

Sea u^0 dado, se calcula $r^0 = b - Au^0$, $p^0 = r^0$.

Para $n = 0, 1, \dots$

1.-

$$\alpha^n = \frac{p^n \cdot p^n}{p^n \cdot Ap^n} \tag{9.70}$$

2.-

$$u^{n+1} = u^n + \alpha^n p^n \tag{9.71}$$

3.-

$$r^{n+1} = r^n - \alpha^n Ap^n \tag{9.72}$$

4.-

$$\beta^n = \frac{r^{n+1} \cdot r^{n+1}}{r^n \cdot r^n} \tag{9.73}$$

5.-

$$p^{n+1} = r^{n+1} + \beta^n p^n \tag{9.74}$$

6.-

$$n = n + 1 \tag{9.75}$$

y regresar a 1.

Cuando se aplica la Ec.(9.68) y se usa

$$(I - \mu_{22}\tau_{22})u = \mu_{12}u_{21}$$

tal que $A = I - \mu_{12}u_{21} = I + T_D$ y $b = \mu_{12}u_{21}$, entonces el esquema general toma la forma:

Algoritmo 2

Sea $p^0 = r^0 = b = \mu_{12}u_{21}$ y $u^0 = 0$.

Para $n = 0, 1, \dots$

1.-

$$\psi^n = \tau_{22}p^n \quad (9.76)$$

2.-

$$\alpha = \frac{p^n \cdot p^n}{p^n \cdot p^n + \psi^n \cdot \psi^n} \quad (9.77)$$

3.-

$$u^{n+1} = u^n + \alpha^n p^n \quad (9.78)$$

4.-

$$q^n = \mu_{22}\psi^n \quad (9.79)$$

5.-

$$r^{n+1} = r^n - \alpha^n q^n \quad (9.80)$$

6.-

$$\beta^n = \frac{r^{n+1} \cdot r^{n+1}}{r^n \cdot r^n} \quad (9.81)$$

7.-

$$p^{n+1} = r^{n+1} + \beta^n p^n \quad (9.82)$$

8.-

$$n = n + 1 \quad (9.83)$$

y regresar a 1.

El algoritmo para la ecuación

$$(I - \tau_{22}\mu_{22})u = \tau_{12}u_{11} \quad (9.84)$$

se obtiene de forma similar.

Ahora, Consideremos el operador diferencial de segundo orden en un dominio Ω y sin pérdida de generalidad consideramos $\Pi = \{\Omega_1, \Omega_2\}$ una descomposición en subdominios del dominio Ω y asumiendo condiciones de frontera tipo Dirichlet igual a cero, i.e.

$$\begin{aligned} \mathcal{L}u &= -\Delta u + u, & \text{en } \Omega_1 \text{ y } \Omega_2 \\ u &= 0, & \text{en } \partial\Omega. \end{aligned} \quad (9.85)$$

Entonces tenemos la siguiente fórmula de Green-Herrera

$$\begin{aligned} G(u, w) &= \int_{\Omega} w \mathcal{L}u d\underline{x} + \int_{\Gamma} \left\{ [[u]] \widehat{\frac{\partial w}{\partial n}} - \dot{w} \left[\left[\frac{\partial u}{\partial n} \right] \right] \right\} d\underline{x} = \\ &= \int_{\Omega} u \mathcal{L}w d\underline{x} + \int_{\Gamma} \left\{ [[w]] \widehat{\frac{\partial u}{\partial n}} - \dot{u} \left[\left[\frac{\partial w}{\partial n} \right] \right] \right\} d\underline{x} = G(w, u) \end{aligned} \quad (9.86)$$

con las siguientes propiedades

$$G(u, w) = \int_{\Omega} \{\nabla u \cdot \nabla w + uw\} d\underline{x} + \int_{\Gamma} \left\{ [[u]] \widehat{\frac{\partial w}{\partial n}} + [[w]] \widehat{\frac{\partial u}{\partial n}} \right\} d\underline{x} \quad (9.87)$$

y

$$\begin{aligned} \int_{\Omega} \{\nabla u \cdot \nabla w + uw\} d\underline{x} &= \int_{\Omega} w \mathcal{L}u d\underline{x} - \int_{\Gamma} \left\{ \dot{w} \left[\left[\frac{\partial u}{\partial n} \right] \right] + [[w]] \widehat{\frac{\partial u}{\partial n}} \right\} d\underline{x} \\ &= \int_{\Omega} u \mathcal{L}w d\underline{x} - \int_{\Gamma} \left\{ \dot{u} \left[\left[\frac{\partial w}{\partial n} \right] \right] + [[u]] \widehat{\frac{\partial w}{\partial n}} \right\} d\underline{x} \end{aligned} \quad (9.88)$$

reescribiendo estas en términos de funciones discontinuas, se tiene que

$$\begin{aligned} -\Delta u + u &= f_{\Omega}, & \text{en } \Omega_1 \text{ y } \Omega_2 \\ u &= 0, & \text{en } \partial\Omega. \end{aligned} \quad (9.89)$$

$$\left. \begin{aligned} [[u]] &= 0 \\ \left[\left[\frac{\partial u}{\partial n} \right] \right] &= 0 \end{aligned} \right\}, \quad \text{en } \Gamma$$

cuya formulación débil es: u es solución si y sólo si

$$G(u, w) = \int_{\Omega} w f_{\Omega} d\underline{x}, \quad \text{para toda } w \in \hat{H}(\Omega). \quad (9.90)$$

considerando ahora a las funciones armónicas definidas como

$$\mathcal{L}u = 0, \quad \text{en } \Omega_1 \text{ y } \Omega_2. \quad (9.91)$$

Entonces

$$G(u, w) = \int_{\Gamma} \left\{ [[u]] \frac{\widehat{\partial w}}{\partial n} - \dot{w} \left[\left[\frac{\partial u}{\partial n} \right] \right] \right\} d\underline{x} = \int_{\Gamma} \left\{ [[w]] \frac{\widehat{\partial u}}{\partial n} - \dot{u} \left[\left[\frac{\partial w}{\partial n} \right] \right] \right\} d\underline{x} \quad (9.92)$$

con las siguientes propiedades

$$G(u, w) = \int_{\Omega} \{ \nabla u \cdot \nabla w + uw \} d\underline{x} + \int_{\Gamma} \left\{ [[u]] \frac{\widehat{\partial w}}{\partial n} + [[w]] \frac{\widehat{\partial u}}{\partial n} \right\} d\underline{x} \quad (9.93)$$

y

$$\begin{aligned} \int_{\Omega} \{ \nabla u \cdot \nabla w + uw \} d\underline{x} &= - \int_{\Gamma} \left\{ \dot{w} \left[\left[\frac{\partial u}{\partial n} \right] \right] + [[w]] \frac{\widehat{\partial u}}{\partial n} \right\} d\underline{x} = \\ &= - \int_{\Gamma} \left\{ \dot{u} \left[\left[\frac{\partial w}{\partial n} \right] \right] + [[u]] \frac{\widehat{\partial w}}{\partial n} \right\} d\underline{x} \end{aligned} \quad (9.94)$$

expresando estas últimas como una formulación armónica usando funciones discontinuas tenemos

$$\begin{aligned} -\Delta u + u &= f_{\Omega}, & \text{en } \Omega_1 \text{ y } \Omega_2 \\ u &= 0, & \text{en } \partial\Omega. \end{aligned} \quad (9.95)$$

$$\left. \begin{aligned} [[u]] &= j_{\Gamma}^0 \\ \left[\left[\frac{\partial u}{\partial n} \right] \right] &= j_{\Gamma}^1 \end{aligned} \right\}, \quad \text{en } \Gamma$$

donde j_{Γ}^0 y j_{Γ}^1 son dadas, cuya formulación débil es: u es solución si y sólo si

$$G(u, w) = \int_{\Gamma} \left\{ j_{\Gamma}^0 \frac{\widehat{\partial w}}{\partial n} + j_{\Gamma}^1 \dot{w} \right\} d\underline{x}, \quad \text{para toda } w \in \hat{H}(\Omega). \quad (9.96)$$

Definición 88 Sea \bar{D} el espacio donde la solución u es buscada y se define como

$$\bar{D} = \{v \in H^2(\Omega) \mid \text{traza } v = 0, \quad \text{sobre } \partial\Omega\} \quad (9.97)$$

así también definimos al espacio \tilde{D} como

$$\tilde{D} = \{v \in \hat{H}^2(\Omega, \Pi) \mid \text{traza } v = 0, \quad \text{sobre } \partial\Omega\}. \quad (9.98)$$

Definición 89 Definimos al espacio de las funciones armónicas como

$$D \equiv \left\{ u \in \tilde{D} \mid -\Delta u = 0, \quad \text{en } \Omega_\alpha; \alpha = 1, 2, \dots, E \right\}. \quad (9.99)$$

Definición 90 Usando el espacio de las funciones armónicas definimos ahora los siguientes subespacios

$$D_{11} \equiv \{w \in D \mid \dot{w} = 0, \quad \text{en } \Gamma\} \quad (9.100)$$

$$D_{12} \equiv \{w \in D \mid [[w]] = 0, \quad \text{en } \Gamma\}$$

$$D_{21} \equiv \left\{ w \in D \mid \widehat{\frac{\partial w}{\partial n}} = 0, \quad \text{en } \Gamma \right\}$$

$$D_{22} \equiv \left\{ w \in D \mid \left[\left[\frac{\partial w}{\partial n} \right] \right] = 0, \quad \text{en } \Gamma \right\}$$

donde $D_{11} \perp D_{12}$ y $D_{21} \perp D_{22}$ en el producto interior euclidiano y $D_{22} \perp D_{12}$ y $D_{11} \perp D_{21}$ en el producto interior de energía.

Entonces, cada función $w \in D$ puede ser escrita de una única forma como

$$w = w_{11} + w_{12} \text{ con } w_{11} \in D_{11} \text{ y } w_{12} \in D_{12} \quad (9.101)$$

además, D_{21} y D_{11} como también D_{22} y D_{12} son ortogonales con respecto al producto interior

$$u \cdot w \equiv \sum_{\alpha=1}^2 \int_{\Omega_\alpha} \nabla w \cdot \nabla u dx \quad (9.102)$$

esto puede verse usando la relación

$$\sum_{\alpha=1}^2 \int_{\Omega_\alpha} \nabla w \cdot \nabla u dx = - \int_{\Gamma} \left\{ \dot{w} \left[\left[\frac{\partial u}{\partial n} \right] \right] + [[w]] \widehat{\frac{\partial u}{\partial n}} \right\} dx \quad (9.103)$$

la cual se demuestra usando la Ec.(7.20). La transformación $\tau_{22} : D_{12} \rightarrow D_{22}$ es caracterizada por: Dada una función $w \in D$ tal que

$$[[w]] = 0 \text{ sobre } \Gamma \quad (9.104)$$

entonces $(\tau_{22}w) \in D$ es tal que

$$\left[\left[\frac{\partial(\tau_{22}w)}{\partial n} \right] \right] = 0 \text{ y } \frac{\partial(\tau_{22}w)}{\partial n} = \widehat{\frac{\partial w}{\partial n}} \text{ sobre } \Gamma \quad (9.105)$$

similarmente, la transformación $\mu_{22} : D_{22} \rightarrow D_{12}$ es caracterizada por: Dada una función $w \in D$ tal que

$$\left[\left[\frac{\partial w}{\partial n} \right] \right] = 0 \text{ sobre } \Gamma \quad (9.106)$$

entonces $(\mu_{22}w) \in D$ es tal que

$$[[\mu_{22}w]] = 0 \text{ y } \mu_{22}w = \dot{w} \text{ sobre } \Gamma. \quad (9.107)$$

Nótese que la evaluación de $\tau_{22}w$ requiere resolver un problema con condiciones Neumann sobre Γ en cada una de las particiones del subdominio Ω_1 y Ω_2 , mientras que la evaluación de $\mu_{22}w$ requiere resolver un problema con condiciones Dirichlet sobre Γ en cada una de las particiones del subdominio Ω_1 y Ω_2 . Entonces la evaluación de $T_D = -\mu_{22}\tau_{22}$ involucra la resolución de un problema Neumann seguido de uno Dirichlet, mientras que la evaluación de $T_N = -\tau_{22}\mu_{22}$ involucra la resolución de un problema Dirichlet seguido de un problema Neumann.

9.4 Procedimiento para Evaluar la Transformación de Componentes

Para aplicar el esquema general de la sección anterior, necesitamos un procedimiento efectivo de evaluación de las transformaciones

$$\tau_{\alpha\beta} : D_{1\alpha} \rightarrow D_{2\beta} \quad \text{y} \quad \mu_{\alpha\beta} : D_{2\alpha} \rightarrow D_{1\beta} \quad (9.108)$$

requeridas. Esto queda totalmente establecido si $\mu_{\alpha\beta} \in D_{\alpha\beta}$, $\alpha, \beta = 1, 2$ tal que

$$u = u_{11} + u_{12} = u_{21} + u_{22} \quad (9.109)$$

y puede ser evaluado efectivamente cuando $u \in D$ es dado.

Considerando las suposiciones de la sección anterior, entonces existen subconjuntos linealmente independientes

$$\left. \begin{array}{lll} \mathcal{B} \subset \tilde{D}, & \mathcal{B}_I \subset \tilde{D}_I, & \mathcal{B}_\Gamma \subset \tilde{D}_\Gamma \\ \bar{\mathcal{B}}_\Gamma \subset \tilde{D}_\Gamma, & \mathcal{B}_{\Gamma M} \subset \tilde{D}_{\Gamma_2}, & \mathcal{B}_{\Gamma J} \subset \tilde{D}_{\Gamma_1} \end{array} \right\} \quad (9.110)$$

los cuales satisfacen

$$\mathcal{B} = \mathcal{B}_I \cup \mathcal{B}_\Gamma \quad \text{y} \quad \bar{\mathcal{B}}_\Gamma = \mathcal{B}_{\Gamma M} \cup \mathcal{B}_{\Gamma J} \quad (9.111)$$

el espacio generado por cada uno de los subconjuntos \mathcal{B}_Γ y $\bar{\mathcal{B}}_\Gamma$, es \tilde{D}_Γ , sin embargo una propiedad distintiva de \mathcal{B}_Γ es que sus miembros tienen soporte local, a continuación se detalla un procedimiento para calcular $u_{11} \in D_{11}$ y $u_{12} \in D_{12}$.

De acuerdo a la Ec.(9.51), podemos escribir

$$u = \tilde{u}_j + \tilde{u}_M + \tilde{u}_I \quad (9.112)$$

donde $\tilde{u}_j \in \tilde{D}_{\Gamma_1}$, $\tilde{u}_M \in \tilde{D}_{\Gamma_2}$ y $\tilde{u}_I \in \tilde{D}_I$, además

$$\begin{aligned} u_{11} &= (\tilde{u}_{11})_J + (\tilde{u}_{11})_I, \text{ donde } (\tilde{u}_{11})_J \in \tilde{D}_{\Gamma_1} \text{ y } (\tilde{u}_{11})_I \in \tilde{D}_I \\ u_{12} &= (\tilde{u}_{12})_M + (\tilde{u}_{12})_I, \text{ donde } (\tilde{u}_{12})_M \in \tilde{D}_{\Gamma_2} \text{ y } (\tilde{u}_{12})_I \in \tilde{D}_I \end{aligned} \quad (9.113)$$

las Ecs.(9.109) y (9.113) juntas implican que

$$(\tilde{u}_{11})_J = \tilde{u}_J \quad \text{y} \quad (\tilde{u}_{12})_M = \tilde{u}_M \quad (9.114)$$

por lo tanto

$$\begin{aligned} u_{11} &= \tilde{u}_J + (\tilde{u}_{11})_I \text{ donde } (\tilde{u}_{11})_I \in \tilde{D}_I \\ u_{12} &= \tilde{u}_M + (\tilde{u}_{12})_I, \text{ donde } (\tilde{u}_{12})_I \in \tilde{D}_I \end{aligned} \quad (9.115)$$

entonces $(\tilde{u}_{11})_I \in \tilde{D}_I$ puede ser determinada por el sistema de ecuaciones

$$(\tilde{u}_{11})_I \cdot \tilde{w} = -\tilde{u}_J \cdot \tilde{w}, \quad \forall \tilde{w} \in \mathcal{B}_I \quad (9.116)$$

mientras $(\tilde{u}_{12})_I \in \tilde{D}_I$ se determina por

$$(\tilde{u}_{12})_I \cdot \tilde{w} = -\tilde{u}_M \cdot \tilde{w}, \quad \forall \tilde{w} \in \mathcal{B}_I \quad (9.117)$$

ya que $u_{11} \in D$ y $u_{12} \in D$ son ortogonales a \tilde{D}_I . Cada una de las Ecs(9.116) y (9.117) constituyen una sistema de "E" problemas locales independientes.

Por otro lado, cuando $u \in D$ es dado, la función $u_{21} \in D_{21}$ es caracterizada por

$$\begin{aligned} u_{21} \cdot \tilde{w} &= 0, \quad \forall \tilde{w} \in \tilde{D}_I & (9.118) \\ (u_{21} - u) \cdot \tilde{w} &= 0, \quad \forall \tilde{w} \in D_{12} \\ u_{21} \cdot \tilde{w} &= 0, \quad \forall \tilde{w} \in D_{11} \end{aligned}$$

usando el hecho de que cualquier $w \in D_{12}$ es

$$w = \tilde{w}_M + \tilde{w}_I, \text{ donde } \tilde{w}_M \in \tilde{D}_{\Gamma_2} \text{ y } \tilde{w}_I \in \tilde{D}_I \quad (9.119)$$

y que cada $w \in D_{11}$ es

$$w = \tilde{w}_J + \tilde{w}_I, \text{ donde } \tilde{w}_J \in \tilde{D}_{\Gamma_1} \text{ y } \tilde{w}_I \in \tilde{D}_I \quad (9.120)$$

y se ve en el sistema de Ecs.(9.118) es equivalente a

$$\begin{aligned} u_{21} \cdot \tilde{w}_I &= 0, \quad \forall \tilde{w}_I \in \tilde{D}_I & (9.121) \\ u_{21} \cdot \tilde{w}_M &= u \cdot \tilde{w}_M, \quad \forall \tilde{w}_M \in \tilde{D}_{\Gamma_2} \\ u_{21} \cdot \tilde{w}_J &= 0, \quad \forall \tilde{w}_J \in \tilde{D}_{\Gamma_1} \end{aligned}$$

además, $\tilde{D}_\Gamma = \tilde{D}_{\Gamma_1} + \tilde{D}_{\Gamma_2}$ y por lo tanto las Ecs.(9.121) se satisface si y sólo si

$$\begin{aligned} u_{21} \cdot \tilde{w}_I &= 0, \quad \forall \tilde{w}_I \in \tilde{D}_I & (9.122) \\ u_{21} \cdot \tilde{w}_\Gamma &= u \cdot (\tilde{w}_\Gamma)_M, \quad \forall \tilde{w} \in \tilde{D}_\Gamma \end{aligned}$$

aquí, se sobreentiende que cada $\tilde{w}_\Gamma \in \tilde{D}_\Gamma$ se puede escribir como

$$\tilde{w}_\Gamma = (\tilde{w}_\Gamma)_J + (\tilde{w}_\Gamma)_M, \text{ con } (\tilde{w}_\Gamma)_J \in \tilde{D}_{\Gamma_1} \text{ y } (\tilde{w}_\Gamma)_M \in \tilde{D}_{\Gamma_2} \quad (9.123)$$

finalmente, introduciendo las bases \mathcal{B}_I y \mathcal{B}_Γ de \tilde{D}_I y \tilde{D}_Γ respectivamente, las Ecs.(9.122) pueden ser reemplazadas por

$$\begin{aligned} u_{21} \cdot \tilde{w}_I &= 0, \quad \forall \tilde{w}_I \in \mathcal{B}_I & (9.124) \\ u_{21} \cdot \tilde{w}_\Gamma &= u \cdot (\tilde{w}_\Gamma)_M, \quad \forall \tilde{w}_\Gamma \in \mathcal{B}_\Gamma \end{aligned}$$

usando el hecho de que todas las funciones de $\tilde{w}_\Gamma \in \mathcal{B}_\Gamma$ tienen soporte local, se ve que las Ecs.(9.124) constituyen un sistema de "E" ecuaciones locales independientes. De forma similar, se muestra que $u_{22} \in D_{22}$ satisface

$$\begin{aligned} u_{22} \cdot \tilde{w}_I &= 0, \quad \forall \tilde{w}_I \in \mathcal{B}_I \\ u_{22} \cdot \tilde{w}_\Gamma &= u \cdot (\tilde{w}_\Gamma), \quad \forall \tilde{w}_\Gamma \in \mathcal{B}_\Gamma \end{aligned} \quad (9.125)$$

aquí, nuevamente las Ecs.(9.125) constituyen un sistema de "E" ecuaciones locales independientes. Finalmente, observamos que generalmente solo una de los dos sistemas (9.124) y (9.125) necesitan ser resueltos ya que $u = u_{21} + u_{22}$. Un comentario similar aplica en el caso del par de sistemas (9.116) y (9.117).

Aplicando el algoritmo de Gradiente Conjugado de la sección (9.3), también requieren el cálculo de u_{11} o de u_{21} . La versión discreta estándar del problema original, usando funciones continuas exclusivamente es: Encontrar $\bar{u} \in \bar{D}$ tal que

$$\bar{u} \cdot \bar{w} = \int_{\Omega} \bar{w} f_{\Omega} dx, \quad \forall \bar{w} \in \bar{D} \quad (9.126)$$

es equivalente a: Encontrar $\bar{u} \in \bar{D}$ tal que

$$\begin{aligned} \bar{u} \cdot \tilde{w}_I &= \int_{\Omega} \tilde{w}_I f_{\Omega} dx, \quad \forall \tilde{w}_I \in \tilde{D}_I \\ \bar{u} \cdot \tilde{w}_M &= \int_{\Omega} \tilde{w}_M f_{\Omega} dx, \quad \forall \tilde{w}_M \in \tilde{D}_{\Gamma_2} \\ (\bar{u})_J &= 0 \end{aligned} \quad (9.127)$$

sea $\tilde{u}_p \in \tilde{D}$ cualquier función que satisfice

$$\begin{cases} \tilde{u}_p \cdot \tilde{w}_I = \int_{\Omega} \tilde{w}_I f_{\Omega} dx, \quad \forall \tilde{w}_I \in \tilde{D}_I \\ (\tilde{u}_p)_J = 0 \end{cases} \quad (9.128)$$

y definiendo $u = \bar{u} - \tilde{u}_p$, entonces tenemos que

$$\begin{cases} u \in D \\ u \cdot \tilde{w}_M = \int_{\Omega} \tilde{w}_M f_{\Omega} dx - \tilde{u}_p \cdot \tilde{w}_M, \quad \forall \tilde{w}_M \in \tilde{D}_{\Gamma_2} \\ \tilde{u}_J = -(\tilde{u}_p)_J = 0 \end{cases} \quad (9.129)$$

entonces, $u \in D_{12}$ y aplicando Ecs.(9.124) para obtener u_{21} . Una segunda opción es definir $\tilde{u}_p \in \tilde{D}$ como una función que satisfice

$$\begin{cases} \tilde{u}_p \cdot \tilde{w}_I = \int_{\Omega} \tilde{w}_I f_{\Omega} dx, \quad \forall \tilde{w}_I \in \tilde{D}_I \\ \tilde{u}_p \cdot \tilde{w}_\Gamma = \int_{\Omega} \tilde{w}_\Gamma f_{\Omega} dx, \quad \forall \tilde{w}_\Gamma \in \tilde{D}_\Gamma \end{cases} \quad (9.130)$$

en este caso

$$u \cdot \tilde{w}_M = 0, \forall \tilde{w}_M \in \tilde{D}_{\Gamma_2} \text{ y } \tilde{u}_J = -(\tilde{u}_p)_J \quad (9.131)$$

tal que $u \in D_{22}$, en virtud de que las Ecs.(9.122), mientras que las Ecs.(9.115) y (9.116) pueden ser usadas para obtener u_{11} .

9.5 Métodos Dual-Primal

Los métodos Dual-Primal son procedimientos que permiten tratar con particiones de vértices. La idea básica de tales métodos consiste en mantener sin dividir las funciones asociadas con los vértices y tratarlas como nodos internos. Un efecto de tal procedimiento es, sin embargo, un acoplamiento de los sistemas de ecuaciones correspondientes a la partición de los subdominios que comparten un vértice, lo cual puede ser un inconveniente en algunas circunstancias. Por completes, en esta sección se incorpora los métodos Dual-Primal en nuestro esquema.

La colección de conjuntos $\{\mathcal{B}_\Gamma^1, \dots, \mathcal{B}_\Gamma^{\bar{N}_\Gamma}\} \subset \{\mathcal{B}^1, \dots, \mathcal{B}^{\bar{N}}\}$ de la sección (9.2) es dividido en dos subfamilias

$$\{\mathcal{B}_\Delta^1, \dots, \mathcal{B}_\Delta^{N_\Delta}\} \subset \{\mathcal{B}_\Gamma^1, \dots, \mathcal{B}_\Gamma^{\bar{N}_\Gamma}\} \text{ y } \{\mathcal{B}_\Pi^1, \dots, \mathcal{B}_\Pi^{N_\Pi}\} \subset \{\mathcal{B}_\Gamma^1, \dots, \mathcal{B}_\Gamma^{\bar{N}_\Gamma}\}$$

ellas son definidas por las siguientes condiciones: $\mathcal{B}_\Gamma^i \in \{\mathcal{B}_\Delta^1, \dots, \mathcal{B}_\Delta^{N_\Delta}\}$ si y sólo si la cardinalidad de \mathcal{B}_Γ^i es dos, y $\mathcal{B}_\Gamma^i \in \{\mathcal{B}_\Pi^1, \dots, \mathcal{B}_\Pi^{N_\Pi}\}$ si y sólo si la cardinalidad de \mathcal{B}_Γ^i es mayor de dos. Uno de estos conjuntos $\{\mathcal{B}_\Delta^1, \dots, \mathcal{B}_\Delta^{N_\Delta}\}$ es llamado en conjunto "Dual" y el otro conjunto $\{\mathcal{B}_\Pi^1, \dots, \mathcal{B}_\Pi^{N_\Pi}\}$ es llamado el conjunto "Primal". Claramente las funciones del conjunto primal corresponden a las funciones asociadas con los vértices.

Sea \mathcal{B}_Π^i el conjunto primal, entonces definimos $\bar{\mathcal{B}}_\Pi^i = \{\tilde{w}_M\}$ donde $\tilde{w}_M^i \in \bar{D}$ es una función madre del conjunto primal \mathcal{B}_Π^i . Además

$$\mathcal{B}_\Pi = \bigcup_{i=1}^{\bar{N}_\Pi} \bar{\mathcal{B}}_\Pi^i = \{\tilde{w}_M^1, \dots, \tilde{w}_M^{\bar{N}_\Pi}\} \quad (9.132)$$

y

$$\mathcal{B}_\mathcal{J} = \mathcal{B}_\Pi \cup \mathcal{B}_I \quad (9.133)$$

por otro lado, cuando la cardinalidad de \mathcal{B}_Γ^i es dos, definimos

$$\mathcal{B}_\Delta = \bigcup_{i=1}^{N_\Delta} \mathcal{B}_\Delta^i, \bar{\mathcal{B}}_\Pi^i = \{w_M^i, w_J^i\}, \bar{\mathcal{B}}_\Delta = \bigcup_{i=1}^{N_\Delta} \bar{\mathcal{B}}_\Delta^i \quad (9.134)$$

$$\mathcal{B} = \mathcal{B}_\Delta \cup \mathcal{B}_\mathcal{J} \quad (9.135)$$

como también

$$\mathcal{B}_{\Delta M} = \{w_M^1, \dots, w_M^N\} \text{ y } \mathcal{B}_{\Delta J} = \{w_J^1, \dots, w_J^N\} \quad (9.136)$$

entonces $\mathcal{B}_{\Delta M} \subset \overline{\mathcal{B}} \subset \overline{D}$, cada conjunto $\overline{\mathcal{B}}_\Delta$ y \mathcal{B}_Δ generan el mismo subespacio lineal y tienen una propiedad evidente y es que todos los elementos de \mathcal{B}_Δ tienen soporte local, lo cual no es cierto en $\overline{\mathcal{B}}_\Delta$. Además

$$\overline{\mathcal{B}} = \mathcal{B}_{\Delta M} \cup \mathcal{B}_\mathcal{J}. \quad (9.137)$$

El subespacio generado por el conjunto de funciones $\mathcal{B}, \mathcal{B}_\mathcal{J}, \mathcal{B}_\Delta, \mathcal{B}_{\Delta J}$ y $\mathcal{B}_{\Delta M}$ será denotado por $\tilde{D}, \tilde{D}_\mathcal{J}, \tilde{D}_\Sigma, \tilde{D}_{\Sigma_1}$ y \tilde{D}_{Σ_2} respectivamente. Haciendo las siguientes sustituciones

$$\left. \begin{array}{l} \tilde{D}_\mathcal{J} \rightarrow \tilde{D}_I, \quad \tilde{D}_\Sigma \rightarrow \tilde{D}_\Gamma, \quad \tilde{D}_\mathcal{J} \rightarrow \tilde{D}_\mathcal{J} \\ \tilde{D}_{\Sigma_1} \rightarrow \tilde{D}_{\Gamma_1}, \quad \tilde{D}_{\Sigma_2} \rightarrow \tilde{D}_{\Gamma_2} \end{array} \right\} \quad (9.138)$$

ya que corresponden con las suposiciones hechas anteriormente en este capítulo

$$\left. \begin{array}{l} \tilde{D} = \tilde{D}_\mathcal{J} + \tilde{D}_\Sigma \quad \text{y} \quad \tilde{D}_\mathcal{J} \cap \tilde{D}_\Sigma = \{0\} \\ \tilde{D}_\Sigma = \tilde{D}_{\Sigma_1} + \tilde{D}_{\Sigma_2} \quad \text{y} \quad \tilde{D}_{\Sigma_1} \cap \tilde{D}_{\Sigma_2} = \{0\} \\ \overline{D} = \tilde{D}_{\Sigma_2} + \tilde{D}_\mathcal{J} \end{array} \right\} \quad (9.139)$$

por lo tanto definimos

$$D = \left(\tilde{D}_\mathcal{J} \right)^\perp \quad (9.140)$$

como se había hecho anteriormente, una vez $D \subset \tilde{D}$ ha sido definido, el complemento ortogonal puede ser tomado con respecto a D . Y las siguientes definiciones son adoptadas

$$D_{11} = \text{Proy}_D \tilde{D}_{\Sigma_1} \quad \text{y} \quad D_{12} = \text{Proy}_D \tilde{D}_{\Sigma_2} \quad (9.141)$$

$$D_{21} = (D_{11})^\perp \quad \text{y} \quad D_{22} = (D_{12})^\perp \quad (9.142)$$

entonces

$$D = D_{11} + D_{12} \quad \text{y} \quad D_{11} \cap D_{12} = \{0\}. \quad (9.143)$$

La diferencia con respecto a lo antes desarrollado es que todas las funciones del espacio \tilde{D}_I tienen soporte local, el cual no es el caso del espacio $\tilde{D}_\mathcal{J}$ como aquí se definió. Debido a este hecho, tenemos algunos acoplamientos entre los sistemas de ecuaciones correspondientes a diferentes subdominios que comparten vértices.

10 Unificación y Simplificación de los Métodos Dual-Primal de Descomposición de Dominio

Entre los métodos más conocidos de descomposición de dominio esta el método FETI Dual-Primal, este destaca por permitir resolver un gran número de problemas y su solución se basa en separar los nodos del dominio en tres tipos de nodos: Nodos Interiores, Nodos Primal (nodos vértices de la frontera interior del subdominio) y nodos Duales (nodos distintos a los vértices de la frontera interior del subdominio).

La teoría propuesta por el Dr. Herrera [60], [61] y [62] permite hacer una unificación entre diversos métodos, entre los que destacan los de Subestructuración, FETI, FETI-DP, entre otros.

10.1 Espacio Dual-Primal

Sea Ω un conjunto finito de cardinalidad d los cuales por definición son tomados como $\{1, \dots, d\}$ y a cuyos miembros nos referiremos como los *grados de libertad originales*. Además, sea el conjunto $\{\Omega_1, \dots, \Omega_E\}$ una cubierta de Ω , i.e. $\{\Omega_1, \dots, \Omega_E\}$ es una familia de subconjuntos de Ω tal que

$$\Omega = \bigcup_{\alpha=1}^E \Omega_{\alpha}. \quad (10.1)$$

Consideraremos al par $\underline{p} = (p, \alpha)$, tal que $p \in \Omega$ y $\alpha \in \{1, \dots, E\}$, entonces definimos:

Definición 91 *Un nodo derivado es el par $\underline{p} \equiv (p, \alpha)$ tal que $p \in \Omega$ y $\alpha \in \{1, \dots, E\}$. Y el conjunto total de nodos derivados $\bar{\Omega}$ es definido por*

$$\bar{\Omega} \equiv \{\underline{p} = (p, \alpha) \mid p \in \Omega_{\alpha}\}. \quad (10.2)$$

Definición 92 *Para todo $p \in \Omega$, escribimos*

$$Z(p) \equiv \{\alpha \in \{1, \dots, E\} \mid (p, \alpha) \in \bar{\Omega}\}. \quad (10.3)$$

Definición 93 *Para todo $p \in \Omega$, la multiplicidad total de p , $m(p)$, es definido por la cardinalidad de $Z(p)$.*

Distinguiremos dos clases de nodos originales:

Definición 94 $p \in \Omega$ será llamado *nodo interior* si $m(p) = 1$, y lo llamaremos *nodo de la frontera interior* cuando $m(p) > 1$.

Distinguiremos dos clases de nodos derivados $\underline{p} \equiv (p, \alpha) \in \bar{\Omega}$, estos son:

Definición 95 Llamaremos a los nodos derivados por *nodo interior (derivado)* o *nodo de frontera (derivado)* dependiendo de que su multiplicidad sea igual a uno o más grande que uno, respectivamente.

El conjunto de nodos interiores y nodos de frontera son disjuntos, los cuales los denotaremos por I y Γ respectivamente, claramente el par $\{I, \Gamma\}$ constituyen una partición de Ω , ya que

$$\Omega = I \cup \Gamma \quad \text{y} \quad \emptyset = I \cap \Gamma. \quad (10.4)$$

Escogeremos a el conjunto $\Omega^\pi \subset \Omega^\Gamma$ y definimos los conjuntos

$$\begin{cases} I \equiv \{\underline{p} = (p, \alpha) \in \bar{\Omega} \mid p \in \Omega^I\} \\ \pi \equiv \{\underline{p} = (p, 0) \in \bar{\Omega} \mid p \in \Omega^\pi\} \\ \Delta \equiv \{\underline{p} = (p, \alpha) \in \bar{\Omega} \mid p \in \Omega^\Gamma - \Omega^\pi\} \end{cases} \quad (10.5)$$

Claramente $\Delta = \Gamma - \pi$, además, definimos $\Pi \equiv I \cup \pi$ y observamos que

$$\bar{\Omega} = I \cup \pi \cup \Delta = \Pi \cup \Delta \quad \text{y} \quad \emptyset = \Pi \cap \Delta = \pi \cap \Delta = \pi \cap I = \Delta \cap I. \quad (10.6)$$

Definición 96 A los elementos $p \in \bar{\Omega}$, les llamaremos *nodos derivados*, los cuales pueden ser *nodos interiores, primales* o *duales* dependiendo de que si pertenecen a I , π o a Δ respectivamente.

Observemos que $\Delta = \Gamma$ cuando $\pi = \emptyset$.

Definición 97 Para todo $p \in \Omega$, tenemos el conjunto

$$Z(p) \equiv \{\alpha \in \{0, 1, \dots, E\} \mid (p, \alpha) \in \bar{\Omega}\} \quad (10.7)$$

entonces definiremos la multiplicidad de $m(p)$ de cualquier $p \in \Omega$, como la cardinalidad de $Z(p)$, en particular, $m(p) = 1$ si y sólo si $p \in \Pi$.

10.2 Espacio de Vectores

Notemos que cada función real-valuada definida en Ω o en $\bar{\Omega}$ es un vector (y en lo que resta de este trabajo no haremos distinción entre una función y vector).

Definición 98 *Los espacios lineales $\tilde{D}(\Omega)$ y $\tilde{D}(\bar{\Omega})$ están constituidos por funciones (vectores) definidos en Ω y $\bar{\Omega}$ respectivamente. Similarmente, $\tilde{D}(\Pi) \subset \tilde{D}(\bar{\Omega})$ y $\tilde{D}(\Delta) \subset \tilde{D}(\bar{\Omega})$ serán subespacios lineales de $\tilde{D}(\bar{\Omega})$ cuyos elementos se nulifican fuera de Π y Δ respectivamente. Los subespacios $\tilde{D}(I)$, $\tilde{D}(\pi)$ y $\tilde{D}(\Gamma)$ de $\tilde{D}(\bar{\Omega})$ son definidas de forma similar.*

Entonces

$$\tilde{D}(\bar{\Omega}) = \tilde{D}(\Pi) \oplus \tilde{D}(\Delta) \quad (10.8)$$

donde \oplus es la suma directa de dos espacios lineales. Esta última ecuación se satisface si y sólo si

$$\begin{cases} \tilde{D}(\bar{\Omega}) = \tilde{D}(\Pi) + \tilde{D}(\Delta) \\ \{0\} = \tilde{D}(\Pi) \cap \tilde{D}(\Delta) \end{cases} \quad (10.9)$$

por lo tanto, los vectores de $\tilde{D}(\bar{\Omega})$ pueden representarse de una única manera como

$$\underline{u} = (u_\Pi, u_\Delta) = u_\Pi + u_\Delta \quad \text{con } u_\Pi \in \tilde{D}(\Pi) \text{ y } u_\Delta \in \tilde{D}(\Delta). \quad (10.10)$$

La inmersión natural de $\tilde{D}(\Omega)$ dentro de $\tilde{D}(\bar{\Omega})$, será denotada por $\tau : \tilde{D}(\Omega) \rightarrow \tilde{D}(\bar{\Omega})$, la cual es definida para cada $\tilde{u} \in \tilde{D}(\Omega)$ por

$$(\tau \tilde{u})(\underline{q}) = \tilde{u}(p), \quad \forall \underline{q} \in Z(p) \subset \bar{\Omega} \quad (10.11)$$

más explícitamente, se tiene que

$$(\tau \tilde{u})_{(p,\alpha)} = \tilde{u}(p), \quad \forall (p, \alpha) \in \bar{\Omega}. \quad (10.12)$$

La imagen $\tau \tilde{D}(\Omega)$ de $\tilde{D}(\Omega)$ under $\tau : \tilde{D}(\Omega) \rightarrow \tilde{D}(\bar{\Omega})$ constituye un subespacio lineal de $\tilde{D}(\bar{\Omega})$. Definimos

$$\bar{D}(\bar{\Omega}) \equiv \tau \tilde{D}(\Omega) \subset \tilde{D}(\bar{\Omega}). \quad (10.13)$$

Definición 99 Un vector $\underline{u} \in \tilde{D}(\overline{\Omega})$ es llamado continuo cuando para todo $p \in \Omega$, se tiene que $u(p, \alpha)$ es independiente de α .

Observemos que cuando $(p, \alpha) \in \Pi$, entonces la cardinalidad de $Z(p)$ es uno; por lo tanto

$$\tilde{D}(\Pi) \subset \bar{D}(\overline{\Omega}). \quad (10.14)$$

Claramente, la función $\tau : \tilde{D}(\Omega) \rightarrow \tilde{D}(\overline{\Omega})$ es una biyección; esto permite definir $\tau^{-1} : \tilde{D}(\overline{\Omega}) \rightarrow \tilde{D}(\Omega)$ la cual tiene la propiedad de que para todo $\tilde{u} \in \tilde{D}(\overline{\Omega})$

$$(\tau^{-1}\tilde{u})(\underline{p}) = \underline{u}(p, \alpha), \quad \forall \underline{p} \in \Omega \text{ y } \alpha \in Z(p). \quad (10.15)$$

10.3 El Producto Interior Euclidiano

Definición 100 El producto interior Euclidiano es definido por

$$\begin{cases} \underline{\hat{u}} \cdot \underline{\hat{w}} \equiv \sum_{p \in \Omega} \underline{\hat{u}}(p) \underline{\hat{w}}(p), \quad \forall \underline{\hat{u}}, \underline{\hat{w}} \in \tilde{D}(\Omega) \\ \underline{u} \cdot \underline{w} \equiv \sum_{p \in \overline{\Omega}} \underline{u}(\underline{p}) \underline{w}(\underline{p}) = \sum_{q \in \Omega} \sum_{p \in Z(q)} \underline{u}(\underline{p}) \underline{w}(\underline{p}), \quad \forall \underline{u}, \underline{w} \in \tilde{D}(\overline{\Omega}). \end{cases} \quad (10.16)$$

Definición 101 En sistemas de ecuaciones diferenciales parciales, se requieren usar matrices cuyo tratamiento adecuado en nuestro esquema requiere la introducción de funciones vector valuadas. En cuyo caso $\underline{\hat{u}}(p)$ y $\underline{u}(p)$ deben de ser vectores, así que las Ecs(10.16) que definen un producto interior deben de ser reemplazadas por

$$\begin{cases} \underline{\hat{u}} \cdot \underline{\hat{w}} \equiv \sum_{p \in \Omega} \underline{\hat{u}}(p) \odot \underline{\hat{w}}(p), \quad \forall \underline{\hat{u}}, \underline{\hat{w}} \in \tilde{D}(\Omega) \\ \underline{u} \cdot \underline{w} \equiv \sum_{p \in \overline{\Omega}} \underline{u}(\underline{p}) \odot \underline{w}(\underline{p}) = \sum_{q \in \Omega} \sum_{p \in Z(q)} \underline{u}(\underline{p}) \odot \underline{w}(\underline{p}), \quad \forall \underline{u}, \underline{w} \in \tilde{D}(\overline{\Omega}). \end{cases} \quad (10.17)$$

en este caso el símbolo \odot denota al producto interior del espacio de vectores al que pertenezca $\underline{\hat{u}}(p)$ y $\underline{u}(p)$.

Dos matrices auxiliares introduciremos a continuación, ellas son $\underline{\hat{m}} : \tilde{D}(\Omega) \rightarrow \tilde{D}(\Omega)$ y $\underline{m} : \tilde{D}(\Omega) \rightarrow \tilde{D}(\Omega)$, las cuales son definidas para cada $\underline{\hat{u}} \in \tilde{D}(\Omega)$ y cada $\underline{u} \in \tilde{D}(\Omega)$ por medio de:

Definición 102 Sean $\widehat{\underline{m}}: \tilde{D}(\Omega) \rightarrow \tilde{D}(\Omega)$ y $\underline{m}: \tilde{D}(\Omega) \rightarrow \tilde{D}(\Omega)$ las matrices diagonales, tales que para cada $\hat{\underline{u}} \in \tilde{D}(\Omega)$ y cada $\underline{u} \in \tilde{D}(\Omega)$ tenemos

$$\begin{aligned}\widehat{\underline{m}}\hat{\underline{u}}(p) &= m(p)\hat{\underline{u}}(p), & \forall p \in \Omega \\ \underline{m}\underline{u}(p) &= m(p)\underline{u}(p), & \forall p = (p, \alpha) \in \bar{\Omega}\end{aligned}\quad (10.18)$$

respectivamente.

El valor de la diagonal principal de $\widehat{\underline{m}}$ y \underline{m} son las multiplicidades de $m(p)$. Algunos resultados son mostrados a continuación

$$\tau\widehat{\underline{m}}\hat{\underline{u}} = \widehat{\underline{m}}\tau\hat{\underline{u}} \quad \text{y} \quad \tau\underline{m}^{-1}\hat{\underline{u}} = \widehat{\underline{m}}^{-1}\tau\hat{\underline{u}}, \quad \forall \hat{\underline{u}} \in \tilde{D}(\Omega) \quad (10.19)$$

junto con

$$\underline{m}\bar{D}(\Omega) = \bar{D}(\Omega) = \underline{m}^{-1}\bar{D}(\Omega). \quad (10.20)$$

Lema 103 Cuando $\hat{\underline{u}}, \hat{\underline{w}} \in \tilde{D}(\Omega)$, entonces, los siguientes enunciados se satisfacen

$$\left. \begin{aligned}\hat{\underline{u}} \cdot \widehat{\underline{m}}\hat{\underline{w}} &= \tau(\hat{\underline{u}}) \cdot \tau(\hat{\underline{w}}) \\ \hat{\underline{u}} \cdot \hat{\underline{w}} &= \tau(\hat{\underline{u}}) \cdot \underline{m}^{-1}\tau(\hat{\underline{w}})\end{aligned}\right\}, \quad \forall \hat{\underline{u}}, \hat{\underline{w}} \in \tilde{D}(\Omega). \quad (10.21)$$

Demostración. Escribimos $\underline{u} = \tau\hat{\underline{u}}$ y $\underline{w} = \tau\hat{\underline{w}}$ y aplicando la Ec.(10.16) y la Ec.(10.11) obtenemos

$$\underline{u} \cdot \underline{w} = \sum_{p \in \Omega} \sum_{q \in Z(p)} \underline{u}(q) \underline{w}(q) = \sum_{p \in \Omega} m(p) \hat{\underline{u}}(p) \hat{\underline{w}}(p) = \hat{\underline{u}} \cdot \widehat{\underline{m}}\hat{\underline{w}} \quad (10.22)$$

esto muestra la primera relación de la Ec.(10.21), entonces aplicando tal relación reemplazando $\hat{\underline{w}}$ por $\underline{m}^{-1}(\hat{\underline{w}})$ la segunda relación es obtenida. ■

Corolario 104 Sea $\underline{u} \in \tau\tilde{D}(\Omega) = \bar{D}(\bar{\Omega})$ tal que para alguna $\hat{\underline{u}} \in D(\Omega)$ se satisfice

$$\hat{\underline{u}} \cdot \hat{\underline{w}} = \underline{u} \cdot \tau(\hat{\underline{w}}), \quad \forall \hat{\underline{w}} \in D(\Omega) \quad (10.23)$$

entonces

$$\underline{u} = \underline{m}^{-1}\tau(\hat{\underline{u}}) = \tau(\widehat{\underline{m}}^{-1}\hat{\underline{u}}). \quad (10.24)$$

Demostración. Como se vio en el lema anterior, tenemos que

$$\hat{\underline{w}} \cdot \hat{\underline{u}} = \tau(\hat{\underline{w}}) \cdot \underline{m}^{-1}\tau(\hat{\underline{u}}), \quad \forall \hat{\underline{w}} \in \tilde{D}(\Omega) \quad (10.25)$$

lo cual implica, que cuando la Ec.(10.23) se satisfice, tenemos

$$(\underline{u} - \underline{m}^{-1}\tau(\hat{\underline{u}})) \cdot \underline{w} = 0, \quad \forall \underline{w} \in \tau\tilde{D}(\Omega) = \bar{D}(\Omega) \quad (10.26)$$

el corolario se sigue de esta última ecuación, ya que ambas \underline{u} y $\underline{m}^{-1}\tau(\hat{\underline{u}})$ pertenecen a $\bar{D}(\Omega)$. ■

10.4 Subespacios de Vectores: Las Matrices de Promedio y de Salto

Dos matrices $\underline{a} : \tilde{D}(\bar{\Omega}) \rightarrow \tilde{D}(\bar{\Omega})$ y $\underline{j} : \tilde{D}(\bar{\Omega}) \rightarrow \tilde{D}(\bar{\Omega})$ son ahora introducidas.

Definición 105 Sea el operador promedio $\underline{a} : \tilde{D}(\bar{\Omega}) \rightarrow \tilde{D}(\bar{\Omega})$ y el operador salto $\underline{j} : \tilde{D}(\bar{\Omega}) \rightarrow \tilde{D}(\bar{\Omega})$ dos matrices definidas por

$$\underline{a}u = \text{Proy}_{\bar{D}}u \quad y \quad \underline{j} = \underline{I} - \underline{a} \quad (10.27)$$

aquí, \underline{I} es la matriz identidad y la proyección sobre \bar{D} es tomada con respecto al producto interior Euclidiano. A tales matrices nos referiremos como las matrices de promedios y la matriz de Salto.

Notemos que $\underline{j}u$ y $\underline{a}u$ son ortogonales, ya que $\underline{j}u = u - \underline{a}u$, ya que el vector $u - \underline{a}u$ es ortogonal a $\underline{a}u$. En vista de esta definición tenemos

$$\bar{D}(\bar{\Omega}) \equiv \underline{a}\tilde{D}(\bar{\Omega}) \quad (10.28)$$

una obvia e importante propiedad es que

$$\underline{I} = \underline{a} + \underline{j} \quad (10.29)$$

además, \underline{j} es también una proyección, verdaderamente, esta es la proyección sobre el complemento ortogonal de \bar{D} . Por lo tanto, \underline{a} y \underline{j} son ambas simétricas, no negativas e idempotentes. También notemos que

$$\underline{a}\underline{j} = \underline{j}\underline{a} = \underline{0} \quad (10.30)$$

ya que $\underline{a}\underline{j}u = (\underline{a}\underline{j}u) \cdot (\underline{j}u) = 0$ pues \underline{a} y \underline{j} son ortogonales, en particular

$$\underline{j}\bar{D}(\bar{\Omega}) = \{0\}. \quad (10.31)$$

La construcción de la matriz \underline{a} es sencilla, escribiendo

$$\underline{a} \equiv (a_{(i,\alpha)(j,\beta)}) \quad (10.32)$$

entonces

$$a_{(i,\alpha)(j,\beta)} = \frac{1}{m(i)} \delta_{ij}, \quad \forall \alpha \in Z(i) \text{ y } \forall \beta \in Z(j). \quad (10.33)$$

Una expresión para la matriz \underline{j} se sigue de la Ec.(10.27), pero para evaluar sobre cualquier vector, es más fácil mediante el uso de

$$\underline{j}\underline{u} = (\underline{I} - \underline{a})\underline{u} = \underline{u} - \underline{a}\underline{u}, \quad \forall \underline{u} \in \tilde{D}(\bar{\Omega}). \quad (10.34)$$

Por ejemplo, la estructura de $\underline{a}^{(q)}$ y $\underline{j}^{(q)}$, donde q es un nodo de multiplicidad 2 queda como

$$\underline{a}^{(q)} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad y \quad \underline{j}^{(q)} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad (10.35)$$

y la estructura de $\underline{a}^{(q)}$ y $\underline{j}^{(q)}$, donde q es un nodo de multiplicidad 4 queda como

$$\underline{a}^{(q)} = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix} \quad y \quad \underline{j}^{(q)} = \begin{bmatrix} \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} \end{bmatrix}. \quad (10.36)$$

Definición 106 *Definimos los subespacios*

$$\begin{cases} \bar{D}_{11}(\bar{\Omega}) \equiv \underline{j}\tilde{D}(\bar{\Omega}) \subset \tilde{D}(\Gamma) \\ \bar{D}_{12}(\bar{\Omega}) \equiv \underline{\underline{D}}(\bar{\Omega}) = \underline{a}\tilde{D}(\bar{\Omega}) \\ \tilde{D}_{11}(\Gamma) \equiv \underline{j}\tilde{D}(\Gamma) = \tilde{D}_{11}(\bar{\Omega}) \\ \tilde{D}_{12}(\Gamma) \equiv \underline{a}\tilde{D}(\Gamma) \end{cases} \quad (10.37)$$

Usando estas definiciones, tenemos las siguientes propiedades de estos subespacios:

- $\tilde{D}_{11}(\bar{\Omega}) = \tilde{D}_{11}(\Gamma)$ es el complemento ortogonal, con respecto al producto interior Euclidiano de $\tilde{D}_{12}(\bar{\Omega})$.

•

$$\begin{aligned} \tilde{D}(\bar{\Omega}) &= \underline{a}\tilde{D}(\bar{\Omega}) \oplus \underline{j}\tilde{D}(\bar{\Omega}) = \bar{D}(\bar{\Omega}) \oplus \underline{j}\tilde{D}(\bar{\Omega}) \\ &= \tilde{D}(\Gamma) \oplus \tilde{D}_{11}(\Gamma) \oplus \tilde{D}_{12}(\Gamma). \end{aligned} \quad (10.38)$$

- $\tilde{D}(\bar{\Omega}) = \tilde{D}_{11}(\Gamma) \oplus \bar{D}(\bar{\Omega})$.
- $\tilde{D}(\Gamma) = \tilde{D}_{11}(\Gamma) \oplus \tilde{D}_{12}(\Gamma)$.
- Además, $\tilde{D}_{11}(\Gamma)$ y $\tilde{D}_{12}(\Gamma)$ son complementos ortogonales relativos a $\tilde{D}(\Gamma)$.
- $\bar{D}(\bar{\Omega}) = \tilde{D}_{12}(I) \oplus \tilde{D}(\Gamma)$.
- $\underline{a}\underline{u} = \underline{u}$ y $\underline{j}\underline{u} = 0$, $\forall \underline{u} \in \tilde{D}(I)$
- $\underline{a}\tilde{D}(I) = \tilde{D}(I)$ y $\underline{j}\tilde{D}(I) = \{0\}$

Otras propiedades implicadas por los resultados anteriores son:

$$\begin{aligned}
 \bar{D}(\bar{\Omega}) &= \left\{ \underline{u} \in \tilde{D}(\bar{\Omega}) \mid \underline{j}\underline{u} = 0 \right\} = \left\{ \underline{u} \in \tilde{D}(\bar{\Omega}) \mid \underline{a}\underline{u} = \underline{u} \right\} \\
 \tilde{D}_{11}(\Delta) &= \left\{ \underline{u} \in \tilde{D}(\bar{\Omega}) \mid \underline{a}\underline{u} = 0 \right\} = \left\{ \underline{u} \in \tilde{D}(\bar{\Omega}) \mid \underline{j}\underline{u} = \underline{u} \right\} \\
 \tilde{D}_{12}(\Delta) &= \left\{ \underline{u} \in \tilde{D}(\Delta) \mid \underline{j}\underline{u} = 0 \right\} = \left\{ \underline{u} \in \tilde{D}(\Delta) \mid \underline{a}\underline{u} = \underline{u} \right\}.
 \end{aligned} \tag{10.39}$$

Nótese que, en vista de la Ec.(10.38) cualquier $\underline{u} \in \tilde{D}(\bar{\Omega})$ puede ser escrita de forma única como

$$\underline{u} = \underline{u}_I + \underline{u}_\Gamma = \underline{u}_I + \underline{u}_{\Gamma_1} + \underline{u}_{\Gamma_2} \tag{10.40}$$

donde $\underline{u}_I \in \tilde{D}(I)$, $\underline{u}_{\Gamma_1} \in \tilde{D}_{11}(\Gamma)$ y $\underline{u}_{\Gamma_2} \in \tilde{D}_{12}(\Gamma)$ y

$$\underline{u} = \underline{u}_\Pi + \underline{u}_\Delta = \underline{u}_\Pi + \underline{u}_{\Delta_1} + \underline{u}_{\Delta_2} \tag{10.41}$$

donde $\underline{u}_\Pi \in \tilde{D}(\Pi)$, $\underline{u}_{\Delta_1} \in \tilde{D}_{11}(\Delta)$ y $\underline{u}_{\Delta_2} \in \tilde{D}_{12}(\Delta)$, además

$$\underline{u}_{\Delta_1} = [[\underline{u}_\Delta]] = [[\underline{u}]], \quad \underline{u}_{\Delta_2} = \hat{\underline{u}}_\Delta \quad \text{y} \quad \underline{u}_\Delta = \underline{u}_{\Delta_1} + \underline{u}_{\Delta_2}. \tag{10.42}$$

10.5 El Subespacio Dual-Primal

Para cada $k \in \Omega$ definimos:

Definición 107 La matriz de salto \underline{j}^k en k es definida como

$$\underline{j}^k \equiv \left(j_{(i,\alpha)(j,\beta)}^k \right) \quad (10.43)$$

donde

$$j_{(i,\alpha)(j,\beta)}^k \equiv \left(\delta_{\alpha\beta} - \frac{1}{m(k)} \right) \delta_{ik} \delta_{jk}. \quad (10.44)$$

Para cualquier vector $\underline{v} \in \tilde{D}(\overline{\Omega})$ la condición de que $\underline{j}^k \underline{v} = 0$ es equivalente a

$$\underline{v}(k, \alpha) = \underline{v}(k, \beta), \quad \forall \alpha, \beta \in Z(k). \quad (10.45)$$

cuando $\underline{j}^k \underline{v} = 0$, decimos que \underline{v} es continuo en el nodo k .

Definición 108 Definimos δ_{ij}^π como

$$\delta_{ij}^\pi \equiv \begin{cases} 1, & \text{si } i, j \in \Omega^\pi \\ 0, & \text{si } i \text{ ó } j \notin \Omega^\pi \end{cases} \quad (10.46)$$

Definición 109 La matriz de salto primal \underline{j}^π es definida como

$$\underline{j}^\pi \equiv \sum_{k \in \Omega^\pi} \underline{j}^k. \quad (10.47)$$

ó

$$j_{(i,\alpha)(j,\beta)}^\pi = \left(\delta_{\alpha\beta} - \frac{1}{m(i)} \right) \delta_{ij} \delta_{ij}^\pi. \quad (10.48)$$

Entonces el espacio dual es definido como:

Definición 110 El espacio dual $\tilde{D}^{DP}(\overline{\Omega})$ es definido como

$$\tilde{D}^{DP}(\overline{\Omega}) \equiv \left\{ \underline{w} \in \tilde{D}(\overline{\Omega}) \mid \underline{j}^\pi \underline{w} = 0 \right\} \subset \tilde{D}(\overline{\Omega}) \quad (10.49)$$

En particular, $\tilde{D}^{DP}(\overline{\Omega}) = \tilde{D}(\overline{\Omega})$ cuando $\Omega^\pi = \emptyset$. Obsérvese que la proyección $\underline{a}^\pi : \tilde{D}(\overline{\Omega}) \rightarrow \tilde{D}^{DP}(\overline{\Omega})$ está dada por

$$\underline{a}^\pi \equiv \underline{I} - \underline{j}^\pi \quad (10.50)$$

por lo tanto

$$a_{(i,\alpha)(j,\beta)}^\pi = \frac{1}{m(i)} \delta_{ij} \delta_{ij}^\pi + \delta_{\alpha\beta} \delta_{ij} (1 - \delta_{ij}^\pi) \quad (10.51)$$

i.e. esta ecuación dice que $\underline{\underline{a}}^\pi$ es igual a la matriz identidad en todo nodo derivado excepto cuando el nodo pertenece al conjunto de los nodos primales π , en cuyo caso es igual a la matriz de promedios dada por la Ec.(10.33). Similarmente, la Ec.(10.48) dice que la *matriz de salto primal* $\underline{\underline{j}}^\pi$ se nulifica en casi todos lados salvo en los nodos primales, donde es igual al operador salto. Por lo tanto, el espacio dual-primal $\tilde{D}^{DP}(\bar{\Omega})$ es el subespacio de $\tilde{D}(\bar{\Omega})$ cuyos elementos son continuos en cada nodo perteneciente a π . Adoptaremos la siguiente notación:

$$\tilde{D}_{11}^{DP}(\bar{\Omega}) \equiv \underline{\underline{j}} \tilde{D}^{DP}(\bar{\Omega}) \subset \tilde{D}^{DP}(\bar{\Omega}) \quad (10.52)$$

y

$$\tilde{D}_{12}^{DP}(\bar{\Omega}) \equiv \underline{\underline{a}} \tilde{D}^{DP}(\bar{\Omega}) = \tilde{D}_{12}(\bar{\Omega}). \quad (10.53)$$

Para probar que $\underline{\underline{j}} \tilde{D}^{DP}(\bar{\Omega}) \subset \tilde{D}^{DP}(\bar{\Omega})$ entonces, dado $\underline{w} \in \tilde{D}^{DP}(\bar{\Omega})$ calculamos la proyección de $\underline{\underline{j}} \underline{w}$ sobre $\tilde{D}^{DP}(\bar{\Omega})$, i.e.

$$\underline{\underline{a}}^\pi \underline{\underline{j}} \underline{w} = \left(\underline{I} - \underline{\underline{j}}^\pi \right) \underline{\underline{j}} \underline{w} = \underline{\underline{j}} \underline{w} - \underline{\underline{j}}^\pi \underline{w} = \underline{\underline{j}} \underline{w}. \quad (10.54)$$

10.6 La Formulación Matricial Discontinua Libre de Multiplicadores de Lagrange

En los que sigue, consideremos varias matrices, entre ellas

$$\hat{\underline{\underline{A}}} : \tilde{D}(\Omega) \rightarrow \tilde{D}(\Omega), \underline{\underline{A}}^t : \tilde{D}(\bar{\Omega}) \rightarrow \tilde{D}(\bar{\Omega}) \text{ y } \underline{\underline{A}} : \tilde{D}(\bar{\Omega}) \rightarrow \tilde{D}^{DP}(\bar{\Omega}) \quad (10.55)$$

la matriz $\hat{\underline{\underline{A}}}$ es generada por alguna discretización y referida como la matriz original, a las matrices $\underline{\underline{A}}^t$ y $\underline{\underline{A}}$ nos referiremos a ellas como la matriz total y la matriz dual primal respectivamente. Escribimos

$$\hat{\underline{\underline{A}}} \equiv \left(\hat{\underline{\underline{A}}}_{pq} \right), \text{ donde } p,q \in \Omega \quad (10.56)$$

y asumiremos en lo que sigue en este trabajo que:

1.- $\hat{\underline{A}} : \tilde{D}(\Omega) \rightarrow \tilde{D}(\Omega)$ es positiva definida.

2.- Usando la Ec.(10.56),

$$\hat{\underline{A}}_{pq} = 0, \text{ siempre y cuando } p \in \Omega^I \cap \Omega_\alpha, q \in \Omega^I \cap \Omega_\beta \text{ y } \alpha \neq \beta. \quad (10.57)$$

3.- La matriz $\underline{A}^t : \tilde{D}(\bar{\Omega}) \rightarrow \tilde{D}(\bar{\Omega})$ es positiva definida y satisface la condición:

$$\hat{w} \cdot \hat{\underline{A}}\hat{u} = \tau(\hat{w}) \cdot \underline{A}^t\tau(\hat{u}), \quad \forall \hat{u}, \hat{w} \in \tilde{D}(\Omega) \quad (10.58)$$

está condición no determina de forma única a \underline{A}^t .

4.- Para cada $\alpha \in \{1, \dots, E\}$ es definida una matriz $\underline{A}^\alpha : \tilde{D}(\bar{\Omega}_\alpha) \rightarrow \tilde{D}(\bar{\Omega}_\alpha)$ tal que

$$\underline{A}^t = \sum_{\alpha=1}^E \underline{A}^\alpha \quad (10.59)$$

un conveniente procedimiento para construir una de estas matrices \underline{A}^t está dada en la sección (10.7). Entonces la matriz \underline{A}^t es positiva definida sobre las funciones continuas.

5.- Cuando \underline{A}^t es dada, la matriz dual-primal \underline{A} es definida por

$$\underline{A} \equiv \underline{a}^\pi \underline{A}^t \underline{a}^\pi. \quad (10.60)$$

6.- El subespacio dual-primal de $\tilde{D}^{DP}(\bar{\Omega}) = \underline{a}^\pi \tilde{D}(\bar{\Omega})$.

Observemos que cuando $\pi \subset \bar{\Omega}$ es vacío, $\underline{A} = \underline{A}^t$ y $\tilde{D}^{DP}(\bar{\Omega}) = \tilde{D}(\bar{\Omega})$. Además la matriz define una función $\underline{A} : \tilde{D}^{DP}(\bar{\Omega}) \rightarrow \tilde{D}^{DP}(\bar{\Omega})$ la cual es simétrica y positiva definida. Esto es a causa

$$\underline{w} \cdot \underline{A}u = \underline{w} \cdot \underline{a}^\pi \underline{A}^t \underline{a}^\pi u = \underline{w} \cdot \underline{A}^t u, \quad \forall u, \underline{w} \in \tilde{D}^{DP}(\bar{\Omega}) \quad (10.61)$$

además, notemos que $\underline{A} : \tilde{D}^{DP}(\bar{\Omega}) \rightarrow \tilde{D}^{DP}(\bar{\Omega})$ es positiva definida y biyectiva, mientras que $\underline{A} : \tilde{D}(\bar{\Omega}) \rightarrow \tilde{D}^{DP}(\bar{\Omega})$ no lo es. Además:

$$\begin{aligned} \bar{D}(\bar{\Omega}) &= \tilde{D}_{12}(\bar{\Omega}) = \underline{a} \tilde{D}^{DP}(\bar{\Omega}) \subset \tilde{D}^{DP}(\bar{\Omega}) \\ \tilde{D}(\Delta) &\subset \tilde{D}^{DP}(\bar{\Omega}) \\ \tilde{D}_{11}(\Delta) &\subset \underline{j} \tilde{D}(\bar{\Omega}) = \underline{j} \tilde{D}^{DP}(\bar{\Omega}). \end{aligned} \quad (10.62)$$

Definición 111 Sea $\underline{\hat{f}} \in \tilde{D}(\Omega)$. Entonces el problema original consiste en buscar una función $\underline{\hat{u}} \in \tilde{D}(\Omega)$ que satisfaga

$$\underline{\hat{A}}\underline{\hat{u}} = \underline{\hat{f}}. \quad (10.63)$$

El problema transformado consiste en buscar para una función $\underline{\hat{u}} \in \tilde{D}^{DP}(\bar{\Omega})$ que satisfaga

$$\underline{aA}\underline{\hat{u}} = \underline{\bar{f}} \quad \text{y} \quad \underline{j}\underline{\hat{u}} = 0 \quad (10.64)$$

donde $\underline{\bar{f}} \in \bar{D}(\bar{\Omega}) = \tilde{D}_{12}(\Omega) \subset \tilde{D}^{DP}(\bar{\Omega})$ es dado por

$$\underline{\bar{f}} \equiv \begin{pmatrix} \underline{\bar{f}}_{\Pi} \\ \underline{\bar{f}}_{\Delta} \end{pmatrix} = \underline{m}^{-1}\tau(\underline{\hat{f}}) \quad \text{y} \quad \underline{\bar{f}}_{\Delta} = \underline{\bar{f}}_{\Delta 2}. \quad (10.65)$$

Teorema 112 Una formulación equivalente del problema transformado es buscar para una función $\underline{\tilde{u}} \in \tilde{D}(\bar{\Omega})$ que satisfaga

$$\begin{cases} \underline{aA}^t \underline{\tilde{u}} = \underline{\bar{f}} \\ \underline{j}\underline{\tilde{u}} = 0 \end{cases} \quad (10.66)$$

además, una función $\underline{\tilde{u}} \in \tilde{D}(\bar{\Omega})$ es solución del problema transformado si y sólo si

$$\underline{\hat{u}} \equiv \tau^{-1}(\underline{\tilde{u}}) \quad (10.67)$$

es solución del problema original.

Demostración. Para hacer la prueba, usaremos el hecho de que cuando $\underline{j}\underline{\tilde{u}} = 0$, entonces $\underline{\tilde{u}} \in \tilde{D}_{12}(\bar{\Omega}) \subset \tilde{D}^{DP}(\bar{\Omega})$ y también

$$\underline{a}^{\pi} \underline{A}^t \underline{\tilde{u}} = \underline{a} \underline{a}^{\pi} \underline{A}^t \underline{a}^{\pi} \underline{\tilde{u}} = \underline{aA}\underline{\tilde{u}} \quad (10.68)$$

recordando la definición de la transformación $\tau^{-1} : \tilde{D}_{12}(\bar{\Omega}) \rightarrow \tilde{D}(\bar{\Omega})$ y asumiendo que $\underline{\tilde{u}} \in \tilde{D}(\bar{\Omega})$ está relacionada con $\underline{\hat{u}} \in \tilde{D}(\Omega)$ por la Ec.(10.67) entonces tenemos que:

\Rightarrow] Asumiendo que $\underline{\hat{u}} \in \tilde{D}(\Omega)$ es solución del problema original, entonces $\underline{\tilde{u}} \equiv \tau(\underline{\hat{u}})$ satisfaga la Ec.(10.66).

$[\Leftarrow$ La Ec.(10.66) implica que $\underline{\tilde{u}} \in \tilde{D}(\bar{\Omega})$, tal que τ^{-1} está bien definida. Tomando $\underline{\tilde{u}} \in \tilde{D}(\bar{\Omega})$ que satisfaga la Ec.(10.67), entonces $\underline{\hat{u}}$ satisfaga la Ec.(10.63). ■

10.7 Construcción de la Matriz $\underline{\underline{A}}$

En esta sección se desarrollará el procedimiento para construir una matriz $\underline{\underline{A}}^t : \tilde{D}(\overline{\Omega}) \rightarrow \tilde{D}(\overline{\Omega})$ que satisfaga la condición dada por la Ec.(10.58) y empezaremos por desarrollar esta condición de manera más explícita.

Recordando que d es la cardinalidad de Ω . Entonces, obsérvese que el conjunto de vectores $\{\underline{e}_1, \dots, \underline{e}_d\} \subset \tilde{D}(\Omega)$ es una base para $\tilde{D}(\Omega)$, donde para cada $i \in \Omega$, \underline{e}_i es definida por

$$\underline{e}_i \equiv (\delta_{i1}, \dots, \delta_{id}) \quad (10.69)$$

aquí y en lo que sigue el símbolo δ_{ij} es la delta de Kronecker.

La inmersión natural de este conjunto es $\{\tau(\underline{e}_1), \dots, \tau(\underline{e}_d)\} \subset \tilde{D}(\overline{\Omega})$, donde

$$\tau(\underline{e}_i)_{(j,\alpha)} = \delta_{ij}, \quad \forall (j, \alpha) \in \overline{\Omega}. \quad (10.70)$$

Cuando la Ec.(10.58) es aplicada a ese conjunto de vectores, una condición equivalente es obtenida. Esta es

$$\sum_{\alpha \in Z(j)} \sum_{\beta \in Z(i)} \underline{\underline{A}}_{(i,\alpha)(j,\beta)} = \hat{\underline{\underline{A}}}_{ij}, \quad \forall i, j \in \Omega \quad (10.71)$$

Aquí, para las matrices como $\underline{\underline{A}}^t : \tilde{D}(\overline{\Omega}) \rightarrow \tilde{D}(\overline{\Omega})$, usaremos la siguiente notación

$$\underline{\underline{A}}^t \equiv \left(\underline{\underline{A}}^t_{(i,\alpha)(j,\beta)} \right), \quad \text{con } (i, \alpha)(j, \beta) \in \overline{\Omega} \quad (10.72)$$

Definición 113 Para cada $\alpha \in \{1, \dots, E\}$ y para cada par $i, j \in \Omega$, definimos el símbolo δ_{ij}^α por

$$\begin{cases} \delta_{ij}^\alpha = 1, & \text{si } i, j \in \Omega_\alpha \\ \delta_{ij}^\alpha = 0, & \text{si } i \text{ ó } j \notin \Omega_\alpha \end{cases} \quad (10.73)$$

además la multiplicidad $m(i, j)$ del par (i, j) es definida por

$$m(i, j) \equiv \sum_{\alpha=1}^E \delta_{ij}^\alpha. \quad (10.74)$$

Usando la notación de la condición dad por la Ec.(10.57), tenemos

$$m(i, j) = 0 \Rightarrow \hat{\underline{\underline{A}}}_{ij} = 0. \quad (10.75)$$

Definición 114 La matriz total $\underline{\underline{A}}^t : \tilde{D}(\bar{\Omega}) \rightarrow \tilde{D}(\bar{\Omega})$ es ahora definida por

$$\left. \begin{aligned} \underline{\underline{A}}^t_{(i,\alpha)(j,\beta)} &\equiv 0, \text{ si } m(i,j) = 0 \\ \underline{\underline{A}}^t_{(i,\alpha)(j,\beta)} &\equiv \frac{1}{m(i,j)} \hat{A}_{ij} \delta_{ij}^\alpha \delta_{\alpha\beta}, \text{ si } m(i,j) \neq 0 \end{aligned} \right\} \forall (i,\alpha)(j,\beta) \in \bar{\Omega}. \quad (10.76)$$

Y es fácil verificar que $\underline{\underline{A}}^t$ como se definió, satisface la condición de la Ec.(10.71). Y usando la Ec.(10.74) tenemos que

$$\begin{aligned} \sum_{\alpha \in Z(j)} \sum_{\beta \in Z(i)} \underline{\underline{A}}^t_{(i,\alpha)(j,\beta)} &= \frac{1}{m(i,j)} \hat{A}_{ij} \sum_{\alpha=1}^E \sum_{\beta=1}^E \delta_{ij}^\alpha \delta_{\alpha\beta} = \\ &= \frac{1}{m(i,j)} \hat{A}_{ij} \sum_{\alpha=1}^E \delta_{ij}^\alpha = \hat{A}_{ij}. \end{aligned} \quad (10.77)$$

Para la actual construcción de $\underline{\underline{A}}^t$, cuando se implementa el algoritmo, es útil usar la siguiente notación, para cada $\gamma = 1, \dots, E$, definiendo $\underline{\underline{A}}^\gamma : \tilde{D}(\bar{\Omega}) \rightarrow \tilde{D}(\bar{\Omega})$ por

$$\left\{ \begin{aligned} (A^\gamma)_{(i,\alpha)(j,\beta)} &\equiv \frac{1}{m(i,j)} \hat{A}_{ij} \delta_{ij}^\gamma \delta_{\gamma\beta} \delta_{\gamma\alpha}, \text{ si } m(i,j) \neq 0 \\ (A^\gamma)_{(i,\alpha)(j,\beta)} &\equiv \hat{A}_{ij} = 0, \text{ si } m(i,j) = 0 \end{aligned} \right. \quad (10.78)$$

entonces

$$\underline{\underline{A}}^t = \sum_{\gamma=1}^E \underline{\underline{A}}^\gamma \quad (10.79)$$

nuevamente cuando $m(i,j) \neq 0$, tenemos que

$$\begin{aligned} \sum_{\gamma=1}^E (A^\gamma)_{(i,\alpha)(j,\beta)} &= \frac{1}{m(i,j)} \hat{A}_{ij} \sum_{\gamma=1}^E \delta_{ij}^\gamma \delta_{\alpha\beta} \delta_{\gamma\alpha} = \\ &= \frac{1}{m(i,j)} \hat{A}_{ij} \delta_{ij}^\alpha \delta_{\alpha\beta} = A^t_{(i,\alpha)(j,\beta)} \end{aligned} \quad (10.80)$$

y cuando $m(i,j) = 0$, tenemos

$$\sum_{\gamma=1}^E (A^\gamma)_{(i,\alpha)(j,\beta)} = A^t_{(i,\alpha)(j,\beta)} = 0. \quad (10.81)$$

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

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

El problema que se considera aquí⁵ es: “Encontrar $u \in H^2(\Omega)$, tal que

$$\begin{cases} \mathcal{L}u = f_\Omega, & \text{en } \Omega \\ u = 0, & \text{sobre } \partial\Omega \end{cases} \quad (11.1)$$

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

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

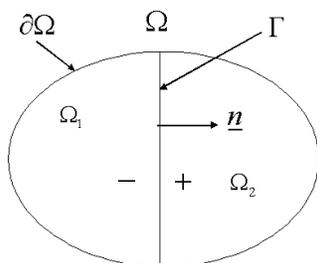


Figura 10: Partición del dominio Ω en dos subdominios Ω_1 y Ω_2 .

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

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

y de sus derivadas normales. La notación para el ‘salto’ y el ‘promedio’ a través de Γ está dado por

$$[[u]] \equiv u_+ - u_- \quad \text{y} \quad \hat{u} = \frac{1}{2}(u_+ + u_-), \text{ sobre } \Gamma \quad (11.2)$$

respectivamente.

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

$$[[u]] = \left[\left[\frac{\partial u}{\partial n} \right] \right] = 0, \text{ sobre } \Gamma. \quad (11.3)$$

Por lo tanto, una formulación del problema dado en la Ec.(11.1) es: “Buscar a $u \in H^2(\Omega_1) \oplus H^2(\Omega_2)$, tal que

$$\begin{cases} \mathcal{L}u = f_\Omega, \text{ en } \Omega \\ [[u]] = 0 \text{ y } \left[\left[\frac{\partial u}{\partial n} \right] \right] = 0, \text{ sobre } \Gamma \\ u = 0, \text{ sobre } \partial\Omega \end{cases} \quad (11.4)$$

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

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

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

11.1 El Problema no Precondicionado Dirichlet-Dirichlet

Para el problema no precondicionado Dirichlet-Dirichlet, se toma u_Γ como la restricción a Γ de la solución u de la Ec.(11.4), entonces u es la única solución de los siguientes dos problemas Dirichlet

$$\begin{cases} \mathcal{L}u = f_\Omega, \text{ en } \Omega_\alpha, \alpha = 1, 2 \\ u = u_\Gamma, \text{ sobre } \Gamma \\ u = 0, \text{ sobre } \partial\Omega \end{cases} \quad (11.5)$$

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

A este respecto, una primera pregunta es: ¿cómo se reconoce cuando la función de prueba es satisfactoria?. Se sabe que cuando u_Γ es la restricción a Γ de la solución del problema, la solución que es obtenida por la resolución de los dos problemas con valores en la frontera de la Ec.(11.5) satisfacen las condiciones de salto indicadas en la Ec.(11.4). Ahora, en el caso del enfoque Dirichlet-Dirichlet, el salto de la función se nulifica necesariamente, ya que

$$[[u]] = u_+ - u_- = u_\Gamma - u_\Gamma = 0 \quad (11.6)$$

sin embargo, por lo general la condición

$$\left[\left[\frac{\partial u}{\partial n} \right] \right] = 0 \quad (11.7)$$

no es satisfecha. La selección de la función de prueba u_Γ será satisfactoria, si y sólo si, la Ec.(11.7) se satisface.

Si se escribe la solución del problema u de la Ec.(11.4) como

$$u = u_p + v \quad (11.8)$$

donde u_p satisface

$$\begin{cases} \mathcal{L}u_p = f_\Omega, \text{ en } \Omega_\alpha, \alpha = 1, 2 \\ u_p = 0, \text{ sobre } \Gamma \\ u_p = 0, \text{ sobre } \partial\Omega \end{cases} \quad (11.9)$$

y por lo tanto v satisface

$$\begin{cases} \mathcal{L}v = 0, \text{ en } \Omega_\alpha, \alpha = 1, 2 \\ v = u_\Gamma, \text{ sobre } \Gamma \\ v = 0, \text{ sobre } \partial\Omega \end{cases} \quad (11.10)$$

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

$$\left[\left[\frac{\partial v}{\partial n} \right] \right] = - \left[\left[\frac{\partial u_p}{\partial n} \right] \right] \quad (11.11)$$

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

$$\begin{cases} \mathcal{L}v = 0, \text{ en } \Omega_\alpha, \alpha = 1, 2 \\ \llbracket v \rrbracket = 0, \llbracket \frac{\partial v}{\partial n} \rrbracket = - \llbracket \frac{\partial u_p}{\partial n} \rrbracket, \text{ sobre } \Gamma \\ v = 0, \text{ sobre } \partial\Omega \end{cases} \quad (11.12)$$

Esta condición puede expresarse teniendo en mente el operador de Steklov-Poincaré⁶ τ , el cual para cualquier función u definida sobre Γ , se genera otra función definida sobre Γ , a saber (véase [3])

$$\tau(u_\Gamma) \equiv \llbracket \frac{\partial v}{\partial n} \rrbracket, \text{ sobre } \Gamma \quad (11.13)$$

donde v satisface la Ec.(11.10). Entonces la Ec.(11.12) es equivalente a

$$\tau(u_\Gamma) \equiv - \llbracket \frac{\partial u_p}{\partial n} \rrbracket, \text{ sobre } \Gamma. \quad (11.14)$$

11.2 El Problema no Precondicionado Neumann-Neumann

Para el caso del problema no precondicionado Neumann-Neumann, se toma a u como la solución de la Ec.(11.4) y sea $q_\Gamma \equiv \frac{\partial u}{\partial n}$ sobre Γ , entonces u es la única solución de los siguientes dos problemas Neumann

$$\begin{cases} \mathcal{L}u = f_\Omega, \text{ en } \Omega_\alpha, \alpha = 1, 2 \\ \frac{\partial u}{\partial n} = q_\Gamma \text{ sobre } \Gamma \\ u = 0 \text{ sobre } \partial\Omega \end{cases} \quad (11.15)$$

El título del enfoque Neumann-Neumann proviene del hecho que la Ec.(11.15) implica resolver un problema Neumann en el subdominio Ω_1 y otro problema Neumann en Ω_2 . Este enfoque consiste en buscar una función q_Γ -independientemente del valor de q_Γ -, ya que, cualquier solución de la Ec.(11.15) satisface

$$\llbracket \frac{\partial u}{\partial n} \rrbracket = \left(\frac{\partial u}{\partial n} \right)_+ - \left(\frac{\partial u}{\partial n} \right)_- = q_\Gamma - q_\Gamma = 0 \quad (11.16)$$

⁶El operador de Steklov-Poincaré generalmente se define como la ecuación para la traza de la solución exacta u sobre Γ (véase [51]).

sin embargo, por lo general, la condición

$$[[u]] = 0 \quad (11.17)$$

no es satisfecha. La selección de la función de prueba u_Γ será satisfactoria, si y sólo si, la Ec.(11.17) se satisface.

Si se toma nuevamente una solución $u = u_p + v$ como en la Ec.(11.8), donde u_p ahora satisface

$$\begin{cases} \mathcal{L}u_p = f_\Omega, \text{ en } \Omega_\alpha, \alpha = 1, 2 \\ \frac{\partial u_p}{\partial n} = 0, \text{ sobre } \Gamma \\ u_p = 0, \text{ sobre } \partial\Omega \end{cases} \quad (11.18)$$

y por lo tanto v satisface

$$\begin{cases} \mathcal{L}v = 0, \text{ en } \Omega_\alpha, \alpha = 1, 2 \\ \frac{\partial v}{\partial n} = q_\Gamma, \text{ sobre } \Gamma \\ v = 0, \text{ sobre } \partial\Omega \end{cases} \quad (11.19)$$

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

$$\begin{cases} \mathcal{L}v = 0, \text{ en } \Omega_\alpha, \alpha = 1, 2 \\ [[v]] = [[u_p]], \left[\frac{\partial v}{\partial n} \right] = 0, \text{ sobre } \Gamma \\ v = 0, \text{ sobre } \partial\Omega \end{cases} \quad (11.20)$$

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

$$\mu(q_\Gamma) \equiv [[v]], \text{ sobre } \Gamma \quad (11.21)$$

aquí, v satisface la Ec.(11.19). Entonces, la Ec.(11.20) es equivalente a

$$\mu(q_\Gamma) \equiv -[[u_p]], \text{ sobre } \Gamma. \quad (11.22)$$

12 Marco Teórico del Espacio de Vectores Derivados

En el marco de los métodos de descomposición de dominio sin traslape se distinguen dos categorías (véase [18], [42], [43], [57], [67] y [72]): Los esquemas duales -como es el caso del método Finite Element Tearing and Interconnect (FETI) y sus variantes- los cuales usan multiplicadores de Lagrange; y los esquemas primales -como es el caso del método Balancing Domain Decomposition (BDD) y sus variantes- que tratan el problema sin el recurso de los multiplicadores de Lagrange.

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

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

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

12.1 Discretización del Dominio para los Métodos Duales y Primitives

Dos de los enfoques más comúnmente usados (véase [18], [40], [41], [42], [43], [57], [67], [68], [70] y [72]) en los métodos de descomposición de dominio son FETI-DP y BDDC. Ambos, inicializan con la ecuación diferencial parcial y de ella, los grados de libertad son asociados con las funciones de base usadas. BDDC es un método directo, i.e. es un método que no hace uso de Multiplicadores de Lagrange, mientras que FETI-DP trabaja en el espacio producto mediante el uso de los Multiplicadores de Lagrange.

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

$$\Omega_\alpha \cap \Omega_\beta = \emptyset \quad \forall \alpha \neq \beta \quad \text{y} \quad \bar{\Omega} = \bigcup_{\alpha=1}^E \bar{\Omega}_\alpha \quad (12.1)$$

y al conjunto

$$\Gamma = \bigcup_{\alpha=1}^E \Gamma_\alpha, \quad \text{si} \quad \Gamma_\alpha = \partial\Omega_\alpha \setminus \partial\Omega \quad (12.2)$$

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

$$\partial\Omega \subset \bigcup_{\alpha=1}^E \partial\Omega_\alpha \quad \text{y} \quad \Omega = \left(\bigcup_{\alpha=1}^E \Omega_\alpha \right) \cup \Gamma. \quad (12.3)$$

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

Para iniciar la discusión, se considera un ejemplo particular, de un conjunto de veinticinco nodos de una descomposición del dominio Ω ‘sin traslape’, se asume que la numeración de los nodos es de izquierda a derecha y de arriba hacia abajo, véase figura (11)

En este caso el conjunto de nodos originales, se particiona usando una malla gruesa de cuatro subdominios Ω_α $\alpha = 1, 2, 3, 4$, véase figura (12).

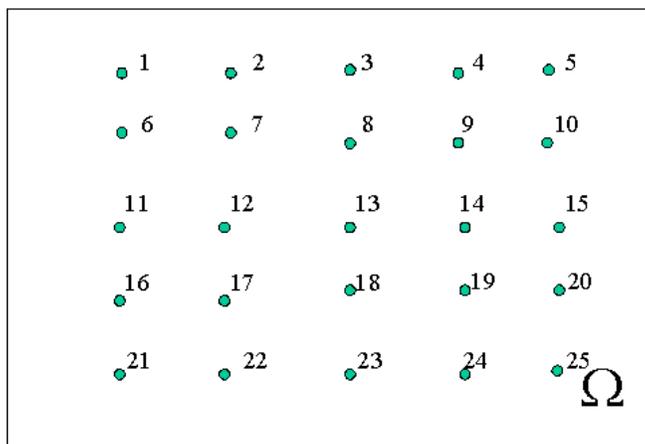


Figura 11: Conjunto de nodos originales

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

$$\hat{N} \equiv \{1, 2, \dots, 25\} \quad \text{y} \quad E \equiv \{1, 2, 3, 4\} \quad (12.4)$$

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

$$\begin{aligned} \hat{N}^1 &= \{1, 2, 3, 6, 7, 8, 11, 12, 13\} \\ \hat{N}^2 &= \{3, 4, 5, 8, 9, 10, 13, 14, 15\} \\ \hat{N}^3 &= \{11, 12, 13, 16, 17, 18, 21, 22, 23\} \\ \hat{N}^4 &= \{13, 14, 15, 18, 19, 20, 23, 24, 25\} \end{aligned} \quad (12.5)$$

no son disjuntos. En efecto, por ejemplo

$$\hat{N}^1 \cap \hat{N}^2 = \{3, 8, 13\}. \quad (12.6)$$

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

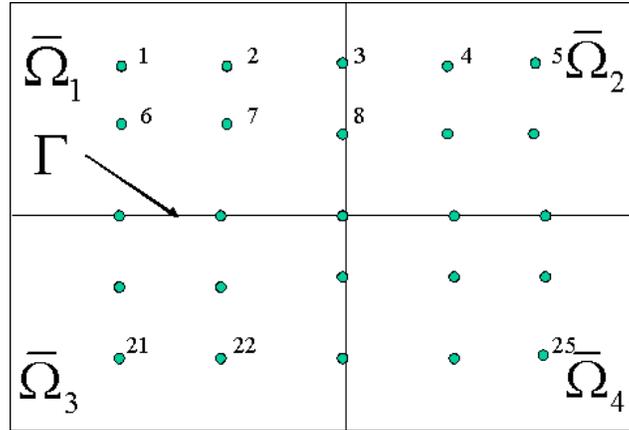


Figura 12: Nodos en una descomposición gruesa de cuatro subdominios

12.2 Una Verdadera Descomposición de Dominio sin Traslapes

A la luz de lo visto en la sección anterior, es ventajoso trabajar con una descomposición de dominio sin traslape alguno del conjunto de nodos y, para este fin, se reemplaza el conjunto \hat{N} de ‘nodos originales’ por otro conjunto de ‘nodos derivados’, los cuales son denotados por X^α , donde cada nodo en la frontera interior Γ se parte en un número que llamaremos la multiplicidad del nodo y es definida como el número de subdominios que comparten al nodo. Cada nodo derivado es definido por un par de números: (p, α) , donde p corresponde a un nodo perteneciente a $\bar{\Omega}_p$, i.e. un ‘nodo derivado’ es el par de números (p, α) tal que $p \in \hat{N}^\alpha$, véase figura (13).

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

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

$$X^\alpha = \left\{ (p, \alpha) \mid \alpha \in \hat{E} \text{ y } p \in \hat{N}^\alpha \right\}. \quad (12.7)$$

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

$$\{X^1, X^2, X^3, X^4\} \quad (12.8)$$

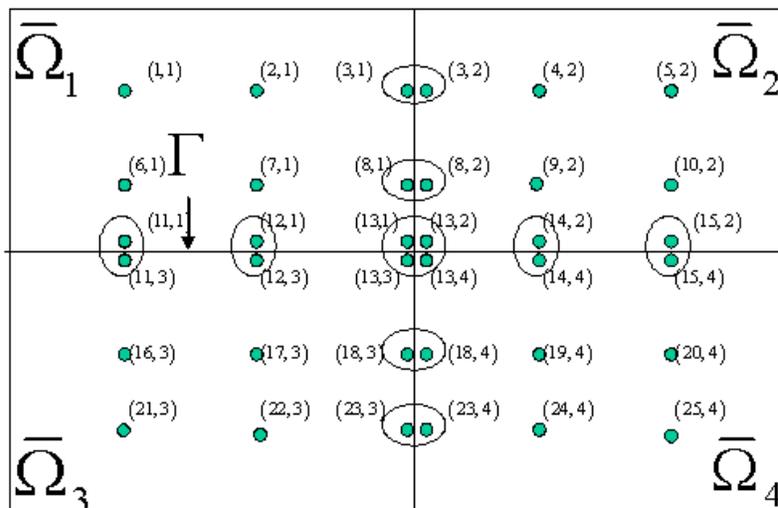


Figura 13: Mitosis de los nodos en la frontera interior

la cual es una descomposición de dominio sin traslape véase figura (14), donde el conjunto X satisface

$$X = \bigcup_{\alpha=1}^4 X^\alpha \quad \text{y} \quad X^\alpha \cap X^\beta = \emptyset \quad \text{cuando} \quad \alpha \neq \beta. \quad (12.9)$$

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

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

$$\hat{N} = \{1, \dots, n\} \quad \text{y} \quad \hat{E} = \{1, \dots, E\} \quad (12.10)$$

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

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

$$X^\alpha \equiv \left\{ (p, \alpha) \mid \alpha \in \hat{E} \quad \text{y} \quad p \in \hat{N}^\alpha \right\} \quad (12.11)$$

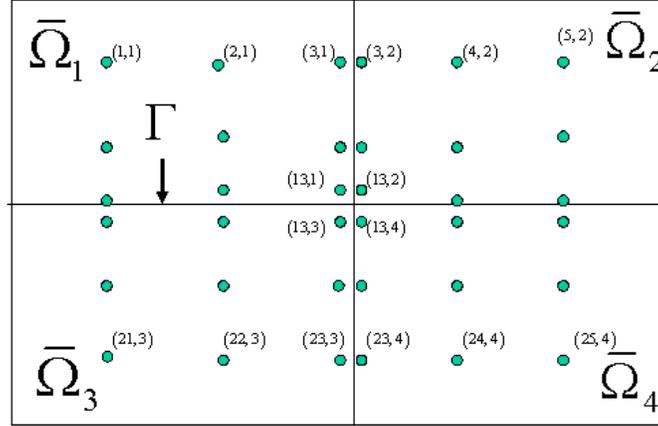


Figura 14: Conjunto de nodos derivados

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

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

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

Los nodos derivados son clasificados como ‘nodos internos’ o de ‘frontera interna’, dependiendo si su multiplicidad es uno o más grande que uno. Los subconjuntos $I \subset \bar{\Omega}$ y $\Gamma \subset \bar{\Omega}$ son el conjunto de nodos internos y de frontera interna respectivamente, que satisfacen la siguiente relación

$$X = I \cup \Gamma \quad \text{y} \quad I \cap \Gamma = \emptyset \quad (12.12)$$

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

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

$$\Gamma_\alpha = \Gamma \cap X^\alpha \quad (12.13)$$

entonces, la familia de subconjuntos $\{\Gamma_1, \Gamma_2, \dots, \Gamma_E\}$ es una partición de Γ ,

en el sentido de que

$$\Gamma \equiv \bigcup_{\alpha=1}^E \Gamma_{\alpha} \quad \text{mientras} \quad \Gamma_{\alpha} \cap \Gamma_{\beta} = \emptyset \quad \text{cuando} \quad \alpha \neq \beta. \quad (12.14)$$

En este desarrollo, Γ es descompuesto dentro de dos subconjuntos

$$\pi \subset \Gamma \subset X \quad \text{y} \quad \Delta \subset \Gamma \subset X \quad (12.15)$$

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

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

entonces

$$\Gamma = \pi \cup \Delta \quad \text{mientras} \quad \pi \cap \Delta = \emptyset. \quad (12.16)$$

Además, se define

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

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

$$X = I \cup \pi \cup \Delta \quad \text{mientras que} \quad I \cap \pi = \pi \cap \Delta = \Delta \cap I = \emptyset \quad (12.17)$$

y

$$X = \Pi \cup \Delta \quad \text{mientras que} \quad \Pi \cap \Delta = \emptyset. \quad (12.18)$$

Nótese que todos los conjuntos considerados hasta ahora son dimensionalmente finitos. Por lo tanto, cualquier función con valores en los reales⁷ definida en cualquier conjunto define unívocamente un vector dimensionalmente finito.

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

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

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

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

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

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

Además, estos conjuntos de nodos satisfacen las siguientes relaciones:

$$\hat{N} = \hat{N}_I \cup \hat{N}_\Gamma = \hat{N}_I \cup \hat{N}_\pi \cup \hat{N}_\Delta \quad \text{y} \quad \hat{N}_\Gamma = \hat{N}_\pi \cup \hat{N}_\Delta \quad (12.19)$$

adicionalmente se define los siguientes conjuntos de X :

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

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

$$X = \bigcup_{\alpha=1}^E X^\alpha \quad \text{y} \quad X^\alpha \cap X^\beta = \emptyset \quad \text{cuando} \quad \alpha \neq \beta. \quad (12.20)$$

12.3 El Problema Original

El esquema DVS puede ser aplicado al sistema matricial que es obtenido después de la discretización, este procedimiento es independiente del método de discretización usado -puede ser el método de Elemento Finito, Diferencias Finitas o cualquier otro-. El procedimiento requiere de algunas suposiciones que deben de ser satisfechas, las cuales se explican a continuación (véase [36] y [38]). Tales suposiciones están dadas en términos del sistema matricial y

dos conceptos adicionales: los nodos originales y una familia de subconjuntos de tales nodos, los cuales están asociados con la partición del dominio.

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

$$a(\hat{u}, v) = (g, v), \quad \forall v \in V \quad (12.21)$$

aquí, V es un espacio lineal de dimensión finita de funciones real valuadas⁸ definidas en cierto dominio espacial Ω , mientras $g \in V$ es una función dada.

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

$$\hat{u} = \sum \hat{u}_i \varphi_i \quad (12.22)$$

aquí, \hat{u}_i es el valor de \hat{u} en el nodo i . Sea $\underline{\hat{u}}$ y $\underline{\hat{f}}$ vectores⁹ $\underline{\hat{u}} \equiv (\hat{u}_1, \dots, \hat{u}_n)$ y $\underline{\hat{f}} \equiv (\hat{f}_1, \dots, \hat{f}_n)$, donde

$$\hat{f}_i \equiv (g, \varphi_i), \quad i \in \hat{N}. \quad (12.23)$$

La formulación variacional de la Ec.(12.21) es equivalente a

$$\underline{\underline{\hat{A}}}\underline{\hat{u}} = \underline{\hat{f}} \quad (12.24)$$

donde la matriz $\underline{\underline{\hat{A}}}$, será referida como la ‘matriz original’ y esta es definida mediante

$$\underline{\underline{\hat{A}}} \equiv (\hat{A}_{ij}) \quad (12.25)$$

con

$$\hat{A}_{ij} \equiv \tilde{a}(\varphi_i, \varphi_j) \quad i, j = 1, \dots, n \quad (12.26)$$

después de que el problema ha sido discretizado $\tilde{a}(\varphi_i, \varphi_j)$ son las funciones base usadas en la discretización-, donde se supone que el dominio

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

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

Ω ha sido particionado mediante un conjunto de subdominios no traslapados $\{\Omega_1, \dots, \Omega_E\}$; más precisamente, para cada $\alpha = 1, \dots, E$; Ω_α es abierto y

$$\Omega_\alpha \cap \Omega_\beta = \emptyset \quad \forall \alpha \neq \beta \quad \text{y} \quad \Omega = \bigcup_{\alpha=1}^E \overline{\Omega}_\alpha \quad (12.27)$$

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

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

Las funciones real valuadas definidas en $\hat{N} = \{1, \dots, n\}$ constituyen un espacio lineal el cual será denotado por \widehat{W} y referido como el ‘espacio vectorial original’. Los vectores $\widehat{u} \in \widehat{W}$ se escriben como $\widehat{u} = (\widehat{u}_1, \dots, \widehat{u}_n)$, donde \widehat{u}_i para $i = 1, \dots, n$, son las componentes de un vector \widehat{u} .

Entonces, el ‘problema original’ consiste en “Dada $\widehat{f} \in \widehat{W}$, encontrar a $\widehat{u} \in \widehat{W}$ tal que la Ec.(12.24) se satisfaga”.

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

$$\widehat{A}_{ij} = 0 \quad \text{siempre y cuando} \quad \alpha \neq \beta. \quad (12.28)$$

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

$$W \equiv \prod_{\alpha=1}^E \widehat{W}^\alpha = \widehat{W}^1 \times \dots \times \widehat{W}^E. \quad (12.29)$$

12.4 El Espacio de Vectores Derivado

Usando la notación de la sección anterior, en la cual se definió el ‘problema original’ dado por la Ec.(12.24), este es un problema formulado en el espacio vectorial original \widehat{W} , en el desarrollo que sigue se transforma en un problema que será reformulado en el espacio W , el cual es un espacio definido sobre las funciones discontinuas.

Para ello, se entiende por un ‘vector derivado’ una función real-valuada definida en el conjunto X de nodos derivados¹⁰. El conjunto de vectores derivados constituyen un espacio lineal, el cual será referido como el ‘espacio de vectores derivados’. De manera análoga para cada subconjunto local de nodos derivados en el subespacio X^α existe un ‘subespacio local de vectores derivados’ W^α , el cual es definido con la condición de que los vectores de W^α se nulifiquen en cada nodo derivado que no pertenezca a X^α . Una manera formal de iniciar esta definición es

$$\underline{u} \in W^\alpha \subset W, \quad \text{si y sólo si} \quad \underline{u}(p, \beta) = 0 \quad \text{cuando} \quad \beta \neq \alpha. \quad (12.30)$$

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

$$W \equiv \prod_{\alpha=1}^E \widehat{W}^\alpha = W^1 \oplus \dots \oplus W^E \quad (12.31)$$

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

$$\{\widehat{W}^1, \dots, \widehat{W}^E\} \quad (12.32)$$

pero al mismo tiempo es la suma directa de la familia

$$\{W^1, \dots, W^E\}. \quad (12.33)$$

En vista de la Ec.(12.31) es sencillo establecer una biyección -de hecho, un isomorfismo- entre el espacio de vectores derivados y el espacio producto. Por lo tanto, en lo que sigue, se identifica a ambos como uno solo.

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

Para cada par de vectores¹¹ $\underline{u} \in W$ y $\underline{w} \in W$, el ‘producto interior Euclidiano’ es definido por

$$\underline{u} \cdot \underline{w} = \sum_{(p,\alpha) \in X} \underline{u}(p, \alpha) \underline{w}(p, \alpha). \quad (12.34)$$

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

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

$$(R\hat{u})(p, \alpha) = \hat{u}(p), \quad \forall (p, \alpha) \in X. \quad (12.35)$$

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

$$\sum_{\alpha=1}^E (R\hat{u})(p, \alpha) = m(p)\hat{u}(p). \quad (12.36)$$

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

$$W = W_{11} + W_{12} \quad \text{y} \quad \{0\} = W_{11} \cap W_{12} \quad (12.37)$$

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

$$W_{12} \equiv R\widehat{W} \subset W \quad (12.38)$$

y $W_{11} \subset W$ es el complemento ortogonal con respecto al producto interior Euclidiano. Además se define la inversa de $R : \widehat{W} \rightarrow W$, cuando ésta es

¹¹En el caso vectorial -que surge en aplicaciones como en la teoría de elasticidad- cuando se trabajan sistemas de ecuaciones, i.e. cuando $\underline{u}(p, \alpha)$ es un vector, la Ec(12.34) es reemplazada por

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

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

restringida a $W_{12} \subset W$ la cual siempre existe y es denotada por $R^{-1} : W_{12} \rightarrow \widehat{W}$.

Es costumbre el uso de la notación de suma directa

$$W = W_{12} \oplus W_{12} \quad (12.39)$$

cuando el par de condiciones de la Ec.(12.37) son satisfechas.

El ‘subespacio de vectores continuos’ es definido como el subespacio

$$W_{12} \subset W \quad (12.40)$$

mientras que el ‘subespacio de vectores de promedio cero’ es definido como el subespacio

$$W_{11} \subset W. \quad (12.41)$$

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

En vista de la Ec.(12.37), cada vector $\underline{u} \in W$, puede ser escrito de forma única como la suma de un vector de promedio cero más un vector continuo (vector de salto cero), es decir

$$\underline{u} = \underline{u}_{11} + \underline{u}_{12} \text{ con } \begin{cases} \underline{u}_{11} \equiv \underline{j}\underline{u} \in W_{11} \\ \underline{u}_{12} \equiv \underline{a}\underline{u} \in W_{12} \end{cases} \quad (12.42)$$

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

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

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

Además

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

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

$$\{W_I, W_\Gamma\}, \{W_I, W_\pi, W_\Delta\}, \{W_\Pi, W_\Delta\} \quad (12.43)$$

son linealmente independientes. Y también satisfacen

$$W = W_I + W_\Gamma = W_I + W_\pi + W_\Delta \text{ y } W_r = W_\Pi + W_\Delta \quad (12.44)$$

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

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

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

Sea la matriz $\widehat{\underline{\underline{A}}} : \widehat{W} \rightarrow \widehat{W}$ dada por la Ec.(12.24) la cual puede ser escrita como

$$\widehat{\underline{\underline{A}}} \equiv \left(\widehat{A}_{pq} \right) \quad (12.45)$$

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

$$\delta_{pq}^\alpha \equiv \begin{cases} 1, & \text{si } p, q \in \hat{N}^\alpha \\ 0, & \text{si } p \notin \hat{N}^\alpha \text{ ó } q \notin \hat{N}^\alpha \end{cases}, \alpha = 1, \dots, E \quad (12.46)$$

junto con

$$m(p, q) \equiv \sum_{\alpha=1}^N \delta_{pq}^\alpha \quad (12.47)$$

y

$$s(p, q) \equiv \begin{cases} 1, & \text{cuando } m(p, q) = 0 \\ m(p, q), & \text{cuando } m(p, q) \neq 0 \end{cases} \quad (12.48)$$

la función $m(p, q)$ es llamada la ‘multiplicidad’ del par (p, q) . Esto puede verse de la suposición básica dada por la Ec.(12.28), teniendo que

$$m(p, q) = 0 \Rightarrow \widehat{A}_{pq} = 0. \quad (12.49)$$

Definiendo ahora

$$\underline{\underline{\widehat{A}}}^\alpha \equiv \left(\widehat{A}_{pq}^\alpha \right) \text{ con } \widehat{A}_{pq}^\alpha = \frac{\widehat{A}_{pq} \delta_{pq}^\alpha}{s(p, q)} \quad (12.50)$$

se observa la identidad

$$\underline{\underline{\widehat{A}}} = \sum_{\gamma=1}^E \underline{\underline{\widehat{A}}}^\gamma \quad (12.51)$$

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

$$\underline{\underline{A}}^\gamma \equiv \left(A_{(p,\alpha)(q,\beta)}^\gamma \right) \quad (12.52)$$

con

$$A_{(p,\alpha)(q,\beta)}^\gamma \equiv \delta_{\alpha\gamma}^\gamma \widehat{A}_{pq}^\gamma \delta_{\beta\gamma} \quad (12.53)$$

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

$$\underline{\underline{A}}^t \equiv \sum_{\gamma=1}^E \underline{\underline{A}}^\gamma \quad (12.54)$$

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

$$(\underline{\underline{A}}^t)^{-1} = \sum_{\gamma=1}^E (\underline{\underline{A}}^\gamma)^{-1} \quad (12.55)$$

ya que las matrices $\underline{\underline{A}}^\gamma$, con $\gamma = 1, 2, \dots, E$ son independientes entre si -al ser una verdadera descomposición del dominio, ver sección (12.2)-. Además, otra propiedad es que

$$R^{-1} \underline{\underline{a}} \underline{\underline{A}} R = R^{-1} \underline{\underline{a}} \underline{\underline{A}} \underline{\underline{a}} R = \widehat{A}. \quad (12.56)$$

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

$$\begin{aligned} \underline{\underline{A}}^t &= \begin{pmatrix} \underline{\underline{A}}_{\text{III}}^t & \underline{\underline{A}}_{\text{II}\Delta}^t \\ \underline{\underline{A}}_{\Delta\text{II}}^t & \underline{\underline{A}}_{\Delta\Delta}^t \end{pmatrix} = \begin{pmatrix} \underline{\underline{A}}_{II}^t & \underline{\underline{A}}_{I\pi}^t & \underline{\underline{A}}_{I\Delta}^t \\ \underline{\underline{A}}_{\pi I}^t & \underline{\underline{A}}_{\pi\pi}^t & \underline{\underline{A}}_{\pi\Delta}^t \\ \underline{\underline{A}}_{\Delta I}^t & \underline{\underline{A}}_{\Delta\pi}^t & \underline{\underline{A}}_{\Delta\Delta}^t \end{pmatrix} \\ &= \begin{pmatrix} \underline{\underline{A}}^1 & \underline{\underline{0}} & \cdots & \underline{\underline{0}} \\ \underline{\underline{0}} & \underline{\underline{A}}^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \underline{\underline{0}} \\ \underline{\underline{0}} & \cdots & \underline{\underline{0}} & \underline{\underline{A}}^N \end{pmatrix}. \end{aligned} \quad (12.57)$$

A la luz de la Ec.(12.55), implica que, si el complemento de Schur local

$$\underline{\underline{S}}^\alpha = \underline{\underline{A}}_{\Gamma\Gamma}^\alpha - \underline{\underline{A}}_{\Gamma I}^\alpha \left(\underline{\underline{A}}_{II}^\alpha \right)^{-1} \underline{\underline{A}}_{I\Gamma}^\alpha \quad (12.58)$$

de cada $\underline{\underline{A}}^\alpha$ existe y es invertible -véase sección (19.7) para la definición y su implementación del complemento de Schur en general-, entonces, la matriz ‘total del complemento de Schur’, $\underline{\underline{S}}^t : W_r \rightarrow W_r$, satisface

$$\underline{\underline{S}}^t = \sum_{\alpha=1}^E \underline{\underline{S}}^\alpha \quad (12.59)$$

y

$$(\underline{\underline{S}}^t)^{-1} = \sum_{\alpha=1}^E (\underline{\underline{S}}^\alpha)^{-1} \quad (12.60)$$

al ser una verdadera descomposición del dominio -ver sección (12.2)- y porque se satisfacen las Ecs.(12.54 y 12.55).

Por otro lado, si se define $\underline{u}' \equiv R\widehat{u}$, usando la Ec.(12.56) entonces el problema original Ec.(12.24), se transforma en uno equivalente a

$$\underline{a}A^t\underline{u}' = \underline{f} \quad \text{con} \quad \underline{j}\underline{u}' = 0 \quad (12.61)$$

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

$$\widehat{u} = R^{-1}\underline{u}' \quad (12.62)$$

esta última es correcta cuando \underline{u}' es evaluada exactamente, pero en las aplicaciones numéricas, \underline{u}' sólo es una evaluación aproximada, por ello puede generar “inestabilidades”, en el sentido de que una aproximación a \underline{u}' , independientemente de cuán pequeño sea el error, puede no ser continua en cuyo caso $\widehat{u} = R^{-1}\underline{u}'$ no está definida. Por lo tanto es conveniente reemplazar la Ec.(12.62) por la siguiente versión estable

$$\widehat{u} = R^{-1} \left(\underline{a}\underline{u}' \right). \quad (12.63)$$

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

$$\underline{A} \equiv \underline{a}^r \underline{A}^t \underline{a}^r. \quad (12.64)$$

Por otro lado, se tiene que

$$\begin{aligned} \underline{A} &\equiv \underline{a}^r \underline{A}^t \underline{a}^r \equiv \begin{pmatrix} \underline{A}_{\text{III}} & \underline{A}_{\text{II}\Delta} \\ \underline{A}_{\Delta\text{II}} & \underline{A}_{\Delta\Delta} \end{pmatrix} \\ &= \begin{pmatrix} \underline{A}_{\text{II}}^t & \underline{A}_{\text{I}\overline{\text{II}}}^t \underline{a}^\pi & \underline{A}_{\text{I}\Delta}^t \\ \underline{a}^\pi \underline{A}_{\text{II}}^t & \underline{a}^\pi \underline{A}_{\text{II}\overline{\text{II}}}^t \underline{a}^\pi & \underline{a}^\pi \underline{A}_{\text{I}\Delta}^t \\ \underline{A}_{\Delta\text{I}}^t & \underline{A}_{\Delta\overline{\text{II}}}^t \underline{a}^\pi & \underline{A}_{\Delta\Delta}^t \end{pmatrix} \end{aligned} \quad (12.65)$$

la notación usada es tal que

$$\begin{cases} \underline{A}_{\text{III}} : W_{\text{II}} \rightarrow W_{\text{II}}, & \underline{A}_{\text{II}\Delta} : W_{\Delta} \rightarrow W_{\text{II}} \\ \underline{A}_{\Delta\text{II}} : W_{\text{II}} \rightarrow W_{\Delta}, & \underline{A}_{\Delta\Delta} : W_{\Delta} \rightarrow W_{\Delta} \end{cases} \quad (12.66)$$

donde

$$\begin{cases} \underline{\underline{A}}_{\Pi\Pi}u = (\underline{\underline{A}}u_{\Pi})_{\Pi}, & \underline{\underline{A}}_{\Delta\Pi}u = (\underline{\underline{A}}u_{\Pi})_{\Delta} \\ \underline{\underline{A}}_{\Pi\Delta}u = (\underline{\underline{A}}u_{\Delta})_{\Pi}, & \underline{\underline{A}}_{\Delta\Delta}u = (\underline{\underline{A}}u_{\Delta})_{\Delta} \end{cases} \quad (12.67)$$

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

$$\begin{aligned} \underline{\underline{A}}_{\Pi\Pi} &\equiv \begin{pmatrix} \underline{\underline{A}}_{\Pi\Pi} & 0 \\ 0 & 0 \end{pmatrix} \\ \underline{\underline{A}}_{\Pi\Delta} &\equiv \begin{pmatrix} 0 & \underline{\underline{A}}_{\Pi\Delta} \\ 0 & 0 \end{pmatrix} \\ \underline{\underline{A}}_{\Delta\Pi} &\equiv \begin{pmatrix} 0 & 0 \\ \underline{\underline{A}}_{\Delta\Pi} & 0 \end{pmatrix} \\ \underline{\underline{A}}_{\Delta\Delta} &\equiv \begin{pmatrix} 0 & 0 \\ 0 & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix}. \end{aligned} \quad (12.68)$$

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

$$\underline{\underline{S}} = \underline{\underline{A}}_{\Delta\Delta} - \underline{\underline{A}}_{\Delta\Pi} \underline{\underline{A}}_{\Pi\Pi}^{-1} \underline{\underline{A}}_{\Pi\Delta} \quad (12.69)$$

que define al sistema virtual

$$\underline{\underline{S}}u_{\Gamma} = \underline{\underline{f}}_{\Gamma}. \quad (12.70)$$

Nótese que

$$\underline{\underline{A}}_{\Pi\Pi} = \begin{pmatrix} \underline{\underline{A}}_{II} & \underline{\underline{A}}_{I\pi} \\ \underline{\underline{A}}_{\pi I} & \underline{\underline{A}}_{\pi\pi} \end{pmatrix} \quad (12.71)$$

así, se definen a las matrices

$$\underline{\underline{A}}_{II}^{\alpha}, \underline{\underline{A}}_{I\pi}^{\alpha}, \underline{\underline{A}}_{I\Delta}^{\alpha}, \underline{\underline{A}}_{\pi I}^{\alpha}, \underline{\underline{A}}_{\pi\pi}^{\alpha}, \underline{\underline{A}}_{\pi\Delta}^{\alpha}, \underline{\underline{A}}_{\Delta I}^{\alpha}, \underline{\underline{A}}_{\Delta\pi}^{\alpha} \text{ y } \underline{\underline{A}}_{\Delta\Delta}^{\alpha} \quad (12.72)$$

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

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

$$\underline{\underline{A}}_{\pi I}^{\alpha} = \left(\widehat{A}_{pq}^{\alpha} \right) \quad (12.73)$$

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

12.6 El Problema General con Restricciones

Recordando lo visto en la sección (12.3) en lo referente a la definición del problema original Ec.(12.24). Entonces “Un vector $\hat{u} \in \widehat{W}$ es solución del problema original, si y sólo si, $\underline{u}' = R\hat{u} \in W_r \subset W$ satisface la expresión

$$\underline{a}A\underline{u}' = \underline{f} \quad \text{y} \quad \underline{j}\underline{u}' = 0 \quad (12.74)$$

el vector

$$\underline{f} \equiv (R\hat{f}) \in W_{12} \subset W_r \quad (12.75)$$

puede ser escrito como

$$\underline{f} \equiv \underline{f}_{\Pi} + \underline{f}_{\Delta} \quad (12.76)$$

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

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

Un vector $\underline{u}' \in W$ satisface la Ec.(12.74) si y sólo si, satisface la Ec.(12.61). Para derivar este resultado se usa la propiedad de que cuando $\underline{u}' \in W$ y $\underline{j}\underline{u}' = 0$, la siguiente relación se satisface

$$\underline{u}' \in W_r \quad \text{y} \quad \underline{a}(\underline{a}^r A^t \underline{a}^r) \underline{u}' = \underline{a}A^t \underline{u}'. \quad (12.77)$$

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

Sea $\underline{u}' \in W_r$ una solución de la Ec.(12.74), entonces $\underline{u}' \in W_{12} \subset W$ necesariamente, ya que $\underline{j}\underline{u}' = 0$ y se puede aplicar la inversa de la proyección natural obteniendo

$$\hat{u} = \underline{R}\underline{u}' \quad (12.78)$$

ya que este problema es formulado en el espacio de vectores derivados; en los algoritmos que se presentan en este trabajo (véase [36]), todas las operaciones

son realizadas en dicho espacio; en particular, para realizar los cálculos, nunca se regresa al espacio vectorial original \widehat{W} , excepto al final cuando se aplica la Ec.(12.78).

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

A nivel continuo, los algoritmos más estudiados son el Neumann-Neumann y el Dirichlet-Dirichlet (véase [19] y [3]) y estos fueron bosquejados en este trabajo en el capítulo 11. Durante el desarrollo del esquema del espacio de vectores derivados (DVS), se muestra una clara correspondencia entre el procedimiento a nivel continuo, y el procedimiento a nivel discreto (véase [50]). Usando tal correspondencia el resultado desarrollado puede ser resumido de una forma breve y efectiva, como sigue:

1. Algoritmos no Precondicionados

- El Algoritmo del Complemento de Schur (la formulación Primal del problema Dirichlet-Dirichlet), sección (13.1.1).
- La formulación Dual del Problema Neumann-Neumann, sección (13.1.2).
- La formulación Primal del Problema Neumann-Neumann, sección (13.1.3).
- La segunda formulación Dual del Problema Neumann-Neumann, sección (13.1.4).

2. Algoritmos Precondicionados

- La Versión DVS del Algoritmo BDDC (la formulación Dirichlet-Dirichlet precondicionada), sección (13.2.1).
- La Versión DVS del Algoritmo FETI-DP (la formulación Dual precondicionada del problema Neumann-Neumann), sección (13.2.2).
- La formulación Primal Precondicionada del problema Neumann-Neumann, sección (13.2.3).
- La segunda formulación Dual Precondicionada del problema Neumann-Neumann, sección (13.2.4).

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

Para la discusión de todo el material de este capítulo se usa la notación de la sección (12.4), en especial la definición de $\underline{\underline{A}}$ en la Ec.(12.65)

$$\underline{\underline{A}} \equiv \begin{pmatrix} \underline{\underline{A}}_{\Pi\Pi} & \underline{\underline{A}}_{\Pi\Delta} \\ \underline{\underline{A}}_{\Delta\Pi} & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix} \quad (13.1)$$

y la dada por la Ec.(12.69) para la matriz del complemento de Schur

$$\underline{\underline{S}} = \underline{\underline{A}}_{\Delta\Delta} - \underline{\underline{A}}_{\Delta\Pi} \underline{\underline{A}}_{\Pi\Pi}^{-1} \underline{\underline{A}}_{\Pi\Delta}. \quad (13.2)$$

13.1 Algoritmos no Precondicionados

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

13.1.1 Algoritmo del Complemento de Schur

Sea $\underline{u} = \underline{u}' - \underline{\underline{A}}_{\Pi\Pi}^{-1} \underline{\bar{f}}_{\Pi}$, entonces la Ec.(12.74) del problema general con restricciones -véase sección (12.6)-

$$\underline{\underline{a}} \underline{\underline{A}} \underline{u}' = \underline{\bar{f}} \quad \text{y} \quad \underline{j} \underline{u}' = 0 \quad (13.3)$$

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

$$\underline{\underline{a}} \underline{\underline{S}} \underline{u}_{\Delta} = \underline{f}_{\Delta} \quad \text{y} \quad \underline{j} \underline{u}_{\Delta} = 0” \quad (13.4)$$

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

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

En el capítulo anterior se mostró la versión continua del problema Dirichlet-Dirichlet no preconditionado, de la cual la Ec.(13.4) es su versión discreta. Por lo tanto este algoritmo pudiera ser llamado ‘el algoritmo Dirichlet-Dirichlet no preconditionado’. Sin embargo, en lo que sigue, el algoritmo correspondiente a la Ec.(13.4) será referido como el ‘algoritmo del complemento de Schur’ ya que es la variante de una de las más simples formas del método de subestructuración la cual se detalla en la sección (19.7).

13.1.2 Formulación Dual del Problema Neumann-Neumann

Para iniciar este desarrollo, se toma la identidad

$$\underline{a}\underline{S} + \underline{j}\underline{S} = \underline{S} \quad (13.5)$$

esta última ecuación es clara ya que

$$\underline{a} + \underline{j} = \underline{I}. \quad (13.6)$$

Usando las Ecs.(13.4 y 13.5) juntas, implican que

$$\underline{S}u_{\Delta} = \underline{a}\underline{S}u_{\Delta} + \underline{j}\underline{S}u_{\Delta} = \underline{f}_{\Delta} - \underline{\lambda}_{\Delta} \quad (13.7)$$

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

$$\underline{\lambda}_{\Delta} = -\underline{j}\underline{S}u_{\Delta}. \quad (13.8)$$

Por lo tanto, $\underline{\lambda}_{\Delta} \in \underline{j}W_{\Delta}$. Entonces, el problema de encontrar \underline{u}_{Δ} ha sido transformado en uno, en que se debe encontrar ‘el multiplicador de Lagrange $\underline{\lambda}_{\Delta}$ ’, una vez encontrado $\underline{\lambda}_{\Delta}$ se puede aplicar \underline{S}^{-1} a la Ec.(13.7) para obtener

$$\underline{u}_{\Delta} = \underline{S}^{-1} \left(\underline{f}_{\Delta} - \underline{\lambda}_{\Delta} \right). \quad (13.9)$$

Además, en la Ec.(13.9), $\underline{u}_{\Delta} \in \underline{a}W_{\Delta}$, tal que

$$\underline{j}\underline{S}^{-1} \left(\underline{f}_{\Delta} - \underline{\lambda}_{\Delta} \right) = 0 \quad (13.10)$$

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

$$\underline{j}\underline{S}^{-1}\underline{\lambda}_{\Delta} = \underline{j}\underline{S}^{-1}\underline{f}_{\Delta} \quad \text{junto con} \quad \underline{a}\underline{\lambda}_{\Delta} = 0 \quad (13.11)$$

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

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

$$J(u) = \frac{1}{2} \underline{u} \cdot \underline{Su} - \underline{u} \cdot \underline{f} \rightarrow \min \left. \vphantom{J(u)} \right\} \begin{array}{l} \underline{j\underline{u}} = 0 \end{array} \quad (13.12)$$

tomando la variación en la Ec.(13.12), se obtiene

$$\underline{Su} + \underline{j\underline{\lambda}} = \underline{f} \quad \text{junto con} \quad \underline{j\underline{u}} = 0 \quad (13.13)$$

para asegurar la unicidad de $\underline{\lambda}$, se impone la condición de que $\underline{au} = 0$, tal que $\underline{j\underline{\lambda}} = \underline{\lambda}$. Entonces la Ec.(13.13) implica

$$\underline{aSu} + \underline{jSu} + \underline{\lambda} = \underline{f} \quad (13.14)$$

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

$$\underline{\lambda} = -\underline{jSu} \quad \text{y} \quad \underline{aSu} = \underline{f} \quad (13.15)$$

entonces la Ec.(13.8) es clara.

13.1.3 Formulación Primal del Problema Neumann-Neumann

Para iniciar este desarrollo, se toma la Ec.(13.4) del algoritmo del complemento de Schur -véase sección (13.1.1)- y multiplicando la primera igualdad por \underline{S}^{-1} , y observando que $\underline{af}_\Delta = \underline{f}_\Delta$ se obtiene

$$\underline{S}^{-1} \underline{aSu}_\Delta = \underline{S}^{-1} \underline{f}_\Delta = \underline{S}^{-1} \underline{af}_\Delta = \underline{S}^{-1} \underline{aS} \left(\underline{S}^{-1} \underline{f}_\Delta \right) \quad (13.16)$$

de este modo, la Ec.(13.4) puede ser transformada en

$$\underline{S}^{-1} \underline{aS} \left(\underline{u}_\Delta - \underline{S}^{-1} \underline{f}_\Delta \right) = 0 \quad \text{y} \quad \underline{j\underline{u}}_\Delta = 0 \quad (13.17)$$

o

$$\underline{aS} \left(\underline{u}_\Delta - \underline{S}^{-1} \underline{f}_\Delta \right) = 0 \quad \text{y} \quad \underline{j\underline{u}}_\Delta = 0. \quad (13.18)$$

Si se define

$$\underline{v}_\Delta = \underline{S}^{-1} \underline{f}_\Delta - \underline{u}_\Delta \quad (13.19)$$

entonces la Ec.(13.18) es transformada en

$$\underline{\underline{j}}v_{\Delta} = \underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\underline{f}}_{\Delta} \quad \text{y} \quad \underline{\underline{a}}\underline{\underline{S}}v_{\Delta} = 0 \quad (13.20)$$

la forma iterativa¹² de este algoritmo se obtiene al multiplicarlo por $\underline{\underline{S}}^{-1}$

$$\underline{\underline{S}}^{-1}\underline{\underline{j}}v_{\Delta} = \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\underline{f}}_{\Delta} \quad \text{y} \quad \underline{\underline{a}}\underline{\underline{S}}v_{\Delta} = 0. \quad (13.21)$$

Si la solución de la Ec.(13.4) es conocida, entonces $v_{\Delta} \in W_{\Delta}$, definida por la Ec.(13.19), satisface la Ec.(13.21); a la inversa, si $v_{\Delta} \in W_{\Delta}$ satisface la Ec.(13.21) entonces

$$\underline{\underline{u}}_{\Delta} \equiv \underline{\underline{S}}^{-1}\underline{\underline{f}}_{\Delta} - v_{\Delta} \quad (13.22)$$

es solución de la Ec.(13.4).

En lo sucesivo, el algoritmo iterativo definido por la Ec.(13.21) será referido como la ‘formulación libre de multiplicadores de Lagrange del problema no preconditionado Neumann-Neumann’.

13.1.4 Segunda Formulación Dual del Problema Neumann-Neumann

En este caso, se toma como inicio la Ec.(13.11) -véase sección (13.1.2)-. Nótese que la siguiente identidad se satisface

$$\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1} \left(\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1} \right) = \underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1} \quad (13.23)$$

entonces, se multiplica la primera igualdad en la Ec.(13.11) por $\underline{\underline{S}}$, obteniéndose

$$\begin{aligned} \underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1} \left(\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1} \right) \lambda_{\Delta} &= \underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1} \lambda_{\Delta} = \underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1} \left(\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1} \right) \underline{\underline{f}}_{\Delta} \\ \text{junto con} \quad \underline{\underline{a}}\lambda_{\Delta} &= 0 \end{aligned} \quad (13.24)$$

o

$$\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1} \left(\left(\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1} \right) \underline{\underline{f}}_{\Delta} - \lambda_{\Delta} \right) \quad \text{junto con} \quad \underline{\underline{a}}\lambda_{\Delta} = 0. \quad (13.25)$$

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

$$\underline{\underline{\mu}}_{\Delta} \equiv \underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\underline{f}}_{\Delta} - \lambda_{\Delta} \quad (13.26)$$

¹²Para que el algoritmo sea iterativo, se necesita que el dominio y contradominio sean el mismo espacio, que en este caso debe de ser W_{Δ} .

la Ec.(13.25) es transformada en

$$\underline{a}\underline{\mu}_\Delta = \underline{a}\underline{S}\underline{j}\underline{S}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{j}\underline{S}^{-1}\underline{\mu}_\Delta = 0. \quad (13.27)$$

Nótese que esta última ecuación es equivalente a la Ec.(13.25) ya que \underline{S}^{-1} es no singular. Si la solución de la Ec.(13.11) es conocida, entonces $\underline{\mu}_\Delta \in \underline{W}_\Delta$ definida por la Ec.(13.26) satisface la Ec.(13.27). A la inversa, si $\underline{\mu}_\Delta \in \underline{W}_\Delta$ satisface la Ec.(13.27), entonces

$$\lambda_\Delta \equiv \underline{S}\underline{j}\underline{S}^{-1}\underline{f}_\Delta - \underline{\mu}_\Delta \quad (13.28)$$

es solución de la Ec.(13.11).

Por otro lado, la Ec.(13.27) no define un algoritmo iterativo. Sin embargo, si se multiplica la Ec.(13.27) por \underline{S} se obtiene el siguiente algoritmo iterativo

$$\underline{S}\underline{a}\underline{\mu}_\Delta = \underline{S}\underline{a}\underline{S}\underline{j}\underline{S}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{j}\underline{S}^{-1}\underline{\mu}_\Delta = 0 \quad (13.29)$$

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

13.2 Algoritmos Precondicionados

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

13.2.1 Versión DVS del Algoritmo BDDC

La versión DVS del algoritmo BDDC -una formulación de este algoritmo se da en la sección (14.2)- se obtiene cuando el algoritmo de complemento de Schur es preconditionado por medio de la matriz $\underline{a}\underline{S}^{-1}$. Entonces: “Dada $\underline{f}_\Delta \in \underline{a}\underline{W}_\Delta$, encontrar $\underline{u}_\Delta \in \underline{W}_\Delta$ tal que

$$\underline{a}\underline{S}^{-1}\underline{a}\underline{S}\underline{u}_\Delta = \underline{a}\underline{S}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{j}\underline{u}_\Delta = 0”. \quad (13.30)$$

Para este algoritmo, las siguientes propiedades llaman la atención:

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

$$\underline{\underline{aS^{-1}w}} = 0 \quad \text{y} \quad \underline{\underline{jw}} = 0 \Rightarrow \underline{w} = 0 \quad (13.31)$$

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

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

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

Para la propiedad 4), cuando ésta se satisface, la Ec.(13.30) implica la Ec.(13.4). Para ver esto, nótese que

$$\underline{\underline{j}}(\underline{\underline{aS}u_{\Delta}} - \underline{f}_{\Delta}) = 0 \quad (13.32)$$

y también que la Ec.(13.30) implica

$$\underline{\underline{aS^{-1}}}(\underline{\underline{aS}u_{\Delta}} - \underline{f}_{\Delta}) = 0 \quad (13.33)$$

cuando la implicación de la Ec.(13.31) se satisface, las Ecs.(13.32 y 13.33) juntas implican que

$$\underline{\underline{aS}u_{\Delta}} - \underline{f}_{\Delta} = 0 \quad (13.34)$$

como se quería. Esto muestra que la Ec.(13.30) implica la Ec.(13.4), cuando la Ec.(13.31) se satisface.

Para la propiedad 5), nótese que la condición de la Ec.(13.31) es débil y requiere que la matriz del complemento de Schur $\underline{\underline{S}}$ sea definida, ya que la implicación de la Ec.(13.31) es siempre satisfecha cuando $\underline{\underline{S}}$ es definida.

Asumiendo que $\underline{\underline{S}}$ es definida, entonces para cualquier vector $\underline{w} \in W_\Delta$ tal que $\underline{\underline{aS}}^{-1}\underline{w} = 0$ y $\underline{\underline{jw}} = 0$, se obtiene

$$\underline{w} \cdot \underline{\underline{S}}^{-1}\underline{w} = \underline{w} \cdot \underline{\underline{jS}}^{-1}\underline{w} = \left(\underline{\underline{jw}} \right) \cdot \underline{\underline{S}}^{-1}\underline{w} = 0 \quad (13.35)$$

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

13.2.2 Versión DVS del Algoritmo FETI-DP

La versión DVS del algoritmo FETI-DP -una formulación de este algoritmo se da en la sección (14.1)- se obtiene cuando la formulación con Multiplicadores de Lagrange del problema no preconditionado Neumann-Neumann de la Ec.(13.11) -véase sección (13.1.2)- es preconditionada por medio de la matriz $\underline{\underline{jS}}$. Entonces: “Dada $\underline{f}_\Delta \in \underline{\underline{aW}}_\Delta$, encontrar $\underline{\lambda}_\Delta \in W_\Delta$ tal que

$$\underline{\underline{jSj}}\underline{\underline{S}}^{-1}\underline{\lambda}_\Delta = \underline{\underline{jSj}}\underline{\underline{S}}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{\underline{a\lambda}}_\Delta = 0” \quad (13.36)$$

donde

$$\underline{u}_\Delta = \underline{\underline{aS}}^{-1} \left(\underline{f}_\Delta - \underline{\underline{j\lambda}}_\Delta \right). \quad (13.37)$$

Para este algoritmo, las siguientes propiedades llaman la atención:

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

$$\underline{\underline{jSw}} = 0 \quad \text{y} \quad \underline{\underline{aw}} = 0 \Rightarrow \underline{w} = 0 \quad (13.38)$$

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

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

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

Para la propiedad 4), cuando esta se satisface, la Ec.(13.36) implica la Ec.(13.11). Para ver esto, se asume la Ec.(13.36) y nótese que

$$\underline{a}\left(\underline{j}\underline{S}^{-1}\lambda_\Delta - \underline{j}\underline{S}^{-1}f_\Delta\right) = 0 \quad (13.39)$$

y también que la Ec.(13.36) implica

$$\underline{j}\underline{S}\left(\underline{j}\underline{S}^{-1}\lambda_\Delta - \underline{j}\underline{S}^{-1}f_\Delta\right) = 0 \quad (13.40)$$

cuando la implicación de la Ec.(13.38) se satisface, las Ecs.(13.39 y 13.40) juntas implican que

$$\underline{j}\underline{S}^{-1}\lambda_\Delta - \underline{j}\underline{S}^{-1}f_\Delta = 0 \quad (13.41)$$

como se quería. Esto muestra que la Ec.(13.36) implica la Ec.(13.11), cuando la Ec.(13.38) se satisface.

Para la propiedad 5), nótese que la condición de la Ec.(13.38) es débil y requiere que la matriz del complemento de Schur \underline{S} sea definida, ya que la implicación de la Ec.(13.38) es siempre satisfecha cuando \underline{S} es definida. Asumiendo que \underline{S} es definida, entonces para cualquier vector $\underline{\mu} \in W_\Delta$ tal que $\underline{j}\underline{S}\underline{\mu} = 0$ y $\underline{a}\underline{\mu} = 0$, se obtiene

$$\underline{\mu} \cdot \underline{S}\underline{\mu} = \underline{\mu} \cdot \underline{a}\underline{S}\underline{\mu} = (\underline{a}\underline{\mu}) \cdot \underline{S}\underline{\mu} = 0 \quad (13.42)$$

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

13.2.3 Formulación Primal Precondicionada del Problema Neumann-Neumann

Este algoritmo es la versión precondicionada de la formulación libre de multiplicadores de Lagrange del problema no precondicionado Neumann-Neumann. Este puede derivarse multiplicando la Ec.(13.20) -véase sección (13.1.3)-

por el preconditionador $\underline{\underline{S}}^{-1}\underline{\underline{jS}}$. “Entonces el algoritmo consiste en buscar $\underline{v}_\Delta \in W_\Delta$, tal que

$$\underline{\underline{S}}^{-1}\underline{\underline{jS}}\underline{\underline{jv}}_\Delta = \underline{\underline{S}}^{-1}\underline{\underline{jS}}\underline{\underline{jS}}^{-1}\underline{f}_\Delta \quad \text{y} \quad \underline{aSv}_\Delta = 0 \quad (13.43)$$

donde

$$\underline{u}_\Delta = \underline{aS}^{-1} \left(\underline{f}_\Delta - \underline{\underline{jS}}\underline{\underline{jv}}_\Delta \right). \quad (13.44)$$

Para este algoritmo, las siguientes propiedades llaman la atención:

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

$$\underline{\underline{jS}}\underline{w} = 0 \quad \text{y} \quad \underline{aw} = 0 \Rightarrow \underline{w} = 0 \quad (13.45)$$

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

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

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

Para la propiedad 4), cuando esta se satisface, la Ec.(13.43) implica la Ec.(13.20). Pare ver esto, se asume la Ec.(13.43) y se define

$$\underline{w} = \underline{\underline{jv}}_\Delta - \underline{\underline{jS}}^{-1}\underline{f}_\Delta \quad \text{tal que} \quad \underline{aw} = 0 \quad (13.46)$$

además, en vista de la Ec.(13.43) implica

$$\underline{\underline{S}}^{-1}\underline{\underline{jS}}\underline{w} = 0 \quad \text{y por lo tanto} \quad \underline{\underline{jS}}\underline{w} = 0 \quad (13.47)$$

usando la Ec.(13.45) se ve que las Ecs.(13.46 y 13.47) juntas implican que

$$\underline{j}v_{\Delta} - \underline{j}S^{-1}\underline{f}_{\Delta} = \underline{w} = 0 \quad (13.48)$$

ahora, la Ec.(13.20) es clara y la prueba se completa.

Para la propiedad 5), nótese que la condición de la Ec.(13.45) es débil y requiere que la matriz del complemento de Schur \underline{S} sea definida, ya que la implicación de la Ec.(13.45) es siempre satisfecha cuando \underline{S} es definida. Asumiendo que \underline{S} es definida, entonces para cualquier vector $\underline{w} \in W_{\Delta}$ tal que $\underline{j}S\underline{w} = 0$ y $\underline{a}w = 0$, se obtiene

$$\underline{w} \cdot \underline{S}w = \underline{w} \cdot \underline{a}S\underline{w} = (\underline{a}w) \cdot \underline{S}w = 0 \quad (13.49)$$

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

13.2.4 Segunda Formulación Dual Precondicionada del Problema Neumann-Neumann

Este algoritmo es la versión precondicionada de la segunda formulación con multiplicadores de Lagrange del problema no precondicionado Neumann-Neumann. Este puede derivarse multiplicando la Ec.(13.27) -véase sección (13.1.4)- por el preconditionador $\underline{S}^{-1}\underline{a}S$. “Entonces el algoritmo consiste en buscar $\underline{\mu}_{\Delta} \in W_{\Delta}$, tal que

$$\underline{S}a\underline{S}^{-1}\underline{a}\underline{\mu}_{\Delta} = \underline{S}a\underline{S}^{-1}\underline{a}S\underline{j}S^{-1}\underline{f}_{\Delta} \quad \text{y} \quad \underline{j}S^{-1}\underline{\mu}_{\Delta} = 0” \quad (13.50)$$

donde

$$\underline{u}_{\Delta} = \underline{a}S^{-1} \left(\underline{f}_{\Delta} + \underline{\mu}_{\Delta} \right) \quad (13.51)$$

este algoritmo es similar a FETI-DP.

Para este algoritmo, las siguientes propiedades llaman la atención:

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

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

$$\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{w}} = 0 \quad \text{y} \quad \underline{\underline{j}}\underline{\underline{w}} = 0 \Rightarrow \underline{\underline{w}} = 0 \quad (13.52)$$

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

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

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

Para la propiedad 4), cuando esta se satisface, la Ec.(13.50) implica la Ec.(13.27). Para ver esto, se asume la Ec.(13.50) y se define

$$\underline{\underline{\eta}} = \underline{\underline{a}}\underline{\underline{\mu}}_\Delta - \underline{\underline{a}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\underline{f}}_\Delta \quad \text{tal que} \quad \underline{\underline{j}}\underline{\underline{\eta}} = 0 \quad (13.53)$$

además, en vista de la Ec.(13.50) se obtiene

$$\underline{\underline{Sa}}\underline{\underline{S}}^{-1}\underline{\underline{\eta}} = 0 \quad (13.54)$$

usando las Ecs.(13.53 y 13.54), juntas implican que

$$\underline{\underline{a}}\underline{\underline{\eta}} - \underline{\underline{a}}\underline{\underline{S}}\underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\underline{f}}_\Delta = \underline{\underline{\eta}} = 0 \quad (13.55)$$

ahora, la Ec.(13.27) es clara, como se quería probar, además se ve que la Ec.(13.50) implica la Ec.(13.27), cuando la condición de la Ec.(13.52) se satisface.

Para la propiedad 5), nótese que la condición de la Ec.(13.52) es débil y requiere que la matriz del complemento de Schur $\underline{\underline{S}}$ sea definida, ya que la implicación de la Ec.(13.52) es siempre satisfecha cuando $\underline{\underline{S}}$ es definida. Asumiendo que $\underline{\underline{S}}$ es definida, entonces para cualquier vector $\underline{\underline{w}} \in W_\Delta$ tal que $\underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{w}} = 0$ y $\underline{\underline{j}}\underline{\underline{w}} = 0$, se obtiene

$$\underline{\underline{w}} \cdot \underline{\underline{S}}^{-1}\underline{\underline{w}} = \underline{\underline{w}} \cdot \underline{\underline{j}}\underline{\underline{S}}^{-1}\underline{\underline{w}} = \left(\underline{\underline{j}}\underline{\underline{w}} \right) \cdot \underline{\underline{S}}^{-1}\underline{\underline{w}} = 0 \quad (13.56)$$

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

13.3 El Operador de Steklov-Poincaré

El operador de Steklov-Poincaré generalmente se define como la ecuación para la traza de la solución exacta u sobre Γ del problema dado por la Ec.(11.4), donde el complemento de Schur -definido en la Ec.(12.70)- es una aproximación discreta del operador de Steklov-Poincaré (véase [51]). La discusión de este operador juega un papel importante en el desarrollo de la teoría del espacio de vectores derivados (véase [50]), para iniciar la discusión, se usa la siguiente definición.

Definición 115 Sea $\underline{\underline{A}} : W_r \rightarrow W_r$ una matriz simétrica y positiva definida. El ‘producto interior de energía’ es definido por

$$(\underline{u}, \underline{w}) \equiv \underline{u} \cdot \underline{\underline{A}} \underline{w} \quad \forall \underline{u}, \underline{w} \in W_r. \quad (13.57)$$

El espacio lineal, W_r , es un espacio de Hilbert (dimensionalmente finito) cuando este es dotado con el producto interior de energía. Usando la notación de la sección (12.4) y recordando que la matriz $\underline{\underline{A}}$ se escribe como

$$\underline{\underline{A}} \equiv \begin{pmatrix} \underline{\underline{A}}_{\Pi\Pi} & \underline{\underline{A}}_{\Pi\Delta} \\ \underline{\underline{A}}_{\Delta\Pi} & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix} \quad (13.58)$$

donde la notación usada es tal que

$$\begin{cases} \underline{\underline{A}}_{\Pi\Pi} : W_{\Pi} \rightarrow W_{\Pi}, & \underline{\underline{A}}_{\Pi\Delta} : W_{\Delta} \rightarrow W_{\Pi} \\ \underline{\underline{A}}_{\Delta\Pi} : W_{\Pi} \rightarrow W_{\Delta}, & \underline{\underline{A}}_{\Delta\Delta} : W_{\Delta} \rightarrow W_{\Delta} \end{cases} \quad (13.59)$$

además

$$\begin{cases} \underline{\underline{A}}_{\Pi\Pi} \underline{u} = (\underline{\underline{A}} \underline{u}_{\Pi})_{\Pi}, & \underline{\underline{A}}_{\Delta\Pi} \underline{u} = (\underline{\underline{A}} \underline{u}_{\Pi})_{\Delta} \\ \underline{\underline{A}}_{\Pi\Delta} \underline{u} = (\underline{\underline{A}} \underline{u}_{\Delta})_{\Pi}, & \underline{\underline{A}}_{\Delta\Delta} \underline{u} = (\underline{\underline{A}} \underline{u}_{\Delta})_{\Delta} \end{cases} \quad (13.60)$$

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

$$\begin{aligned} \underline{\underline{A}}_{\Pi\Pi} &\equiv \begin{pmatrix} \underline{\underline{A}}_{\Pi\Pi} & 0 \\ 0 & 0 \end{pmatrix} \\ \underline{\underline{A}}_{\Pi\Delta} &\equiv \begin{pmatrix} 0 & \underline{\underline{A}}_{\Pi\Delta} \\ 0 & 0 \end{pmatrix} \\ \underline{\underline{A}}_{\Delta\Pi} &\equiv \begin{pmatrix} 0 & 0 \\ \underline{\underline{A}}_{\Delta\Pi} & 0 \end{pmatrix} \\ \underline{\underline{A}}_{\Delta\Delta} &\equiv \begin{pmatrix} 0 & 0 \\ 0 & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix}. \end{aligned} \quad (13.61)$$

Ahora, se introducen las siguientes definiciones:

Definición 116 Sea la matriz $\underline{\underline{L}}$ definida como

$$\underline{\underline{L}} \equiv \begin{pmatrix} \underline{\underline{A}}_{\Pi\Pi} & \underline{\underline{A}}_{\Pi\Delta} \\ 0 & 0 \end{pmatrix} \quad (13.62)$$

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

$$\underline{\underline{R}} \equiv \begin{pmatrix} 0 & 0 \\ \underline{\underline{A}}_{\Delta\Pi} & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix}. \quad (13.63)$$

Además, nótese la siguiente identidad

$$\underline{\underline{R}} = \underline{\underline{aR}} + \underline{\underline{jR}} \quad (13.64)$$

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

$$\underline{\underline{L}} + \underline{\underline{aR}} + \underline{\underline{jR}} = \underline{\underline{L}}^T + \underline{\underline{R}}^T \underline{\underline{a}} + \underline{\underline{R}}^T \underline{\underline{j}}. \quad (13.65)$$

Definición 117 A la identidad

$$\underline{\underline{L}} + \underline{\underline{aR}} - \underline{\underline{R}}^T \underline{\underline{j}} = \underline{\underline{L}}^T + \underline{\underline{R}}^T \underline{\underline{a}} - \underline{\underline{jR}} \quad (13.66)$$

la cual se deriva de la Ec.(13.65), la cual será referida como la fórmula Green-Herrera para matrices.

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

$$\left(\underline{\underline{L}} + \underline{\underline{aR}} - \underline{\underline{R}}^T \underline{\underline{j}} \right) \underline{v} = 0 \quad (13.67)$$

si y sólo si

$$\underline{\underline{Lv}} = 0, \underline{\underline{aRv}} = 0 \quad \text{y} \quad \underline{\underline{jv}} = 0. \quad (13.68)$$

Esto establece la equivalencia entre las Ec.(13.67) y Ec.(13.68) y se puede usar el hecho de que los rangos de $\underline{\underline{L}}$ y $\underline{\underline{aR}}$ son linealmente independientes, junto con la ecuación

$$\left(\underline{\underline{jv}} \right) \cdot \underline{\underline{R}}^T \underline{\underline{jv}} = \left(\underline{\underline{jv}} \right) \cdot \underline{\underline{A}}_{\Delta\Delta} \underline{\underline{jv}} = \left(\underline{\underline{jv}} \right) \cdot \underline{\underline{A}} \underline{\underline{jv}} = 0 \quad (13.69)$$

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

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

$$\hat{\underline{u}} \equiv \underline{au} \quad y \quad \llbracket \underline{u} \rrbracket \equiv \underline{ju} \quad (13.70)$$

entonces $\hat{\underline{u}} \in W_r$, mientras $\llbracket \underline{u} \rrbracket$ pertenecen a $W_{11}(\Gamma) \subset W_r$. La fórmula de Green-Herrera de la Ec.(13.66) es equivalente a

$$\underline{w} \cdot \underline{Lu} + \hat{\underline{w}} \cdot \underline{aRu} - \llbracket \underline{u} \rrbracket \underline{jRw} = \underline{u} \cdot \underline{Lw} + \hat{\underline{u}} \cdot \underline{aRw} - \llbracket \underline{w} \rrbracket \underline{jRu} \quad (13.71)$$

$\forall \underline{u}, \underline{w} \in W_r$. Ahora, las fórmulas de Green-Herrera que originalmente fueron introducidas para operadores diferenciales parciales actuando sobre funciones discontinuas, pueden ser aplicadas a cualquier operador cuando este es lineal. La Ec.(13.66) por otro lado, es vista como una extensión de este tipo de fórmulas actuando sobre vectores discontinuos y se tiene interés de comparar la Ec.(13.71) con las fórmulas de Green-Herrera para operadores diferenciales parciales. Para hacer esto, se introduce la siguiente notación

$$\llbracket \underline{R} \rrbracket = -\underline{aR} \quad y \quad \hat{\underline{R}} \equiv -\underline{jR} \quad (13.72)$$

haciendo uso de esta notación, la Ec.(13.71) se reescribe como

$$\underline{w} \cdot \underline{Lu} + \llbracket \underline{u} \rrbracket \cdot \hat{\underline{R}} \underline{w} - \hat{\underline{w}} \cdot \llbracket \underline{R} \rrbracket \underline{u} = \underline{u} \cdot \underline{Lw} + \llbracket \underline{w} \rrbracket \cdot \hat{\underline{R}} \underline{u} - \hat{\underline{u}} \cdot \llbracket \underline{R} \rrbracket \underline{w} \quad (13.73)$$

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

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

$$\begin{aligned} \int_{\Omega} w \mathcal{L}u dx + \int_{\Gamma} \left\{ \llbracket \underline{u} \rrbracket \frac{\hat{\partial w}}{\partial n} - \hat{\underline{w}} \llbracket \underline{\frac{\partial u}{\partial n}} \rrbracket \right\} dx = \\ \int_{\Omega} u \mathcal{L}w dx + \int_{\Gamma} \left\{ \llbracket \underline{w} \rrbracket \frac{\hat{\partial u}}{\partial n} - \hat{\underline{u}} \llbracket \underline{\frac{\partial w}{\partial n}} \rrbracket \right\} dx \end{aligned} \quad (13.74)$$

la siguiente correspondencia entre las funcionales bilineales involucradas en ambas ecuaciones, comparando con las Ecs. (13.73 y 13.74) se obtiene

$$\begin{aligned} \int_{\Omega} w \mathcal{L}u dx &\leftrightarrow \underline{w} \cdot \underline{Lu} \\ \int_{\Gamma} \llbracket u \rrbracket \widehat{\frac{\partial w}{\partial n}} dx &\leftrightarrow \llbracket u \rrbracket \cdot \widehat{\underline{R}} w \\ \int_{\Gamma} \widehat{w} \llbracket \left[\frac{\partial u}{\partial n} \right] \rrbracket dx &\leftrightarrow \widehat{w} \cdot \llbracket \underline{R} \rrbracket \underline{u}. \end{aligned} \quad (13.75)$$

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

$$\int_{\Gamma} \widehat{w} \llbracket \left[\frac{\partial u}{\partial n} \right] \rrbracket dx \quad (13.76)$$

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

$$\widehat{w} \cdot \llbracket \underline{R} \rrbracket \underline{u} = \underline{w} \cdot \underline{aRu}, \forall \underline{u}, \underline{w} \in W_r \quad (13.77)$$

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

Definición 118 *Se define al operador de Steklov-Poincaré como la matriz*

$$\underline{aR}. \quad (13.78)$$

En el esquema de vectores derivados (DVS), la versión discreta del operador de Steklov-Poincaré es \underline{aS} , por lo tanto, la versión discreta de la Ec.(11.14) es

$$\underline{aSv} = \underline{aS}u_p \quad \text{junto con } \underline{jv} = 0 \quad (13.79)$$

la cual puede ponerse en correspondencia con la Ec.(13.4) reemplazando

$$- \llbracket \left[\frac{\partial u_p}{\partial n} \right] \rrbracket \leftrightarrow \underline{f}_{\Delta} \quad \text{y} \quad \underline{v} \leftrightarrow \underline{u}_{\Delta}. \quad (13.80)$$

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

$$\int_{\Omega} w \mathcal{L}u dx + \int_{\Gamma} \left\{ \llbracket u \rrbracket \widehat{\underline{a}_n} \cdot \widehat{\nabla} w - \widehat{w} \llbracket \underline{a}_n \cdot \nabla u \rrbracket \right\} dx =$$

$$\int_{\Omega} u \mathcal{L} w dx + \int_{\Gamma} \left\{ \llbracket w \rrbracket \widehat{\underline{a}_n \cdot \nabla} u - \widehat{u} \llbracket \underline{a}_n \cdot \nabla w \rrbracket \right\} dx. \quad (13.81)$$

La correspondencia de la ecuación (13.75) todavía permanece para este caso, excepto que

$$\begin{cases} \llbracket \underline{a}_n \cdot \nabla u \rrbracket \leftrightarrow - \llbracket \underline{R} \rrbracket u \\ \widehat{\underline{a}_n \cdot \nabla} w \leftrightarrow - \widehat{\underline{R} u} \end{cases}. \quad (13.82)$$

Correspondencias similares a las dadas por las Ecs. (13.75 y 13.82) pueden ser establecidas en general -su aplicación incluye a los sistemas gobernantes de ecuaciones de elasticidad lineal y muchos problemas más-. Nótese que las Ecs. (13.75 y 13.82) implican una nueva fórmula para el operador *Steklov-Poincaré*, i.e., el salto de la derivada normal en el nivel discreto, el cual es diferente a las interpretaciones estándar que han sido presentadas por muchos autores. La fórmula (véase [34]) para el operador *Steklov-Poincaré* es

$$- \llbracket \underline{R} \rrbracket u \equiv -j \underline{R} \quad (13.83)$$

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

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

Teorema 119 Sea $\underline{f} = \begin{pmatrix} \underline{f}_{\Pi} \\ \underline{f}_{\Delta} \end{pmatrix} \in W_r = W_{12}(\overline{\Omega})$, entonces una $\underline{v} \in W_r$ satisface

$$\left(\underline{L} + \underline{aR} - \underline{R}^T \underline{j} \right) \underline{v} = \underline{f} \quad (13.84)$$

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

Demostración. Sea $\underline{u} \in W_r$ una solución del problema transformado y asumiendo que $\underline{v} \in W_r$ satisface la Ec.(13.84) entonces

$$\left(\underline{L} + \underline{aR} - \underline{R}^T \underline{j} \right) \underline{u} = \left(\underline{L} + \underline{aR} \right) \underline{u} = \underline{aAu} = \underline{f} \quad (13.85)$$

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

$$\left(\underline{L} + \underline{aR} - \underline{R}^T \underline{j} \right) \underline{w} = 0 \quad (13.86)$$

y usando las Ec.(13.67) y Ec.(13.68), se tiene que $\underline{v} = \underline{u} + \underline{w}$ satisfice

$$\underline{A}\underline{v} = (\underline{L} + \underline{aR})\underline{v} = (\underline{L} + \underline{aR})\underline{u} = \underline{aAu} = \underline{f} \quad (13.87)$$

y

$$\underline{j}\underline{v} = \underline{j}\underline{u} = 0. \quad (13.88)$$

■

Por otro lado, para obtener la versión discreta del operador μ definido en la Ec.(11.21), primero se escribe la versión discreta de la Ec.(11.19), esta es

$$\underline{j}\underline{S}\underline{v} = \underline{q}_\Gamma \quad \text{junto con} \quad \underline{aS}\underline{v} = 0 \quad (13.89)$$

por lo tanto, se tiene que

$$\underline{a}\underline{q}_\Gamma = 0 \text{ y } \underline{j}\underline{v} = \underline{j}\underline{S}^{-1}\underline{S}\underline{v} = \underline{j}\underline{S}^{-1}\underline{j}\underline{S}\underline{v} = \underline{j}\underline{S}^{-1}\underline{q}_\Gamma \quad (13.90)$$

esto establece la correspondencia de

$$\mu \leftrightarrow \underline{j}\underline{S}^{-1} \quad (13.91)$$

la versión discreta de la Ec.(11.22) es

$$\underline{j}\underline{S}^{-1}\underline{q}_\Gamma = -\underline{j}\underline{u}_p \quad \text{junto con} \quad \underline{a}\underline{q}_\Gamma = 0. \quad (13.92)$$

Otra opción de abordar la versión discreta de la Ec.(11.20) sin hacer uso de la contraparte del operador de Steklov-Poincaré μ , en el cual, el correspondiente problema es

$$\underline{j}\underline{v} = -\underline{j}\underline{u}_p \quad \text{y} \quad \underline{aS}\underline{v} = 0 \quad (13.93)$$

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

$$\underline{aS}\underline{j} \neq 0. \quad (13.94)$$

Una ecuación equivalente a la Ec.(13.93), la cual puede ser aplicada en un algoritmo iterativo es

$$\underline{S}^{-1}\underline{j}\underline{v} = -\underline{S}^{-1}\underline{j}\underline{u}_p \quad \text{y} \quad \underline{aS}\underline{v} = 0. \quad (13.95)$$

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

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

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

$$\begin{aligned}\mathcal{L}u &= f \quad \text{en } \Omega \\ u &= g \quad \text{sobre } \partial\Omega\end{aligned}\tag{14.1}$$

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

$$\Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j \quad \text{y} \quad \bar{\Omega} = \bigcup_{i=1}^E \bar{\Omega}_i\tag{14.2}$$

y al conjunto

$$\Gamma = \bigcup_{i=1}^E \Gamma_i, \quad \text{si } \Gamma_i = \partial\Omega_i \setminus \partial\Omega\tag{14.3}$$

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

$$\partial\Omega \subset \bigcup_{i=1}^E \partial\Omega_i \quad \text{y} \quad \Omega = \left(\bigcup_{i=1}^E \Omega_i \right) \cup \Gamma.\tag{14.4}$$

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

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

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

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} u_I \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_I \\ f_\Gamma \end{pmatrix} \quad (14.5)$$

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

$$A_{II} = \begin{pmatrix} A_{II}^{(1)} & & 0 \\ & \ddots & \\ 0 & \cdots & A_{II}^{(E)} \end{pmatrix}. \quad (14.6)$$

Eliminando las variables asociadas a los nodos interiores u_I -véase sección (19.7)-, se obtiene el complemento de Schur

$$Su_\Gamma = \tilde{f}_\Gamma \quad (14.7)$$

donde

$$S = A_{\Gamma\Gamma} - A_{\Gamma I} (A_{II})^{-1} A_{I\Gamma} \quad (14.8)$$

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

$$S_i = A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)} \right)^{-1} A_{I\Gamma}^{(i)}. \quad (14.9)$$

14.1 El Método FETI-DP en el Marco de DVS

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

$$\left. \begin{aligned} J(u) = \frac{1}{2} \langle Su, u \rangle - \langle f, u \rangle \rightarrow \min \\ B_r u = 0 \end{aligned} \right\} \quad (14.10)$$

donde la matriz B_r denota al operador de salto

$$B_r = [B_r^{(1)}, B_r^{(2)}, \dots, B_r^{(E)}] \quad (14.11)$$

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

$$B_r u = 0 \quad (14.12)$$

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

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

$$B_{D_r} = (D_r^{(1)} B_r^{(1)}, D_r^{(2)} B_r^{(2)}, \dots, D_r^{(E)} B_r^{(E)}). \quad (14.13)$$

Con la introducción de un vector de Multiplicadores de Lagrange λ para hacer cumplir las restricciones $B_r u = 0$, se obtiene una formulación punto silla de la Ec.(14.10): Encontrar $(u, \lambda) \in W \times U$ tal que

$$\begin{cases} Su + B_r^T \lambda = f \\ B_r u = 0 \end{cases} \quad (14.14)$$

sustituyendo la expresión para u en la segunda ecuación de Ec.(14.14), entonces

$$B_r S^\dagger B_r^T \lambda = B_r S^\dagger f - B_r R \alpha \quad (14.15)$$

y finalmente se obtiene el sistema

$$\begin{cases} F_r \lambda - G_r \alpha = d_r \\ G_r^T \lambda = e \end{cases} \quad (14.16)$$

La matriz del sistema lineal reducido puede ser escrito como

$$F_r = B_r S^\dagger B_r^T \quad (14.17)$$

donde el preconditionador de FETI esta dado por

$$\widehat{M}_r^{-1} = B_{D_r} S B_{D_r}^T = \sum_{i=1}^E D_r^{(i)} B_r^{(i)} S^{(i)} B_r^{(i)T} D_r^{(i)} \quad (14.18)$$

que define el sistema preconditionado

$$P_r \widehat{M}_r^{-1} P_r^T F_r \lambda_r = P_r \widehat{M}_r^{-1} P_r^T d_r \quad (14.19)$$

con

$$P_r = I - Q_r G_r (G_r^T Q_r G_r)^{-1} G_r^T \quad (14.20)$$

donde

$$G_r = B_r R \quad \text{y} \quad d_r = B_r S^\dagger f \quad (14.21)$$

con la condición inicial λ_0 escogida tal que

$$G_r \lambda_0 = e \quad (14.22)$$

donde $e = R^T f$.

Desarrollando la expresión dada por la Ec.(14.19) y reemplazando \widehat{M}_r^{-1} de la Ec.(14.18), se obtiene

$$P_r (B_{D_r} S B_{D_r}^T) P_r^T (B_r S^\dagger B_r^T) \lambda_r = P_r (B_{D_r} S B_{D_r}^T) P_r^T (B_r S^\dagger f) \quad (14.23)$$

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

$$(B_{D_r} S B_{D_r}^T) (B_r S^\dagger B_r^T) \lambda_r = (B_{D_r} S B_{D_r}^T) (B_r S^\dagger f) \quad (14.24)$$

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

$$D_r B_r S (D_r B_r)^T B_r S^\dagger B_r^T \lambda_r = D_r B_r S (D_r B_r)^T B_r S^\dagger f \quad (14.25)$$

y simplificando, se obtiene

$$D_r B_r S B_r^T D_r B_r S^\dagger B_r^T \lambda_r = D_r B_r S B_r^T D_r B_r S^\dagger f. \quad (14.26)$$

Ahora, ya que B_r sólo debe satisfacer la restricción $B_r u = 0$, entonces es posible sustituir con j , definida como $j = I - a$ -véase la Ec.(13.6)-, ésta también satisface $j u = 0$. Entonces reemplazando B_r por j , se obtiene

$$D_r j S j^T D_r j S^\dagger j^T \lambda_r = D_r j S j^T D_r j S^\dagger f \quad (14.27)$$

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

$$D_r j S j D_r j S^\dagger j \lambda_r = D_r j S j D_r j S^\dagger f \quad (14.28)$$

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

$$D_r j S j D_r j S^{-1} j \lambda_r = D_r j S j D_r j S^{-1} f. \quad (14.29)$$

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

$$jSjjS^{-1}j\lambda_r = jSjjS^{-1}f \quad (14.30)$$

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

$$jSjS^{-1}j\lambda_r = jSjS^{-1}f \quad (14.31)$$

esta es una formulación equivalente a la formulación del método de FETI preconditionado en términos de DVS -véase la Ec.(13.36)-.

Ahora, se desarrolla la expresión dada por la Ec.(14.19) sin preconditionar

$$F_r\lambda_r = d_r \quad (14.32)$$

sustituyendo F_r y d_r , entonces

$$B_rS^\dagger B_r^T\lambda_r = B_rS^\dagger f \quad (14.33)$$

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

$$jS^{-1}j\lambda_r = jS^{-1}f \quad (14.34)$$

esta es una formulación equivalente al método de FETI sin preconditionar en términos del esquema DVS -véase la Ec.(13.36)-.

14.2 El Método BDDC en el Marco de DVS

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

$$S_i = A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)} \right)^{-1} A_{I\Gamma}^{(i)} \quad (14.35)$$

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

$$S = \sum_{i=1}^E \bar{R}_i^T S_i \bar{R}_i. \quad (14.36)$$

Observación 6 *Nótese que*

$$S_i \neq \bar{R}_i S \bar{R}_i^T \quad (14.37)$$

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

$$u^T S_i u \leq u^T (\bar{R}_i S \bar{R}_i^T) u \quad (14.38)$$

ya que S y S_i son positivas definidas.

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

$$M^{-1} = \sum_{i=1}^E R_i^T S_i^{-1} R_i \quad (14.39)$$

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

$$R_i = D_i^{-1} \bar{R}_i \quad (14.40)$$

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

$$\bar{R}_i : \Gamma \rightarrow \Gamma_i \quad (14.41)$$

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

$$\sum_{i=1}^N \bar{R}_i^T D_i^{-1} \bar{R}_i = I \quad (14.42)$$

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

$$\delta_i(x) = \{\text{numero de subdominios que comparten el nodo } x \in \Gamma_i\}. \quad (14.43)$$

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

El método BDD preconditionado queda en términos de

$$M^{-1}Su = M^{-1}f \quad (14.44)$$

sustituyendo S y M^{-1} -Ecs.(14.8 y 14.39)-, se tiene que

$$\begin{aligned} & \left(\sum_{i=1}^E (D_i^{-1} \bar{R}_i)^T S_i^{-1} D_i^{-1} \bar{R}_i \right) \left(\sum_{i=1}^E \bar{R}_i^T S_i \bar{R}_i \right) u \\ &= \left(\sum_{i=1}^E (D_i^{-1} \bar{R}_i)^T S_i^{-1} D_i^{-1} \bar{R}_i \right) f \end{aligned} \quad (14.45)$$

y finalmente sustituyendo S_i , i.e Ec.(14.35), se obtiene

$$\begin{aligned} & \left(\sum_{i=1}^E \bar{R}_i^T D_i^{-1} \left(A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)} \right)^{-1} A_{I\Gamma}^{(i)} \right)^{-1} D_i^{-1} \bar{R}_i \right) \\ & \left(\sum_{i=1}^E \bar{R}_i^T \left(A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)} \right)^{-1} A_{I\Gamma}^{(i)} \right) \bar{R}_i \right) u \\ &= \left(\sum_{i=1}^E \bar{R}_i^T D_i^{-1} \left(A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)} \right)^{-1} A_{I\Gamma}^{(i)} \right)^{-1} D_i^{-1} \bar{R}_i \right) f. \end{aligned} \quad (14.46)$$

Multiplicando u por $\left(A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)} \right)^{-1} A_{I\Gamma}^{(i)} \right) \bar{R}_i$ se obtienen E vectores \tilde{u}_i , tales que al aplicar $D_i^{-1} \bar{R}_i \left(\sum_{i=1}^E \bar{R}_i^T \tilde{u}_i \right)$ es equivalente a aSu . A estos E vectores resultantes \hat{u}_i , se aplica $\sum_{i=1}^E \bar{R}_i^T D_i^{-1} \left(A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)} \right)^{-1} A_{I\Gamma}^{(i)} \right)^{-1} \hat{u}_i$, siendo equivalente a $aS^{-1}aSu$. Que es el lado derecho de la Ec.(13.30).

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

Así, la Ec.(14.46) es equivalente a

$$aS^{-1}aSu = aS^{-1}f \quad (14.47)$$

que es la formulación del método BDDC en términos del esquema DVS -véase Ec.(13.30)-.

14.3 Comparaciones

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

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

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

$$(\underline{\underline{S}}^t)^{-1} = \sum_{\alpha=1}^E (\underline{\underline{S}}^\alpha)^{-1} \quad (14.48)$$

una relación similar no es satisfecha por el algoritmo BDDC. A saber, la ecuación íntimamente relacionada es por ejemplo Ec.(14.36)

$$\underline{\underline{S}} = \sum_{\alpha=1}^E \underline{\underline{\bar{R}}}_\alpha^T \underline{\underline{S}}_\alpha \underline{\underline{\bar{R}}}_\alpha \quad (14.49)$$

sin embargo

$$(\underline{\underline{S}})^{-1} \neq \sum_{\alpha=1}^E \left(\underline{\underline{\bar{R}}}_\alpha^T \underline{\underline{S}}_\alpha \underline{\underline{\bar{R}}}_\alpha \right)^{-1} \quad (14.50)$$

aquí, la igualdad no se satisface debido a que los rangos de $\underline{\underline{\bar{R}}}_\alpha^T \underline{\underline{S}}_\alpha \underline{\underline{\bar{R}}}_\alpha$ y $\underline{\underline{\bar{R}}}_\beta^T \underline{\underline{S}}_\beta \underline{\underline{\bar{R}}}_\beta$ no son ajenos, cuando $\alpha \neq \beta$. Esta última limitación es debido

al hecho que en BDDC, la formulación no se establece directamente en el espacio producto y que en su implementación se retorna a los grados de libertad asociados con los nodos originales, mientras en el esquema DVS uno se olvida completamente de los nodos originales y se trabaja exclusivamente con los nodos derivados, es decir, de los nodos que resultan después de que los nodos originales han sido partidos.

Por lo tanto, las fórmulas desarrolladas para el esquema DVS y sus códigos computacionales se simplifican de manera notable. La unificación y simplificación lograda de esta manera permite producir Software genérico, robusto y eficiente.

15 Formulación Numérica de los Métodos DVS

Partiendo de las formulaciones Dirichlet-Dirichlet y Neumann-Neumann en el marco del espacio de vectores derivados (DVS) -véase sección (13)-, aquí se muestran los detalles de la formulación numérica la cual está íntimamente relacionada con la implementación computacional, tanto en la versión secuencial como paralela del código (véase [39] y [38]).

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

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

$$\begin{aligned}\mathcal{L}u &= f \quad \text{en } \Omega \\ u &= g \quad \text{sobre } \partial\Omega\end{aligned}\tag{15.1}$$

en el dominio Ω , el cual es subdividido en E subdominios Ω_α , $\alpha = 1, 2, \dots, E$ mediante una descomposición de dominio sin traslape -véase sección (12.2)-. Para resolver dicho problema, se puede usar cualquiera de los ocho algoritmos que se han desarrollado, de cada uno de ellos se han dado formulaciones explícitas en términos de matrices.

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

$$\underline{\underline{A}}_{\Pi\Pi}^\alpha, \underline{\underline{A}}_{\Pi\Delta}^\alpha, \underline{\underline{A}}_{\Delta\Pi}^\alpha, \underline{\underline{A}}_{\Delta\Delta}^\alpha\tag{15.2}$$

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

$$\underline{\underline{A}}_{II}^\alpha, \underline{\underline{A}}_{I\pi}^\alpha, \underline{\underline{A}}_{I\Delta}^\alpha, \underline{\underline{A}}_{\pi I}^\alpha, \underline{\underline{A}}_{\pi\pi}^\alpha, \underline{\underline{A}}_{\pi\Delta}^\alpha, \underline{\underline{A}}_{\Delta I}^\alpha, \underline{\underline{A}}_{\Delta\pi}^\alpha \text{ y } \underline{\underline{A}}_{\Delta\Delta}^\alpha\tag{15.3}$$

ello se puede hacer de dos formas, a saber:

- A partir de la matriz que es obtenida después de que el problema se ha discretizado -a partir de una discretización por el método de Elemento Finito, Volumen Finito o Diferencias Finitas- y para su aplicación no se requiere ninguna información acerca de la ecuación diferencial parcial de la cual se originó.

- A partir la discretización de la ecuación diferencial parcial generada localmente en cada subdominio por algún método de descomposición de dominio sin traslapes, como el usado para FETI-DP o BDDC.

En la sección (12.5) se derivó la forma de obtener las matrices locales a partir de la matriz $\underline{\underline{A}}^t : W \rightarrow W$ que es obtenida después de que el problema se ha discretizado y la cual es puesta en el espacio de vectores derivados. En la siguiente sección se da la forma de derivar cada una de las matrices locales a partir la discretización de la ecuación diferencial parcial generada localmente en cada subdominio; para que en las siguientes secciones, mediante el uso de dichas matrices se pueda implementar cualquiera de los métodos desarrollados.

15.1 Discretización de los Métodos Partiendo de la Formulación Local

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

$$\underline{\underline{A}}_{II}^\alpha, \underline{\underline{A}}_{I\pi}^\alpha, \underline{\underline{A}}_{I\Delta}^\alpha, \underline{\underline{A}}_{\pi I}^\alpha, \underline{\underline{A}}_{\pi\pi}^\alpha, \underline{\underline{A}}_{\pi\Delta}^\alpha, \underline{\underline{A}}_{\Delta I}^\alpha, \underline{\underline{A}}_{\Delta\pi}^\alpha \text{ y } \underline{\underline{A}}_{\Delta\Delta}^\alpha \quad (15.4)$$

para cada Ω_α con $\alpha = 1, \dots, E$. Una vez construida las matrices locales, se procede a definir al complemento de Schur local por subdominio -más detalles véase sección (19.7)-

$$\underline{\underline{S}}_\pi^\alpha = \underline{\underline{A}}_{\pi\pi}^\alpha - \underline{\underline{A}}_{\pi I}^\alpha \left(\underline{\underline{A}}_{II}^\alpha \right)^{-1} \underline{\underline{A}}_{I\pi}^\alpha \quad (15.5)$$

con él se define

$$\underline{\underline{A}}_{\text{III}}^\alpha = \begin{pmatrix} \underline{\underline{A}}_{II}^\alpha & \underline{\underline{A}}_{I\pi}^\alpha \\ \underline{\underline{A}}_{\pi I}^\alpha & \underline{\underline{A}}_{\pi\pi}^\alpha \end{pmatrix} \quad (15.6)$$

que a su vez define

$$\underline{\underline{S}}_\Delta^\alpha = \underline{\underline{A}}_{\Delta\Delta}^\alpha - \underline{\underline{A}}_{\Delta\text{III}}^\alpha \left(\underline{\underline{A}}_{\text{III}}^\alpha \right)^{-1} \underline{\underline{A}}_{\text{III}\Delta}^\alpha \quad (15.7)$$

recordando que

$$\left\{ \begin{array}{ll} \underline{\underline{A}}_{\text{III}} = \sum_{\alpha=1}^E \underline{\underline{A}}_{\text{III}}^{\alpha}, & \underline{\underline{A}}_{\text{II}\Delta} = \sum_{\alpha=1}^E \underline{\underline{A}}_{\text{II}\Delta}^{\alpha} \\ \underline{\underline{A}}_{\Delta\text{II}} = \sum_{\alpha=1}^E \underline{\underline{A}}_{\Delta\text{II}}^{\alpha}, & \underline{\underline{A}}_{\Delta\Delta} = \sum_{\alpha=1}^E \underline{\underline{A}}_{\Delta\Delta}^{\alpha} \end{array} \right. \quad (15.8)$$

$$\underline{\underline{S}} = \sum_{\alpha=1}^E \underline{\underline{S}}^{\alpha} \quad \text{y} \quad (\underline{\underline{S}})^{-1} = \sum_{\alpha=1}^E (\underline{\underline{S}}^{\alpha})^{-1} \quad (15.9)$$

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

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

$$\left(\underline{\underline{A}}_{\text{III}} \right)^{-1} = \sum_{\alpha=1}^E \left(\underline{\underline{A}}_{\text{III}}^{\alpha} \right)^{-1} \quad \text{y} \quad \left(\underline{\underline{A}} \right)^{-1} = \sum_{\alpha=1}^E \left(\underline{\underline{A}}^{\alpha} \right)^{-1} \quad (15.10)$$

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

15.2 Formulación Operacional de los Métodos DVS

Para la resolución de problemas de interés en Ciencias e Ingenierías donde es necesario resolver, por ejemplo el problema dado por la Ec.(12.74) -más detalles véase la sección (12.6)- que consiste en buscar una función $\underline{u} \in W_r$ que satisfaga

$$\underline{\underline{a}}\underline{\underline{A}}\underline{u}' = \underline{\underline{f}} \quad \text{y} \quad \underline{\underline{j}}\underline{u}' = 0 \quad (15.11)$$

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

$$\begin{pmatrix} u_{\text{II}} \\ u_{\Delta} \end{pmatrix} = \underline{u}' \quad \text{y} \quad \begin{pmatrix} f_{\text{II}} \\ f_{\Delta} \end{pmatrix} = \underline{\underline{f}} \quad (15.12)$$

que satisfaga -véase sección (13.3) del operador Steklov-Poincaré Ecs.(13.62 y 13.63)-

$$\left(\underline{\underline{L}} + \underline{\underline{a}}\underline{\underline{R}} - \underline{\underline{R}}^T \underline{\underline{j}} \right) \underline{u}' = \underline{\underline{f}}. \quad (15.13)$$

Entonces es necesario transformar el problema Ec.(15.13) en uno que

$$\underline{f}_{\Pi} = 0 \quad (15.14)$$

para ello, se introduce un vector auxiliar

$$\underline{u}_p = \underline{A}_{\Delta\Pi} \left(\underline{A}_{\Pi\Pi} \right)^{-1} \underline{f}_{\Pi} \quad (15.15)$$

en el cual $(\underline{u}_p)_{\Delta} = 0$, por lo tanto la Ec.(15.13) toma la forma

$$\left(\underline{aR} - \underline{R}^T \underline{j} \right) \underline{u}_{\Delta} = \underline{f}_{\Delta} - \underline{u}_p \quad (15.16)$$

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

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

$$\underline{f}_{\Delta_2} = \underline{a} \underline{f}_{\Delta}, \quad y \quad \underline{f}_{\Delta_1} = \underline{j} \underline{f}_{\Delta} = 0 \quad (15.17)$$

$$(\underline{u}_p)_{\Delta} = \underline{a} \underline{A}_{\Delta\Pi} \left(\underline{A}_{\Pi\Pi} \right)^{-1} \underline{f}_{\Pi} \quad (15.18)$$

de lo anterior, la expresión dada por la Ec.(15.16), se puede reescribir como

$$\left(\underline{aS} - \underline{S} \underline{j} \right) \underline{u}_{\Delta} = \underline{a} \underline{f}_{\Delta} - \underline{a} \underline{A}_{\Delta\Pi} \left(\underline{A}_{\Pi\Pi} \right)^{-1} \underline{f}_{\Pi} \quad (15.19)$$

donde

$$\underline{S} = \underline{A}_{\Delta\Delta} - \underline{A}_{\Delta\Pi} \left(\underline{A}_{\Pi\Pi} \right)^{-1} \underline{A}_{\Pi\Delta} \quad (15.20)$$

o más compactamente se escribe como

$$\left(\underline{aS} - \underline{S} \underline{j} \right) \underline{u}_{\Delta} = \underline{f}_{\Delta} - (\underline{u}_p)_{\Delta}. \quad (15.21)$$

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

1. **Formulación del Algoritmo del Complemento de Schur** (PRIMAL#1) -véase ecuación (13.4) en la sección (13.1.1)-:

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

$$\underline{a}S\underline{u}_\Delta = \underline{f}_\Delta - (\underline{u}_p)_\Delta \quad (15.22)$$

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

$$\underline{a}S\underline{u}_\Delta = \underline{f}_\Delta - \underline{a}A_{\Delta\Pi} \left(\underline{A}_{\text{III}} \right)^{-1} \underline{f}_\Pi. \quad (15.23)$$

2. **Formulación Dual del Problema Neumann-Neumann** (DUAL#1) -véase la ecuación (13.11) de la sección (13.1.2)-:

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

$$\underline{j}\underline{S}^{-1}\underline{\lambda}_\Delta = \underline{j}\underline{S}^{-1} \left(\underline{f}_\Delta - (\underline{u}_p)_\Delta \right) \quad (15.24)$$

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

$$\underline{j}\underline{S}^{-1}\underline{\lambda}_\Delta = \underline{j}\underline{S}^{-1} \left(\underline{f}_\Delta - \underline{a}A_{\Delta\Pi} \left(\underline{A}_{\text{III}} \right)^{-1} \underline{f}_\Pi \right). \quad (15.25)$$

3. **Formulación Primal del Problema Neumann-Neumann** (PRIMAL#2) -véase ecuación (13.21) en la sección (13.1.3)-:

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

$$\underline{S}^{-1}\underline{j}\underline{v}_\Delta = \underline{S}^{-1}\underline{j}\underline{S}^{-1} \left(\underline{f}_\Delta - (\underline{u}_p)_\Delta \right) \quad (15.26)$$

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

$$\underline{S}^{-1}\underline{j}\underline{v}_\Delta = \underline{S}^{-1}\underline{j}\underline{S}^{-1} \left(\underline{f}_\Delta - \underline{a}A_{\Delta\Pi} \left(\underline{A}_{\text{III}} \right)^{-1} \underline{f}_\Pi \right). \quad (15.27)$$

4. Formulación Dual del Problema Neumann-Neumann (DUAL#2)

-véase la ecuación (13.29) de la sección (13.1.4)-:

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

$$\underline{S}a\underline{\mu}_\Delta = \underline{S}a\underline{S}j\underline{S}^{-1} \left(\underline{f}_\Delta - (\underline{u}_p)_\Delta \right) \quad (15.28)$$

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

$$\underline{S}a\underline{\mu}_\Delta = \underline{S}a\underline{S}j\underline{S}^{-1} \left(\underline{f}_\Delta - \underline{a}A_{\Delta\Pi} \left(\underline{A}_{\text{III}} \right)^{-1} \underline{f}_\Pi \right). \quad (15.29)$$

5. Formulación Precondicionada del Algoritmo BDDC (PRIMAL#1)

-véase la ecuación (13.30) en la sección (13.2.1)-:

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

$$\underline{a}S^{-1}\underline{a}S\underline{u}_\Delta = \underline{a}S^{-1} \left(\underline{f}_\Delta - (\underline{u}_p)_\Delta \right) \quad (15.30)$$

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

$$\underline{a}S^{-1}\underline{a}S\underline{u}_\Delta = \underline{a}S^{-1} \left(\underline{f}_\Delta - \underline{a}A_{\Delta\Pi} \left(\underline{A}_{\text{III}} \right)^{-1} \underline{f}_\Pi \right). \quad (15.31)$$

6. Formulación Precondicionada del Algoritmo FETI-DP (DUAL#1)

-véase la ecuación (13.36) en la sección (13.2.2)-:

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

$$\underline{j}S\underline{j}S^{-1}\underline{\lambda}_\Delta = \underline{j}S\underline{j}S^{-1} \left(\underline{f}_\Delta - (\underline{u}_p)_\Delta \right) \quad (15.32)$$

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

$$\underline{j}S\underline{j}S^{-1}\underline{\lambda}_\Delta = \underline{j}S\underline{j}S^{-1} \left(\underline{f}_\Delta - \underline{a}A_{\Delta\Pi} \left(\underline{A}_{\text{III}} \right)^{-1} \underline{f}_\Pi \right) \quad (15.33)$$

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

$$\underline{u}_\Delta = \underline{a}S^{-1} \left(\underline{f}_\Delta - \underline{j}\underline{\lambda}_\Delta \right). \quad (15.34)$$

7. **Formulación Primal Precondicionada del Problema Neumann-Neumann DVS-PRIMAL** (PRIMAL#2) -véase la ecuación (13.43) en la sección (13.2.3)-:

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

$$\underline{S}^{-1} \underline{j} \underline{S} \underline{j} \underline{v}_\Delta = \underline{S}^{-1} \underline{j} \underline{S} \underline{j} \underline{S}^{-1} \left(\underline{f}_\Delta - (\underline{u}_p)_\Delta \right) \quad (15.35)$$

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

$$\underline{S}^{-1} \underline{j} \underline{S} \underline{j} \underline{v}_\Delta = \underline{S}^{-1} \underline{j} \underline{S} \underline{j} \underline{S}^{-1} \left(\underline{f}_\Delta - \underline{a} \underline{A}_{\Delta\Pi} \left(\underline{A}_{\text{III}} \right)^{-1} \underline{f}_\Pi \right) \quad (15.36)$$

donde una vez calculada \underline{v}_Δ entonces \underline{u}_Δ es obtenida mediante

$$\underline{u}_\Delta = \underline{a} \underline{S}^{-1} \left(\underline{f}_\Delta - \underline{j} \underline{S} \underline{v}_\Delta \right). \quad (15.37)$$

8. **Formulación Dual Precondicionada del Problema Neumann-Neumann DVS-DUAL** (DUAL#2) -véase la ecuación (13.50) en la sección (13.2.4)-:

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

$$\underline{S} \underline{a} \underline{S}^{-1} \underline{a} \underline{\mu}_\Delta = \underline{S} \underline{a} \underline{S}^{-1} \underline{a} \underline{S} \underline{j} \underline{S}^{-1} \left(\underline{f}_\Delta - (\underline{u}_p)_\Delta \right) \quad (15.38)$$

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

$$\underline{S} \underline{a} \underline{S}^{-1} \underline{a} \underline{\mu}_\Delta = \underline{S} \underline{a} \underline{S}^{-1} \underline{a} \underline{S} \underline{j} \underline{S}^{-1} \left(\underline{f}_\Delta - \underline{a} \underline{A}_{\Delta\Pi} \left(\underline{A}_{\text{III}} \right)^{-1} \underline{f}_\Pi \right) \quad (15.39)$$

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

$$\underline{u}_\Delta = \underline{a} \underline{S}^{-1} \left(\underline{f}_\Delta + \underline{\mu}_\Delta \right). \quad (15.40)$$

15.3 Implementación Numérica de DVS

La implementación numérica de por ejemplo, el algoritmo DVS-BDDC (PRIMAL#1) puesto en su forma operacional, Ec.(15.31) es

$$\underline{a} \underline{S}^{-1} \underline{a} \underline{S} \underline{u}_\Delta = \underline{a} \underline{S}^{-1} \left(\underline{f}_\Delta - \underline{a} \underline{A}_{\Delta\Pi} \left(\underline{A}_{\text{III}} \right)^{-1} \underline{f}_\Pi \right) \quad (15.41)$$

en la cual define el sistema lineal virtual a resolver

$$\underline{\underline{M}}u_{\Delta} = \underline{\underline{b}} \quad (15.42)$$

donde

$$\underline{\underline{M}} = \underline{\underline{a}}\underline{\underline{S}}^{-1}\underline{\underline{a}}\underline{\underline{S}} \quad \text{y} \quad \underline{\underline{b}} = \underline{\underline{a}}\underline{\underline{S}}^{-1} \left(\underline{\underline{f}}_{\Delta} - \underline{\underline{a}}\underline{\underline{A}}_{\Delta\Pi} \left(\underline{\underline{A}}_{\text{III}} \right)^{-1} \underline{\underline{f}}_{\Pi} \right) \quad (15.43)$$

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

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

15.3.1 Implementación para Matrices Simétricas

Suponiendo que $\underline{\underline{S}}$ es simétrica y definida positiva, la formulación $\underline{\underline{M}}u_{\Delta} = \underline{\underline{b}}$ puede ser implementada como el método de Gradiente Conjugado -véase sección (21.2.5)- usando el algoritmo dado en la Ec.(21.28) en el cual se usa el producto interior según corresponda:

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

$$\langle \underline{\underline{u}}, \underline{\underline{w}} \rangle = \underline{\underline{u}} \cdot \underline{\underline{w}} \quad (15.44)$$

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

$$\langle \underline{\underline{u}}, \underline{\underline{w}} \rangle = \underline{\underline{S}}\underline{\underline{u}} \cdot \underline{\underline{w}} \quad (15.45)$$

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

$$\underline{\underline{r}}^0 = \underline{\underline{a}}\underline{\underline{S}}^{-1} \left(\underline{\underline{f}}_{\Delta} - \underline{\underline{a}}\underline{\underline{A}}_{\Delta\Pi} \left(\underline{\underline{A}}_{\text{III}} \right)^{-1} \underline{\underline{f}}_{\Pi} \right), \quad \underline{\underline{p}}^0 = \underline{\underline{r}}^0 \quad (15.46)$$

y la parte iterativa¹³ queda como:

$$\begin{aligned}
 &\text{Para } n = 1, 2, \dots \{ \\
 &\quad \alpha^n = \frac{\langle \underline{p}^n, \underline{S}\underline{p}^n \rangle}{\langle \underline{p}^n, \underline{S}\underline{a}\underline{S}^{-1}\underline{a}\underline{S}\underline{p}^n \rangle} \\
 &\quad \underline{u}^{n+1} = \underline{u}^n + \alpha^n \underline{p}^n \\
 &\quad \underline{r}^{n+1} = \underline{r}^n - \alpha^n \underline{a}\underline{S}^{-1}\underline{a}\underline{S}\underline{p}^n \\
 &\quad < \text{Prueba de convergencia} > \\
 &\quad \beta^n = \frac{\langle \underline{r}^{n+1}, \underline{S}\underline{r}^{n+1} \rangle}{\langle \underline{r}^n, \underline{S}\underline{r}^n \rangle} \\
 &\quad \underline{p}^{n+1} = \underline{r}^{n+1} + \beta^n \underline{p}^n \\
 &\}
 \end{aligned} \tag{15.47}$$

15.3.2 Implementación para Matrices no Simétricas e Indefinidas

Suponiendo que \underline{S} es no simétrica o indefinida, la formulación $\underline{M}\underline{u}_\Delta = \underline{b}$ puede ser implementada como el método de GMRES -véase sección (21.2.7)- usando el algoritmo dado en la Ec.(15.31), en cual se inicia tomando \underline{u}^0 , una elección común es tomar a $\underline{u}^0 = 0$, si este es el caso, entonces

$$\underline{r}^0 = \underline{a}\underline{S}^{-1} \left(\underline{f}_\Delta - \underline{a}\underline{A}_{\Delta\Pi} \left(\underline{A}_{\Pi\Pi} \right)^{-1} \underline{f}_\Pi \right), \beta^0 = \|\underline{r}^0\|, \underline{v}^1 = \underline{r}^0/\beta^0 \tag{15.48}$$

y la parte iterativa queda como:

$$\begin{aligned}
 &\text{Para } n = 1, 2, \dots, \text{Mientras } \beta^n < \tau\beta^0 \{ \\
 &\quad \underline{w}_0^{n+1} = \underline{a}\underline{S}^{-1}\underline{a}\underline{S}\underline{v}^n \\
 &\quad \text{Para } l = 1 \text{ hasta } n \{ \\
 &\quad \quad h_{l,n} = \langle \underline{w}_l^{n+1}, \underline{v}^l \rangle \\
 &\quad \quad \underline{w}_{l+1}^{n+1} = \underline{w}_l^{n+1} - h_{l,n}\underline{v}^l \\
 &\quad \} \\
 &\quad h_{n+1,n} = \|\underline{w}_{n+1}^{n+1}\| \\
 &\quad \underline{v}^{n+1} = \underline{w}_{n+1}^{n+1}/h_{n+1,n} \\
 &\quad \text{Calcular } \underline{y}^n \text{ tal que } \beta^n = \left\| \beta^0 \underline{e}_1 - \underline{\hat{H}}_n \underline{y}^n \right\| \text{ es mínima} \\
 &\}
 \end{aligned} \tag{15.49}$$

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

$$\underline{r}^n = \underline{r}^0 - \underline{a}\underline{S}^{-1}\underline{a}\underline{S}\underline{V}_k \underline{y}^n = \underline{V}_{n+1} \left(\beta^0 \underline{e}_1 - \underline{\hat{H}}_n \underline{y}^n \right). \tag{15.50}$$

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

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

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

$$\underline{\underline{A}} = \begin{pmatrix} \underline{\underline{A}}_{\text{III}} & \underline{\underline{A}}_{\text{II}\Delta} \\ \underline{\underline{A}}_{\Delta\text{II}} & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix} = \begin{pmatrix} \underline{\underline{A}}_{\text{II}} & \underline{\underline{A}}_{\text{I}\pi} & \underline{\underline{A}}_{\text{I}\Delta} \\ \underline{\underline{A}}_{\pi\text{I}} & \underline{\underline{A}}_{\pi\pi} & \underline{\underline{A}}_{\pi\Delta} \\ \underline{\underline{A}}_{\Delta\text{I}} & \underline{\underline{A}}_{\Delta\pi} & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix} \quad (15.51)$$

dónde

$$\begin{aligned} \underline{\underline{A}}_{\text{III}} &= \begin{pmatrix} \underline{\underline{A}}_{\text{II}} & \underline{\underline{A}}_{\text{I}\pi} \\ \underline{\underline{A}}_{\pi\text{I}} & \underline{\underline{A}}_{\pi\pi} \end{pmatrix} & \underline{\underline{A}}_{\text{II}\Delta} &= \begin{pmatrix} \underline{\underline{A}}_{\text{I}\Delta} \\ \underline{\underline{A}}_{\pi\Delta} \end{pmatrix} \\ \underline{\underline{A}}_{\Delta\text{II}} &= \begin{pmatrix} \underline{\underline{A}}_{\Delta\text{I}} & \underline{\underline{A}}_{\Delta\pi} \end{pmatrix} \end{aligned} \quad (15.52)$$

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

$$\underline{\underline{S}} = \underline{\underline{A}}_{\Delta\Delta} - \underline{\underline{A}}_{\Delta\text{II}} \left(\underline{\underline{A}}_{\text{III}}\right)^{-1} \underline{\underline{A}}_{\text{II}\Delta} \quad (15.53)$$

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

$$\underline{\underline{S}}^{\alpha} = \underline{\underline{A}}_{\Delta\Delta}^{\alpha} - \underline{\underline{A}}_{\Delta\text{II}}^{\alpha} \left(\underline{\underline{A}}_{\text{III}}^{\alpha}\right)^{-1} \underline{\underline{A}}_{\text{II}\Delta}^{\alpha}. \quad (15.54)$$

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

$$\begin{pmatrix} \underline{\underline{w}}_{\text{I}} \\ \underline{\underline{w}}_{\pi} \end{pmatrix} = \begin{pmatrix} \underline{\underline{A}}_{\text{II}} & \underline{\underline{A}}_{\text{I}\pi} \\ \underline{\underline{A}}_{\pi\text{I}} & \underline{\underline{A}}_{\pi\pi} \end{pmatrix}^{-1} \begin{pmatrix} \underline{\underline{u}}_{\text{I}} \\ \underline{\underline{u}}_{\pi} \end{pmatrix} \quad (15.55)$$

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

$$\begin{pmatrix} \underline{\underline{A}}_{II} & \underline{\underline{A}}_{I\pi} \\ \underline{\underline{A}}_{\pi I} & \underline{\underline{A}}_{\pi\pi} \end{pmatrix} \begin{pmatrix} \underline{w}_I \\ \underline{w}_\pi \end{pmatrix} = \begin{pmatrix} \underline{u}_I \\ \underline{u}_\pi \end{pmatrix} \quad (15.56)$$

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

$$\left(\underline{\underline{A}}_{\pi\pi} - \underline{\underline{A}}_{\pi I} \left(\underline{\underline{A}}_{II} \right)^{-1} \underline{\underline{A}}_{I\pi} \right) \underline{w}_\pi \equiv \underline{\underline{S}}_\pi \underline{w}_\pi = \underline{u}_\pi - \underline{\underline{A}}_{\pi I} \left(\underline{\underline{A}}_{II} \right)^{-1} \underline{u}_I \quad (15.57)$$

mientras

$$\underline{w}_I = \left(\underline{\underline{A}}_{II} \right)^{-1} \left(\underline{u}_I - \underline{\underline{A}}_{I\pi} \underline{w}_\pi \right) \quad (15.58)$$

donde

$$\left(\underline{\underline{A}}_{II} \right)^{-1} = \sum_{i=1}^N \left(\underline{\underline{A}}_{II}^i \right)^{-1}. \quad (15.59)$$

Por último, para evaluar $\underline{\underline{S}}\underline{v}$ se aplica el procedimiento similar al detallado en la sección (19.2) y para evaluar $\underline{\underline{S}}^{-1}\underline{v}$ se aplica el procedimiento detallado en la sección (19.3).

De las evaluaciones indicadas en esta sección, una gran parte de ellas es posible realizar en paralelo, y como se mostrará más tarde, la granularidad¹⁴ paralela es gruesa, la cual es ideal para implementarse en equipos de cómputo paralelos como los Clusters, esto se demuestra mediante ejemplos en el capítulo de Análisis de Rendimiento (véase [36], [39] y [38]).

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

16 Implementación Computacional de DVS

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

El esquema Maestro-Eslavo parece ser un forma óptima¹⁵ de dividir la carga computacional requerida para solucionar un problema de descomposición de dominio sin traslapes, en el cual, uno o más subdominios son asignados a un nodo esclavo, tanto en su implementación secuencial -donde cada nodo esclavo es un objeto- como en su implementación paralela -donde cada nodo esclavo está asignado a un procesador-, en el cual el nodo maestro de forma síncrona controla las tareas que requiere el esquema DVS, las cuales son llevadas a cabo por los nodos esclavos, donde la comunicación sólo se da entre el nodo maestro y cada nodo esclavo -no existiendo comunicación entre los nodos esclavos-, optimizando así las comunicaciones.

16.1 Esquema Maestro-Eslavo como una Forma de Implementación

El esquema Maestro-Eslavo permite que en los nodos esclavos se definan uno o más subdominios -en los cuales se generen y manipulen las matrices locales de cada subdominio- y que el maestro controle las actividades necesarias para implementar cualquiera de los métodos desarrollados. En particular la implementación del método de resolución del sistema lineal virtual $\underline{Mu}_\Delta = \underline{b}$ esquematizados por los algoritmos descritos en la Ec.(15.47 ó 15.49), donde el nodo maestro controlará a cada uno de sus nodos esclavos mediante comunicaciones entre este y los esclavos, pero no entre esclavos, como se muestra en la figura.

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

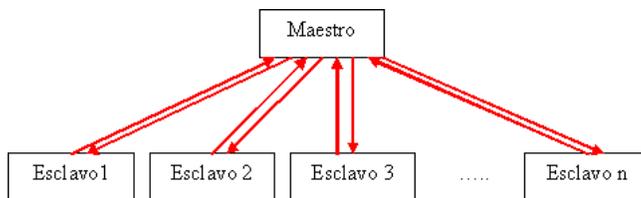


Figura 15: Esquema del Maestro-Esclavo

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

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

16.2 Análisis, Diseño y Programación Orientada a Objetos

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

usando la biblioteca de paso de mensajes MPI.

Dentro de las consideraciones básicas en el análisis orientado a objetos es que el código debería de correr tanto en equipos secuenciales como en paralelos, con un mínimo de cambios y que la interdependencia de la parte paralela no debería afectar la parte secuencial. Para que cualquier cambio en el código de los métodos desarrollados no requiera grandes cambios en el código paralelo. Esto se logra mediante la programación orientada a objetos haciendo uso de clases abstractas¹⁶ o contenedores.

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

Por ejemplo, en el uso de los métodos numéricos tanto directos como iterativos para resolver sistemas lineales en los cuales la matriz es real - existe como tal- o es virtual -está dispersa por los distintos subdominios- se creó una jerarquía de clases que implementa mediante herencia a la clase abstracta, la cual usan los algoritmos que requerían solucionar un sistema lineal, esta clase abstracta se llama Solvable -véase apéndice 21-. La jerarquía¹⁷ de clases mostrada en la figura (16) permite contener a cualquiera de los métodos numéricos de solución de sistemas lineales actuales y cualquier implementación futura y es independiente de si se usa para generar código secuencial o paralelo.

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

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

¹⁷Las jerarquías de clases de herencia mostradas en las figuras fueron generadas usando Doxygen documentation (véase [77]) a partir del código fuente en C++.

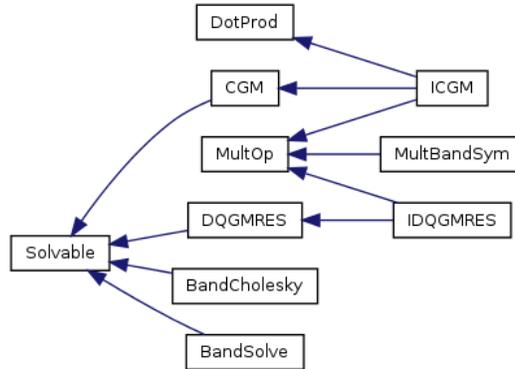


Figura 16: Jerarquía de clases para la implementación de los métodos de resolución de sistemas lineales.

16.2.1 Implementación Secuencial en C++

Usando la filosofía del manejo de clases abstractas desde el análisis y durante el diseño de la implementación computacional de los ocho métodos de descomposición de dominio en el espacio de vectores derivados, se pensó en usar una jerarquía de clases que especializarían a una clase abstracta llamada `DPMMethod`, la cual permite implementar uno o más de los métodos de descomposición de dominio desarrollados; y dada una ecuación o sistemas de ecuaciones diferenciales parciales, se usaría el método iterativo -Gradiente Conjugado o el método Residual Mínimo Generalizado o cualquier otro- dependiendo de que la matriz global virtual fueran simétrica o no simétrica; su jerarquía de clases se muestra en la figura (17).

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

Además de la flexibilidad anteriormente comentada, también se reutiliza la jerarquía de clases para la resolución de sistemas lineales, permitiendo que cualquier cambio o refinamiento a estas clases redunde en el desempeño global del sistema, permitiendo que en un futuro se agreguen y refinen métodos

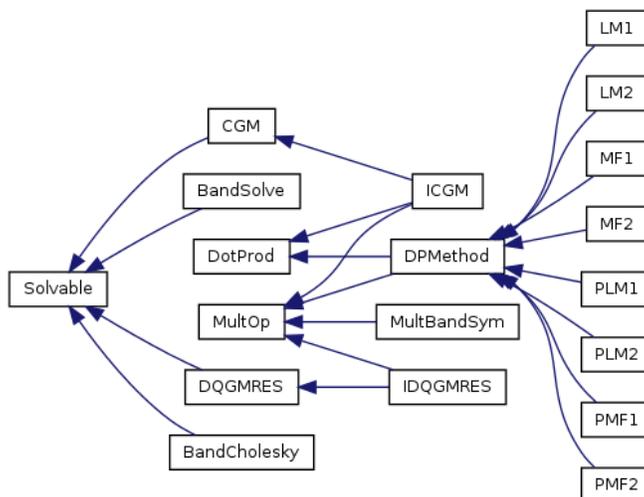


Figura 17: Jerarquía de clases para la implementación secuencial

que manejen con eficiencia la solución de los sistemas lineales asociados al método de descomposición de dominio DVS.

16.2.2 Implementación Paralela en C++ Usando MPI

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

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

En el modelo de paso de mensajes mediante MPI para equipos con uno o más Cores, los procesos se ejecutan en paralelo, teniendo direcciones de memoria separada para cada proceso, la comunicación ocurre cuando una

porción de la dirección de memoria de un proceso es copiada mediante el envío de un mensaje dentro de otro proceso en la memoria local mediante la recepción del mismo.

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

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

Send(dir, lg, td, dest, etiq, com)

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

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

Recv(dir, mlg, td, fuent, etiq, com, st)

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

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

MPI::Init	Inicializa al MPI
MPI::COMM_WORLD.Get_size	Busca el número de procesos existentes
MPI::COMM_WORLD.Get_rank	Busca el identificador del proceso
MPI::COMM_WORLD.Send	Envía un mensaje
MPI::COMM_WORLD.Recv	Recibe un mensaje
MPI::Finalize	Termina al MPI

La estructura básica del programa bajo el esquema Maestro-Esclavo codificada en C++ y usando MPI es:

```
main(int argc, char *argv[])
{
    MPI::Init(argc,argv);
    ME_id = MPI::COMM_WORLD.Get_rank();
    MP_np = MPI::COMM_WORLD.Get_size();
    if (ME_id == 0) {
        // Operaciones del Maestro
    } else {
        // Operaciones del esclavo con identificador ME_id
    }
    MPI::Finalize();
}
```

En este único programa se deberá de codificar todas las tareas necesarias para el nodo maestro y cada uno de los nodos esclavos, así como las formas de intercomunicación entre ellos usando como distintivo de los distintos procesos a la variable *ME_id* (véase [15] y [16]).

La jerarquía de clases del esquema Maestro-Esclavo en su implementación paralela permite repartir la carga de varias maneras en uno o más Cores. Reutilizando toda la jerarquía de clases de la implementación secuencial de los algoritmos DVS y sólo es necesario agregar clase que especializa algunos comportamientos que requieren hacer uso de las comunicaciones, mediante la biblioteca de paso de mensajes MPI. La jerarquía de clases es mostrada en la figura siguiente:

La reutilización de toda la jerarquía de clases generada para la implementación secuencial permite que el código paralelo soporte una gran cantidad de cambios sin afectación a la implementación paralela, teniendo así, un código robusto, flexible, modular y de fácil mantenimiento (véase [17]).

16.3 Alcances y Limitaciones del Esquema Maestro-Esclavo

El esquema Maestro-Esclavo es eficiente cuando se tiene una carga casi homogénea en cada nodo esclavo y se manejan una cantidad moderada de ellos. Un factor limitante en el esquema Maestro-Esclavo, es que el nodo maestro deberá de atender todas las peticiones hechas por todos y cada uno de los nodos esclavos, esto toma especial relevancia cuando todos o casi todos los nodos esclavos compiten por ser atendidos por el nodo maestro.

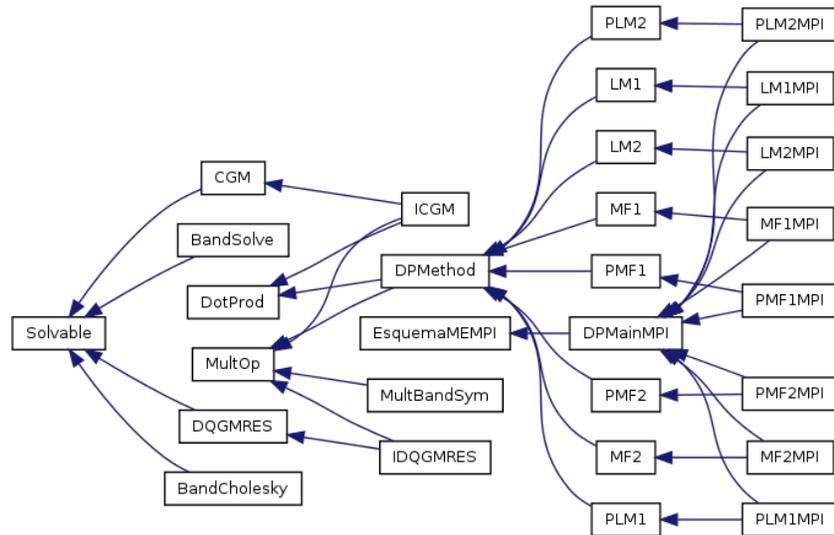


Figura 18: Jerarquía de clases para la implementación paralela rehusando toda la jerarquía de la implementación secuencial y de resolución de sistemas lineales

Una opción para optimizar el esquema Maestro-Esclavo es contar con un nodo maestro lo suficientemente poderoso para atender simultáneamente la mayor cantidad de las tareas síncronas del método de descomposición de dominio en el menor tiempo posible. Pero los factores limitantes del esquema Maestro-Esclavo son de tres tipos, a saber:

1. Los inherentes al método de descomposición de dominio.
2. Los inherentes al propio esquema Maestro-Esclavo.
3. Los inherentes al equipo de cómputo en paralelo en el que se ejecute el programa.

En el primer caso, en cuanto a los inherentes al método de descomposición de dominio destacan:

- El método de descomposición de dominio es síncrono, es decir, si un nodo esclavo acaba la tarea asignada y avisa al nodo maestro, este no podrá asignarle otra tarea hasta que todos los nodos esclavos concluyan

la suya, y se realicen las operaciones necesarias para asignar las nuevas tareas a los nodos esclavos.

- El nodo maestro sólo realiza tareas de control y sincronización pero no conoce o realiza cálculos relacionados con los sistemas lineales locales a cada uno de los subdominios que están asignados a los nodos esclavos.
- Por lo anterior, el esquema Maestro-Eslavo no es eficiente si sólo se usan dos procesos o Cores -uno para el nodo maestro y otro para el nodo esclavo-, por otro lado, cuando se realiza el análisis de rendimiento en P Cores, hay que tomar en cuenta que los únicos nodos que manipulan los sistemas lineales asociados al método de descomposición de dominio son los esclavos $(P - 1)$ y el nodo maestro sólo realiza el control y sincronización de las tareas de los métodos DVS.

En el segundo caso, en cuanto a los inherentes al propio esquema Maestro-Eslavo destacan:

- El nodo maestro deberá distribuir las tareas a los nodos esclavos acorde al número de subdominios existentes en la descomposición y la malla fina de cada subdominio, de tal forma que cada nodo esclavo tenga una carga computacional equivalente a los demás nodos esclavos.
- En el caso de una carga homogénea en cada subdominio, si se usan P Cores en el equipo paralelo y la descomposición del dominio tiene E subdominios, tal que $(P - 1) \nmid E$, esa descomposición de dominio no es adecuada para trabajar en dicha cantidad de Cores. En este caso, el número de procesadores P que se usen para tener buen balance de cargas es conocido a priori cuando el dominio Ω se descompone en $n \times m$ - $n \times m \times o$ - subdominios homogéneos, entonces se generarán $E = n * m$ - $E = n * m * o$ - subdominios Ω_α , teniendo un buen balanceo de cargas si $(P - 1) \mid E$.
- Pese al buen balanceo de la carga en los nodos esclavos, es común que, un gran número de nodos esclavos envíen simultáneamente datos al nodo maestro saturando su canal de comunicación; y este en algún momento tendrá que tratar atender las múltiples comunicaciones, degradando su rendimiento al aumentar el número de nodos esclavos involucrados en la descomposición.

En el caso de generar desbalance de la carga en los nodos esclavos o una saturación de comunicaciones en el nodo maestro, se propicia a que algunos procesadores terminen antes que otros, generando tiempos muertos de ejecución en dichos Cores; propiciando una notoria degradación en la eficiencia global en el procesamiento, es por esto que, en algunos casos al aumentar el número de procesadores no se aprecia una disminución sustancial del tiempo de ejecución y en casos extremos puede ocasionar un aumento en el tiempo.

En el tercer caso, en cuanto a los inherentes al equipo de cómputo en paralelo en el que se ejecute el programa destacan:

- El programa se diseñó para correr en cualquier cantidad de procesos o Cores y no hay límite establecido en cuanto al número de subdominios que soporta el programa, pero el equipo en el que se ejecute tiene un número predeterminado de Cores y cada uno de ellos tiene asignado una cantidad limitada de RAM, es por ello que, las dimensiones del problema que es posible correr en un equipo paralelo dado está determinado por estas limitantes.
- En los equipos paralelos, el cuello de botella en cuanto a la eficiencia global de la ejecución, lo presentan las comunicaciones, entre más comunicaciones necesite el programa, es imperante el contar con una infraestructura que permita la mejor velocidad de comunicaciones entre el nodo maestro y los nodos esclavos; además de que esta cuente con la menor latencia posible en las comunicaciones. Por otro lado, el acceso al disco duro es mínimo y no representa un costo significativo en las comunicaciones totales de la ejecución.

Para ejemplificar lo discutido anteriormente, se considera como modelo matemático el problema de valor en la frontera (BVP) asociado con la ecuación de Poisson, con condiciones de frontera Dirichlet, definido en Ω como:

$$\begin{aligned} -\nabla^2 u &= f_\Omega \text{ en } \Omega \\ u &= g_{\partial\Omega} \text{ en } \partial\Omega \end{aligned} \tag{16.1}$$

este ejemplo, gobierna los modelos de muchos sistemas de la ingeniería y de la ciencia, entre ellos el flujo de agua subterránea a través de un acuífero

isotrópico, homogéneo bajo condiciones de equilibrio y es muy usado en múltiples ramas de la física, por ejemplo, gobierna la ecuación de la conducción de calor en un sólido bajo condiciones de equilibrio.

En particular se considera el problema con Ω definido en:

$$\Omega = [-1, -1] \times [1, 1] \tag{16.2}$$

donde

$$f_{\Omega} = 2n^2\pi^2 \sin(n\pi x) * \sin(n\pi y) \quad \text{y} \quad g_{\partial\Omega} = 0 \tag{16.3}$$

cuya solución es

$$u(x, y) = \sin(n\pi x) * \sin(n\pi y) \tag{16.4}$$

Por ejemplo para $n = 4$, la solución es $u(x, y) = \sin(4\pi x) * \sin(4\pi y)$, cuya gráfica se muestra a continuación:

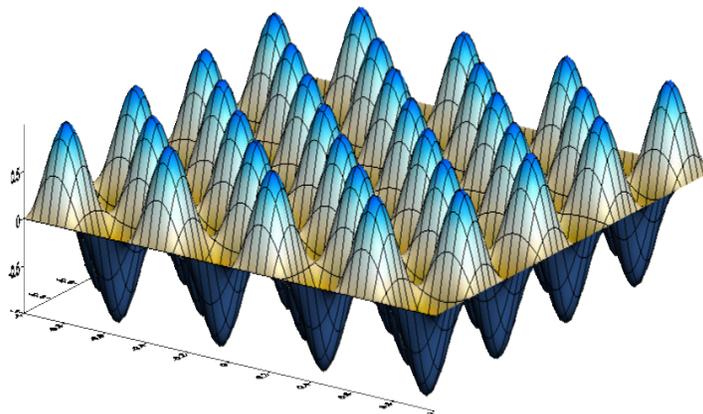


Figura 19: Solución a la ecuación de Poisson para $n=4$.

Para las pruebas de rendimiento¹⁸ en las cuales se evalúa el desempeño de los programas realizados se usa $n = 100$, pero es posible hacerlo con $n \in \mathbb{N}$ grande.

¹⁸En todos los ejemplos del presente trabajo no se realiza una análisis de comunicación de forma independiente al tiempo de cálculo, ya que los Clusters a los que se obtuvo acceso carecen de herramientas que permitan realizar dicho análisis, pero sí se realizaron algunas pruebas con XMPI (véase [78]) y Vampir (véase [79]) para algunos ejemplos representativos en los cuales se muestra que se tiene granularidad gruesa (véase [39]).

Supóngase que se desea resolver el dominio Ω usando 1024×1024 nodos -1,048,576 grados de libertad- mediante el algoritmo preconditionado PRIMAL#2, de manera inmediata surgen las siguientes preguntas: ¿cuáles son las posibles descomposiciones posibles? y ¿en cuántos procesadores se pueden resolver cada descomposición?. Para este ejemplo en particular, sin hacer la tabla exhaustiva, se tiene:

Partición	Subdominios	Procesadores
2x2 y 512x512	4	2,3,5
4x4 y 256x256	16	2,3,5,9,17
8x8 y 128x128	64	2,3,5,9,17,33,65
16x16 y 64x64	256	2,3,5,9,17,33,65,129,257
32x32 y 32x32	1024	2,3,5,9,17,33,65,129,...,1025
64x64 y 16x16	4096	2,3,5,9,17,33,65,129,...,4097
128x128 y 8x8	16384	2,3,5,9,17,33,65,129,...,16385
256x256 y 4x4	65536	2,3,5,9,17,33,65,129,...,65537
512x512 y 2x2	262144	2,3,5,9,17,33,65,129,...,262145

De esta tabla es posible seleccionar las descomposiciones que se adecuen a las necesidades particulares del equipo paralelo con que se cuente, para evaluar el tiempo de ejecución de este ejemplo se usó la PC Antipolis Intel Xeon a 2.33 GHz de 64 bits con 8 Cores y 32 GB de RAM, obteniendo los siguientes resultados para una tolerancia de 10^{-6} usando norma infinita en todos los casos (tiempo en segundos):

	1 Core	2 Cores	3 Cores	4 Cores	5 Cores	6 Cores	7 Cores	8 Cores
Partición	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo
2x2 y 512x512	16465	10659	7207	7105	4641			
4x4 y 256x256	2251	5063	2252	2103	1643	1233	1068	947
8x8 y 128x128	855	885	482	395	314	311	283	272
16x16 y 64x64	321	348	190	149	121	125	118	117
32x32 y 32x32	26	39	26	24	23	21	21	21
64x64 y 16x16	205	595	485	477	481	461	469	469
128x128 y 8x8	1026	5453	5352	5431	5633	5843	5843	5903
256x256 y 4x4	8544	26167	25892	25902	25939	25950	25969	26003
512x512 y 2x2	34845	64230	63293	63308	63389	63475	63502	63693

De estos resultados, se desprende que:

1. Dependiendo del tamaño de la malla gruesa -número de subdominios a trabajar- y de la malla fina, es siempre posible encontrar una descomposición de dominio -32×32 y 32×32 - en que el tiempo de cálculo sea mínimo:
 - Al usar un solo Core (programa secuencial 26 seg.).
 - Al usar múltiples Cores interconectados mediante MPI (6 Cores en 21 seg.).
2. Es notorio el efecto que genera el mal balance de carga¹⁹, el cual se refleja en que no disminuye el tiempo de ejecución al aumentar el número de procesadores y en algunos casos el tiempo aumenta conforme se agregan más Cores.

En contraste con los 110 segundos en que se resolvió el mismo problema usando los métodos de Elemento Finito y Diferencias Finitas, usando en ambos casos Factorización Cholesky para resolver el sistema lineal asociado.

16.4 Afectación del Rendimiento al Refinar la Descomposición

Una parte fundamental al trabajar con problemas reales usando una descomposición fina es conocer a priori que factores afectan el rendimiento de la aplicación ante las posibles elecciones en la descomposición de dominio, la afectación se da por:

1. En el caso de contar con un gran número de subdominios que estén asignados a distintos nodos esclavos, la afectación se da por la saturación del nodo maestro con una gran cantidad de comunicaciones simultáneas por parte de los nodos esclavos que el nodo maestro deberá de atender y la velocidad de comunicación del canal usado para ello. Esto es especialmente importante en la implementación paralela en la cual la interconexión del equipo paralelo se hace mediante un canal de comunicación lento u ocupado por otros procesos.

¹⁹Por ejemplo, al usar una descomposición gruesa de $64 \times 64 = 4096$ subdominios repartidos en 3, 5, 6, 7 nodos esclavos.

2. En el caso de realizar una descomposición muy fina en cada subdominio, la afectación del rendimiento se da al aumentar el número de nodos involucrados en el complemento de Schur local $\underline{\underline{S}}^i$, ya que esto significa, por un lado generar matrices locales más grandes

$$\underline{\underline{A}}_{II}^\alpha, \underline{\underline{A}}_{I\pi}^\alpha, \underline{\underline{A}}_{I\Delta}^\alpha, \underline{\underline{A}}_{\pi I}^\alpha, \underline{\underline{A}}_{\pi\pi}^\alpha, \underline{\underline{A}}_{\pi\Delta}^\alpha, \underline{\underline{A}}_{\Delta I}^\alpha, \underline{\underline{A}}_{\Delta\pi}^\alpha \text{ y } \underline{\underline{A}}_{\Delta\Delta}^\alpha \quad (16.5)$$

además de resolver el sistema $y = \left(\underline{\underline{A}}_{II}^i\right)^{-1} x$ de alguna forma. Si el número de nodos interiores en el subdominio es grande entonces solucionar el complemento de Schur local será costoso computacionalmente.

Para el primer caso, el uso de algunos cientos o miles de subdominios no afectan de manera considerable el desempeño del Esquema Maestro-Eslavo si la red es relativamente rápida (de un Gigabit por segundo o más), y como los avances en las comunicaciones son vertiginosos, en un corto tiempo se tendrá acceso a redes de mayor velocidad reduciendo el efecto de manipular un gran número de subdominios simultáneamente.

Para el segundo caso, al resolver el complemento de Schur local, se puede emplear diversos métodos de solución, la selección del método más adecuado al problema en particular depende por un lado de las capacidades computacionales del equipo donde se implemente la ejecución y las características propias de los sistemas lineales asociados al problema. Así, para solucionar el sistema $y = \left(\underline{\underline{A}}_{II}^i\right)^{-1} x$ correspondiente al complemento de Schur local $\underline{\underline{S}}^i$ se puede usar por ejemplo: Factorización LU, Factorización Cholesky, Gradiente Conjugado o alguna variante de GMRES, pero deberá de usarse aquel método que proporcione la mayor velocidad en el cálculo o que consuma la menor cantidad de memoria -ambas condicionantes son mutuamente excluyentes-, por ello la decisión de qué método usar deberá de tomarse al momento de tener que resolver un problema particular en un equipo dado y básicamente el condicionante es el tamaño de la matriz $\underline{\underline{A}}_{II}^i$ versus el método numérico usado para resolver el sistema lineal asociado -todas esas opciones están implementadas en el código y pueden seleccionarse conforme sean requeridos en la ejecución, mediante directivas de compilación-

Por lo visto en el ejemplo anterior, si el problema involucra una gran cantidad de nodos interiores y el equipo -secuencial o paralelo- en el que se implantará la ejecución del programa tiene una cantidad de memoria reducida, es recomendable en los procesos locales a los subdominios usar métodos iterativos -Gradiente Conjugado o alguna variante de GMRES-, estos

consume una cantidad de memoria pequeña comparada con los métodos directos -Factorización LU o Cholesky- pero requieren una gran cantidad de iteraciones para obtener la misma precisión que los directos.

Hay que tomar en cuenta que al aumentar el número de subdominios en una descomposición particular, se garantiza que las matrices a generar y calcular sean cada vez más pequeñas y fáciles de manejar. Pero hay un límite al aumento del número de subdominio y disminución del tamaño de las matrices a generar por subdominio; y esto se refleja en una pérdida de eficiencia en el tiempo de ejecución, esto es generado por la gran cantidad de subdominios que es necesario crear y manejar por el nodo maestro, incrementando sustancialmente las comunicaciones y por otro lado, cada subdominio manejará cada vez matrices más pequeñas con el consecuente aumento de los tiempos muertos, al invertir mucho más tiempo en comunicaciones que en cálculos.

Para mitigar los factores limitantes inherente al propio esquema Maestro-Escavo, es posible implementar algunas operaciones del nodo maestro en paralelo, usando uno o más Cores distintos a los asignados a los nodos esclavos. Para la parte inherente al método de descomposición de dominio, la parte medular la da el balanceo de cargas. Es decir, cada nodo esclavo debe tener una carga de trabajo equivalente al resto de los nodos.

Tomando en cuenta lo discutido, para un problema particular y la descomposición del dominio Ω en la implementación paralela, hay que tomar en cuenta lo siguiente:

- Buscar que la descomposición de malla gruesa y su asociada malla fina, en la que cada nodo esclavo -asociado a un procesador- tenga una carga casi homogénea con respecto a los demás nodos esclavos, .i.e. buscar que en su conjunto, todos los subdominios Ω_α de la malla gruesa y su descomposición fina de cada uno de ellos, que estén asignados a cada nodo esclavo sean computacionalmente equivalentes.
- Elegir el método numérico local a cada subdominio para garantizar el uso de la menor cantidad de memoria posible y/o la mayor velocidad de ejecución versus la precisión global esperada del método de descomposición de dominio.
- Elegir de las distintas descomposiciones balanceadas del dominio Ω y las diferentes opciones de los métodos numéricos usados localmente en cada subdominio, aquella que presente el mejor rendimiento computa-

cional acorde al equipo paralelo en el cual se implemente la solución del problema.

Nótese que el esquema Maestro-Esclavo paralelo lanza P procesos -uno para el nodo maestro y $P-1$ para los nodos esclavos-, estos en principio corren en un solo procesador pero pueden ser lanzados en múltiples procesadores usando una directiva de ejecución, de esta manera es posible que en una sola máquina se programe, depure y sea puesto a punto el código usando mallas relativamente pequeñas -del orden de miles o millones de nodos- y cuando este listo para producción se puede mandar a cualquier equipo paralelo sin cambio alguno en el código.

16.5 Opciones para Soportar una Descomposición Fina del Dominio

Supóngase ahora que se necesita resolver el problema de una descomposición fina del dominio Ω , sin pérdida de generalidad, se puede suponer por ejemplo, que se usa una malla de 8192×8192 nodos, este tipo de problemas es común y surgen cotidianamente en la resolución de sistemas reales y las opciones para implantarlo en un equipo paralelo son viables, existen y son actualmente usadas. Aquí las opciones de partición del dominio son muchas y variadas, y la variante seleccionada dependerá fuertemente de las características del equipo de cómputo paralelo del que se disponga. Si se supone que una descomposición de 100×100 nodos en un subdominio consume 1 GB de RAM y que el consumo de memoria crece linealmente con el número de nodos, entonces algunas posibles descomposiciones son:

Procesadores	Descomposición	Nodos Subdominio	RAM Mínimo
5	2×2 y 4096×4096	4096×4096	≈ 40.0 GB
257	16×16 y 512×512	512×512	≈ 5.0 GB
1025	32×32 y 256×256	256×256	≈ 2.5 GB
4097	64×64 y 128×128	128×128	≈ 1.2 GB

Nótese que para las primeras particiones, el consumo de RAM es excesivo y en las últimas particiones la cantidad de procesadores en paralelo necesarios es grande -pero ya de uso común en nuestros días-. Como en general, contar con equipos paralelos de ese tamaño es en extremo difícil, ¿es posible resolver

este tipo de problemas con una cantidad de procesadores fijo menor al sugerido y donde cada uno de ellos tiene solo memoria suficiente para soportar uno o más subdominios?, la respuesta es sí.

Primero, nótese que al considerar una descomposición fina del tipo 64×64 y 128×128 se requiere aproximadamente 1.2 GB de RAM por Core, si además se supone que sólo se tienen unos cuantos procesadores con poca memoria -por ejemplo 2 GB-, entonces no es posible tener en memoria de manera conjunta a las matrices generadas por el método.

Una de las grandes ventajas de los métodos de descomposición de dominio es que los subdominios son en principio independientes entre si y que sólo están acoplados a través de la solución en la interfase de los subdominios que es desconocida.

Como sólo se requiere tener en memoria la información de la frontera interior, es posible bajar a disco duro todas las matrices y datos complementarios generados en cada subdominio -que consumen el 99% de la memoria del objeto *RectSub*-, que no se requieran en ese instante para la operación del esquema Maestro-Eslavo.

Recuperando del disco duro solamente los datos del subdominio a usarse en ese momento -ya que el proceso realizado por el nodo maestro es secuencial- y manteniéndolos en memoria por el tiempo mínimo necesario. Así, es posible resolver un problema de una descomposición fina, usando una cantidad de procesadores fija y con una cantidad de memoria reducida por procesador.

En un caso extremo, la implementación para resolver un dominio Ω descompuesto en un número de nodos grande es posible implementarla usando sólo dos procesos en un procesador, uno para el proceso maestro y otro para el proceso esclavo, en donde el proceso esclavo construiría las matrices necesarias por cada subdominio y las guardaría en disco duro, recuperándolas conforme el proceso del nodo maestro lo requiera. Nótese que la descomposición del dominio Ω estará sujeta a que cada subdominio Ω_i sea soportado en memoria conjuntamente con los procesos Maestro y Esclavo.

De esta forma es posible resolver un problema de gran envergadura usando recursos computacionales limitados, sacrificando velocidad de procesamiento en aras de poder resolver el problema. Está es una de las grandes ventajas de los métodos de descomposición de dominio con respecto a los otros métodos de discretización tipo Diferencias Finitas y Elemento Finito.

El ejemplo anterior da una buena idea de las limitantes que existen en la resolución de problemas con dominios que tienen una descomposición fina y

pone de manifiesto las características mínimas necesarias del equipo paralelo para soportar dicha implantación.

16.6 Otras Opciones de Paralelización

En la actualidad, casi todos los equipos de cómputo usados en estaciones de trabajo y Clusters cuentan con dos o más Cores, en ellos siempre es posible usar MPI para intercambiar mensajes entre procesos corriendo en el mismo equipo de cómputo, pero no es un proceso tan eficiente como se puede querer. En estas arquitecturas llamadas de memoria compartida es mejor usar OpenMP o cualquiera de sus variantes para trabajar en paralelo. Por otro lado es ya común contar con las cada vez más omnipresentes tarjetas NVIDIA, con los cada vez más numerosos Cores CUDA -que una sola tarjeta NVIDIA TESLA puede tener del orden de cientos de ellos- y que en un futuro serán cada vez más numerosos.

Para lograr obtener la mayor eficiencia posible de estos tres niveles de paralelización, se están implementando procesos híbridos (véase [75] y [76]), en donde la intercomunicación de equipos con memoria compartida se realiza mediante MPI y la intercomunicación entre Cores que comparten la misma memoria se realiza con OpenMP, además las operaciones matriciales se le encargan a los numerosos Cores CUDA de las tarjetas NVIDIA.

Los métodos de descomposición de dominio sin traslape y en particular el esquema DVS con sus ocho algoritmos, pueden hacer uso de esta forma integradora de paralelismo. Para ello, la interconexión de equipos de memoria compartida se realizaría mediante MPI y en cada equipo de memoria compartida se manipularían uno o más subdominios mediante OpenMP -ya que cada subdominio es independiente de los demás- y la manipulación de matrices y operaciones entre matrices y vectores que requiere cada subdominio se realizarían en las tarjetas NVIDIA mediante los numerosos Cores CUDA sin salir a la RAM de la computadora.

Para integrar esta forma de paralelismo en los códigos, es necesario hacer cambios mínimos²⁰ al mismo, ya que sólo es necesario reimplementar los comportamientos locales que requieren otro tipo de paralelismo, ya que la

²⁰Ya se tiene una versión operacional del código en los cuales se han realizado algunas pruebas de rendimiento, pero los resultados son limitados por la complejidad de la programación de las tarjetas NVIDIA y la falta de herramientas y bibliotecas de código abierto que optimicen y depuren las implementaciones.

jerarquía de clases del código desarrollado permite especializar los comportamientos que implementan las comunicaciones, esto queda de manifiesto al reutilizar toda la jerarquía de clases de la implementación secuencial en la implementación paralela como se aprecia en la figura (18).

Además, el esquema Maestro-Esclavo sólo requiere enviar un vector a cada subdominio en cada paso de la iteración del sistema lineal virtual -mediante el paso de mensajes usando MPI- el cual se coloca en la RAM de la memoria compartida, después este es copiado²¹ a la RAM de la tarjeta NVIDIA según el subdominio que se este trabajando -se controla usando OpenMP-, aquí los múltiples Cores CUDA sin salir de su RAM local efectuarían las operaciones de multiplicación de matriz vector necesarias para regresar un único vector a la RAM de la memoria compartida y de ahí se enviaría por MPI al nodo maestro, concluyendo la iteración.

Permitiendo así, tener una creciente eficiencia de paralelización que optimizan en gran medida los recursos computacionales, ya que todas las matrices y vectores se generarían en la RAM de la tarjeta NVIDIA, véase figura (20).

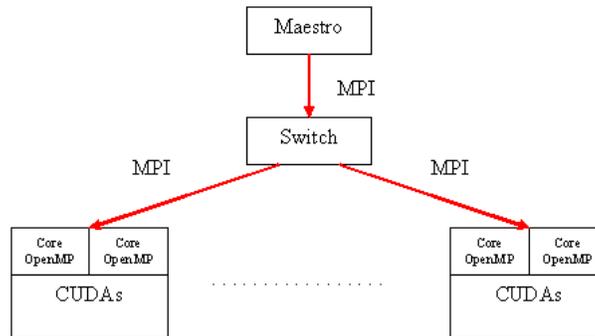


Figura 20: La intercomunicación de equipos con memoria compartida se realiza mediante MPI y la intercomunicación entre Cores que comparten la misma memoria se realiza con OpenMP, además las operaciones matriciales se le encargan a los numerosos Cores CUDA de las tarjetas NVIDIA.

²¹En tránsito de datos entre la RAM de la computadora y la RAM de los CUDA's, no es tan rápido como se requiere. Esto genera una baja de rendimiento considerable, que en ciertos problemas como los no lineales y con coeficientes variables es notorio.

De esta manera es posible adaptar el código para todos y cada uno de los métodos desarrollados, de forma tal que sea reutilizable y que pueda usarse en problemas en los que el número de grados de libertad sea grande, permitiendo hacer uso de equipos de cómputo cada vez más asequibles y de menor costo, pero con una creciente eficiencia computacional que podrán competir en un futuro con los grandes equipos de cómputo de alto desempeño.

17 Análisis y Discusión de Resultados

La uniformidad de las fórmulas presentadas en la sección (15.2) para los métodos de descomposición de dominio en el espacio de vectores derivados (DVS), nos han permitido implementar de forma eficiente dichos algoritmos en múltiples equipos secuenciales y paralelos, aprovechando el hecho que los algoritmos derivados son capaces de obtener *la solución global por la resolución de problemas locales exclusivamente*.

El grupo de ocho algoritmos desarrollados -de los cuales cuatro son preconditionados- a los que nos referiremos como los *algoritmos DVS* (véase [36], [38]), operan exclusivamente sobre los nodos primales y duales en la frontera interior y estos se implementan eficientemente mediante el uso de métodos iterativos -CGM para el caso simétrico o GMRES para el caso no simétrico- para resolver el sistema algebraico virtual asociado.

En esta sección, se mostrará -mediante los resultados de algunos experimentos numéricos conspicuos en Ciencias de la Tierra e Ingeniería- la eficiencia del esquema DVS (véase [39]), para ello; primero, se muestra el rendimiento en problemas simétrico y no simétricos -en dos y tres dimensiones-; segundo, se muestra el rendimiento en problemas indefinidos; tercero, se muestra el rendimiento en problemas de Advección-Difusión; después, se muestra el análisis de rendimiento en equipos paralelos hasta con 1024 Cores y por último se muestra lo que se consideran como criterios integrales para evaluar métodos de descomposición de dominio sin traslape y en especial al esquema DVS.

Para los experimentos numéricos reportados en esta sección, sólo se muestran los resultados de los cuatro métodos preconditionados del esquema DVS; en donde el dominio Ω fue discretizado usando una malla estructurada uniforme. Y en todos los casos, se tomaron como nodos primales a los vértices de los subdominios de la partición gruesa en dos dimensiones y a las aristas de los subdominios de la partición gruesa para tres dimensiones. Además, la tolerancia usada para concluir los métodos iterativos de resolución de sistemas lineal virtual asociado es en norma infinita.

17.1 Análisis de Rendimiento para Problemas Simétricos y no Simétricos

Para realizar el análisis de rendimiento, se usa la ecuación elíptica

$$-a\nabla^2 \underline{u} + \underline{b} \cdot \nabla \underline{u} + c\underline{u} = f \quad (17.1)$$

con condiciones de frontera Dirichlet cero, donde $a, c > 0$ son constantes, mientras que \underline{b} es un vector constante de dimensión n . El dominio $\Omega \subset \mathbb{R}^n$ fue tomado con $n = 2, 3$ donde Ω es el cuadrado o cubo unitario según corresponda.

Las matrices generadas fueron obtenidas por discretización local en ambos casos -en dos y tres dimensiones- del problema con valores en la frontera descrito anteriormente, donde $a = 1$. La elección $\underline{b} = (1, 1)$ y $\underline{b} = (1, 1, 1)$ con $c = 0$ generan matrices no simétricas, escogiendo $c = 1$ y $\underline{b} = 0$ se obtienen matrices simétricas las cuales son usadas con propósitos de comparación. En todos los ejemplos se usa una tolerancia de $1e - 6$.

Problemas en Dos Dimensiones En la primera tabla se muestra la descomposición usada, los grados de libertad asociados al sistema y el número de vértices primales usados:

Ejemplo	Partición	Subdominios	Grados Libertad	Primales
1	2×2 y 2×2	4	9	1
2	4×4 y 4×4	16	225	9
3	6×6 y 6×6	36	1225	25
4	8×8 y 8×8	64	3969	49
5	10×10 y 10×10	100	9801	81
6	12×12 y 12×12	144	20449	121
7	14×14 y 14×14	196	38025	169
8	16×16 y 16×16	256	65025	225
9	18×18 y 18×18	324	104329	289
10	20×20 y 20×20	400	159201	361
11	22×22 y 22×22	484	233289	441
12	24×24 y 24×24	576	330625	529
13	26×26 y 26×26	676	455625	625
14	28×29 y 28×28	784	613089	729
15	30×30 y 30×30	900	808201	841

En la segunda tabla se muestra el número de iteraciones requeridas para satisfacer la tolerancia solicitada al método CGM para el caso simétrico:

Ejemplo	PRIMAL#1	PRIMAL#1	DUAL#1	DUAL#2
1	2	1	2	1
2	7	7	6	5
3	9	9	7	6
4	10	10	9	7
5	11	11	10	8
6	12	11	13	9
7	12	12	13	12
8	13	12	14	12
9	13	13	15	13
10	13	13	15	14
11	13	14	15	16
12	14	14	15	15
13	14	14	15	15
14	14	14	15	15
15	15	14	15	15

En la tercer tabla se muestra el número de iteraciones requeridas para satisfacer la tolerancia solicitada al método GMRES para el caso no simétrico:

Ejemplo	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
1	2	1	2	1
2	8	6	6	6
3	10	8	8	8
4	12	10	9	9
5	13	12	9	10
6	14	12	10	10
7	15	13	11	11
8	15	14	11	11
9	16	14	11	12
10	16	15	12	12
11	17	16	12	12
12	17	16	12	13
13	17	16	13	13
14	18	17	13	13
15	18	17	13	13

Cuando la eficiencia de los algoritmos para matrices simétricas y no simétricas son comparadas, se observa que el número de iteraciones son del mismo orden y comparables con los resultados para este mismo tipo de problemas (véase [34]).

Problemas en Tres Dimensiones En la primera tabla se muestra la descomposición usada, los grados de libertad asociados al sistema y el número de vértices primales usados:

Ejemplo	Partición	Subdominios	Grados Libertad	Primales
1	$2 \times 2 \times 2$ y $2 \times 2 \times 2$	8	27	7
2	$3 \times 3 \times 3$ y $3 \times 3 \times 3$	27	512	80
3	$4 \times 4 \times 4$ y $4 \times 4 \times 4$	64	3375	351
4	$5 \times 5 \times 5$ y $5 \times 5 \times 5$	125	13824	1024
5	$6 \times 6 \times 6$ y $6 \times 6 \times 6$	216	42875	2375
6	$7 \times 7 \times 7$ y $7 \times 7 \times 7$	343	110592	4752
7	$8 \times 8 \times 8$ y $8 \times 8 \times 8$	512	250047	8575
8	$9 \times 9 \times 9$ y $9 \times 9 \times 9$	729	512000	14336
9	$10 \times 10 \times 10$ y $10 \times 10 \times 10$	1000	970299	22599

En la segunda tabla se muestra el número de iteraciones requeridas para satisfacer la tolerancia solicitada al método CGM para el caso simétrico:

Ejemplo	PRIMAL#1	PRIMAL#1	DUAL#1	DUAL#2
1	2	2	2	2
2	4	4	3	3
3	5	5	4	3
4	6	5	4	3
5	6	6	4	4
6	7	6	4	4
7	8	7	5	6
8	8	8	7	7
9	8	8	8	8

En la tercer tabla se muestra el número de iteraciones requeridas para satisfacer la tolerancia solicitada al método GMRES para el caso no simétrico:

Ejemplo	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
1	3	2	2	2
2	6	4	4	4
3	7	6	5	5
4	8	7	5	5
5	10	7	6	6
6	11	8	6	6
7	11	9	7	7
8	12	10	8	8
9	13	11	9	9

Cuando la eficiencia de los algoritmos para matrices simétricas y no simétricas son comparadas, se observa que el número de iteraciones son del mismo orden y comparables con los resultados para este mismo tipo de problemas (véase [34]).

De estos resultados, se puede concluir que los métodos de descomposición de dominio en el espacio de vectores derivados desarrollados tanto para tratar problemas simétricos y no simétricos en experimentos numéricos en dos y tres dimensiones presentan una eficiencia del mismo orden (véase [34] y [19]). Además, el desarrollo de los códigos se simplifica, al poder tratar con problemas simétricos y no simétricos en un mismo código.

17.2 Análisis de Rendimiento para Problemas Indefinidos

Para los problemas indefinidos, como es en el caso de la ecuación de Helmholtz, interesa encontrar una malla -lo más gruesa posible- en la cual el problema sea soluble sin obtener un error considerable para valores grandes de k , que normalmente generan inestabilidad numérica en los métodos de discretización, las cuales siempre se eliminan al refinar adecuadamente la malla, la ecuación utilizada es:

$$-\Delta u - k^2 u = f \tag{17.2}$$

en este caso, la discretización se realizó mediante el método de Diferencias Finitas centradas, los ejemplos se resolvieron mediante el método de GMRES con una tolerancia de 10^{-6} , con fines de ejemplificación, aquí mostramos los resultados para $k = 10$.

Para el primer ejemplo, la ecuación utilizada es en 2D; $(x, y) \in [-1, 1] \times [-1, 1]$, donde $u(x, y) = 0$ sobre $\partial\Omega$, los resultados obtenidos para las distintas descomposiciones de dominio, se muestran en la siguiente tabla:

Partición	Grados de Libertad	Primales	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
6×6 y 6×6	1225	25	8	8	8	7
10×10 y 10×10	9801	81	16	13	16	13
14×14 y 14×14	38025	169	18	15	18	15
18×18 y 18×18	104329	289	21	16	20	16
22×22 y 22×22	233289	441	20	17	21	16
26×26 y 26×26	455625	625	21	17	20	17
30×30 y 30×30	808201	841	26	18	21	17

Para el segundo ejemplo, la ecuación utilizada es en 3D; $(x, y, z) \in [-1, 1] \times [-1, 1] \times [-1, 1]$, donde $u(x, y, z) = 0$ sobre $\partial\Omega$, los resultados obtenidos para las distintas descomposiciones de dominio, se muestran en la siguiente tabla:

Partición	Grados de Libertad	Primales	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
$2 \times 2 \times 2$ y $2 \times 2 \times 2$	27	7	1	1	1	1
$3 \times 3 \times 3$ y $3 \times 3 \times 3$	512	80	4	4	4	3
$4 \times 4 \times 4$ y $4 \times 4 \times 4$	3375	351	5	4	4	3
$5 \times 5 \times 5$ y $5 \times 5 \times 5$	13824	1024	6	6	5	5
$6 \times 6 \times 6$ y $6 \times 6 \times 6$	42875	2375	7	7	6	5
$7 \times 7 \times 7$ y $7 \times 7 \times 7$	110592	4752	7	7	6	5
$8 \times 8 \times 8$ y $8 \times 8 \times 8$	250047	8575	8	8	6	5
$9 \times 9 \times 9$ y $9 \times 9 \times 9$	512000	14336	8	8	6	6
$10 \times 10 \times 10$ y $10 \times 10 \times 10$	970299	22599	9	6	6	6

De los resultados mostrados en esta sección, se puede concluir que el esquema de descomposición de dominio en el espacio de vectores derivados para problemas indefinidos presenta buenos resultados para distintos valores de k sin mostrar significativas inestabilidades numéricas en mallas burdas.

Además, haciendo los ajustes pertinentes al esquema de discretización de diferencias finitas -sin hacer cambio alguno al esquema DVS-, es posible resolver la ecuación de Helmholtz tal que no se introduzca error de truncamiento, consecuentemente se puede calcular la solución numérica exacta para la ecuación de Helmholtz para cualquier número de onda sin usar una malla fina (véase [74]).

17.3 Análisis de Rendimiento para Problemas de Advección-Difusión

En el caso de los problemas de Advección-Difusión interesa encontrar una malla -lo más gruesa posible- en la cual el problema sea soluble sin obtener un error considerable al usar valores de viscosidad pequeños que normalmente generan inestabilidad numérica en los métodos de discretización, las cuales siempre se eliminan al refinar adecuadamente la malla.

Para el primer ejemplo, la ecuación utilizada es:

$$-\nu \Delta u + \underline{b} \cdot \nabla u = 0 \tag{17.3}$$

en $(x, y) \in [0, 1] \times [0, 1]$, donde

$$u(x, y) = \begin{cases} 0, & (x, y) \in \xi_1 \\ 1, & (x, y) \in \xi_2 \end{cases} \tag{17.4}$$

y $\underline{b} = (1, 3)$, como se muestra en la figura:

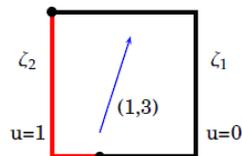


Figura 21: Dominio del problema

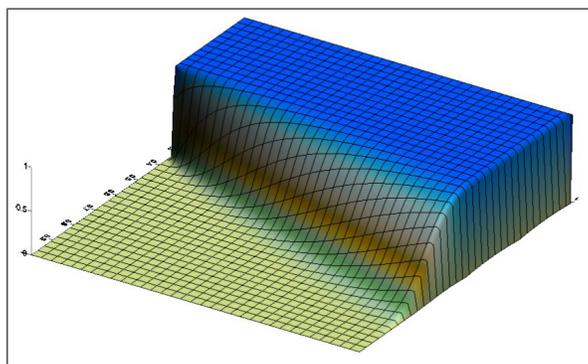


Figura 22: Solución del problema para $\nu = 0.01$

En este caso, la discretización se realizó mediante el método de Diferencias Finitas centradas y en la estabilización se usa el método de Difusión Artificial (véase [44]). Los ejemplos se resolvieron mediante el método de GMRES con una tolerancia de 10^{-6} , en una malla global de 512×512 (261,121 grados de libertad), para distintos valores de la viscosidad ν (véase [29]). Los resultados obtenidos para las distintas descomposiciones de dominio usando el método BDDC²² versus los algoritmos DVS, se muestran en la siguiente tabla:

²²Ejemplo realizado conjuntamente con Alberto Rosas Medina (véase [44]).

Partición	ν	BDDC	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
8×8 y 64×64	0.01	12	12	11	11	11
8×8 y 64×64	0.001	9	8	8	8	7
8×8 y 64×64	0.0001	9	7	7	7	7
8×8 y 64×64	0.00001	9	7	7	7	7
16×16 y 32×32	0.01	20	19	17	17	18
16×16 y 32×32	0.001	17	14	13	14	13
16×16 y 32×32	0.0001	15	13	13	13	13
16×16 y 32×32	0.00001	16	13	13	13	13
32×32 y 16×16	0.01	33	33	29	29	31
32×32 y 16×16	0.001	30	26	25	25	25
32×32 y 16×16	0.0001	28	25	25	25	25
32×32 y 16×16	0.00001	29	25	25	25	26
64×64 y 8×8	0.01	52	53	53	52	59
64×64 y 8×8	0.001	53	46	46	46	47
64×64 y 8×8	0.0001	53	45	45	47	47
64×64 y 8×8	0.00001	54	45	45	47	48

Además se muestra el residual relativo de decaimiento para la malla gruesa 16×16 y varias mallas finas en las cuales se ve que la mejor convergencia se obtiene cuando la malla fina se incrementa y la convergencia es lenta cuando el subdominio tiene una pequeña cantidad de grados de libertad.

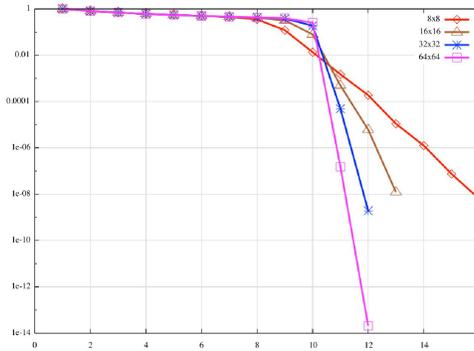


Figura 23: Residual relativo para la malla local de 16×16 , en este caso $\underline{b} = (1, 3)$ y $\nu = 0.00001$ que corresponde a un valor de $Pe = 3.16e + 5$.

Para el segundo ejemplo, la ecuación a trabajar es

$$-\nu \Delta u + \underline{b} \cdot \nabla u + cu = 0 \tag{17.5}$$

en $(x, y) \in [-1, 1] \times [0 - 1, 1]$, donde

$$u(x, y) = 1 \begin{cases} y = -1, & 0 < x \leq 1 \\ y = 1, & 0 < x \leq 1 \\ x = 1, & -1 \leq y \leq 1 \end{cases}$$

$$u(x, y) = 0, \quad \text{en cualquier otro caso} \tag{17.6}$$

el coeficiente advectivo está dado por $\underline{b} = (y, -x)$, el valor de $c = 10^{-4}$.

En este caso, la discretización se realizó mediante el método de Diferencias Finitas centradas y en la estabilización se usa el método de Difusión Artificial (véase [44]). Los ejemplos se resolvieron mediante el método de GMRES con una tolerancia de 10^{-6} , en una malla global de 32×32 (961 grados de libertad), para distintos valores de la viscosidad ν (véase [52]), cuya solución es mostrada en la gráfica:

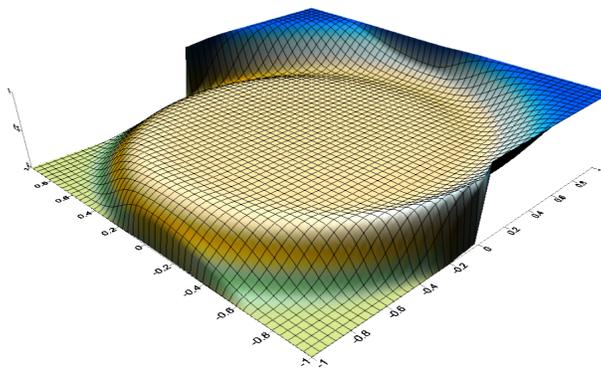


Figura 24: Solución del problema para $\nu = 0.01$

Los resultados obtenidos para las distintas descomposiciones de dominio usando el método FETI²³ versus los algoritmos DVS, se muestran en la siguiente tabla:

²³Ejemplo realizado conjuntamente con Alberto Rosas Medina (véase [44]).

Partición	ν	FETI-DP	PRIMAL#1	DUAL#1	PRIMAL#2	DUAL#2
4×4 y 8×8	1	11	9	8	8	8
4×4 y 8×8	0.01	12	11	8	10	9
4×4 y 8×8	0.001	23	20	16	20	16
4×4 y 8×8	0.0001	45	24	19	24	18
4×4 y 8×8	0.00001	69	24	19	24	18
8×8 y 4×4	1	10	9	8	8	8
8×8 y 4×4	0.01	11	16	9	10	13
8×8 y 4×4	0.001	27	24	21	24	22
8×8 y 4×4	0.0001	68	32	25	30	26
8×8 y 4×4	0.00001	111	33	24	29	27
16×16 y 2×2	1	9	8	6	6	6
16×16 y 2×2	0.01	16	26	8	9	21
16×16 y 2×2	0.001	63	47	23	28	41
16×16 y 2×2	0.0001	176	48	29	34	42
16×16 y 2×2	0.00001	200	48	30	34	42

Para el tercer ejemplo, la ecuación a trabajar es

$$-\nu \Delta u + \underline{b} \cdot \nabla u + cu = 0 \quad (17.7)$$

en $(x, y) \in [-1, 1] \times [-1, 1]$, donde

$$\begin{aligned} u(x, y) &= 1 \begin{cases} x = -1, & -1 < y \leq 1 \\ y = 1, & -1 \leq x \leq 1 \end{cases} \\ u(x, y) &= 0, \quad y = -1, -1 \leq x \leq 1 \\ u(x, y) &= \frac{1+y}{2}, \quad x = 1, -1 \leq y \leq 1 \end{aligned} \quad (17.8)$$

el coeficiente advectivo está dado por $\underline{b} = (\frac{1+y}{2}, 0)$, el valor de $c = 10^{-4}$.

En este caso, la discretización se realizó mediante el método de Diferencias Finitas centradas y en la estabilización se usa el método de Difusión Artificial (véase [44]), Los ejemplos se resolvieron mediante el método de GMRES con una tolerancia de 10^{-6} en la norma infinita en una malla global de 32×32 (961 grados de libertad), para distintos valores de la viscosidad ν (véase [52]), cuya solución es mostrada en la gráfica:

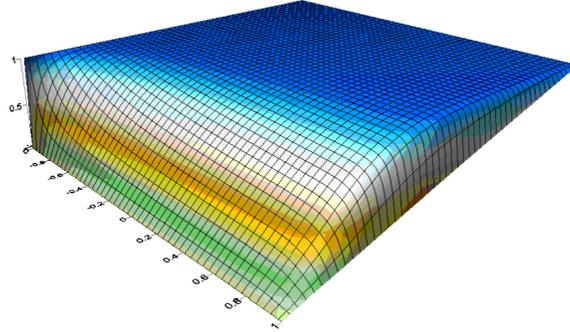


Figura 25: Solución del problema para $\nu = 0.01$

Los resultados obtenidos para las distintas descomposiciones de dominio usando el método FETI-DP²⁴ versus los algoritmos DVS, se muestran en la siguiente tabla:

Partición	ν	FETI-DP	PRIMAL#1	DUAL#1	PRIMAL#2	DUAL#2
4×4 y 8×8	1	13	10	10	8	8
4×4 y 8×8	0.01	13	11	10	8	7
4×4 y 8×8	0.001	9	8	9	6	6
4×4 y 8×8	0.0001	10	10	10	4	4
4×4 y 8×8	0.00001	11	9	10	3	4
4×4 y 8×8	0.000001	11	9	9	2	3
8×8 y 4×4	1	44	9	9	8	8
8×8 y 4×4	0.01	34	15	14	10	10
8×8 y 4×4	0.001	16	15	15	10	10
8×8 y 4×4	0.0001	16	27	28	9	9
8×8 y 4×4	0.00001	16	32	32	8	8
8×8 y 4×4	0.000001	16	25	25	6	5
16×16 y 2×2	1	159	8	8	6	5
16×16 y 2×2	0.01	98	22	21	9	8
16×16 y 2×2	0.001	38	37	37	18	18
16×16 y 2×2	0.0001	33	48	48	23	22
16×16 y 2×2	0.00001	46	42	41	20	20
16×16 y 2×2	0.000001	51	37	36	15	15

²⁴Ejemplo realizado conjuntamente con Alberto Rosas Medina (véase [44]).

De los resultados mostrados en esta sección, se puede concluir que el esquema de descomposición de dominio en el espacio de vectores derivados para problemas de Advección-Difusión presenta una eficiencia del mismo orden y en algunos casos mejoran la mostrada por los métodos FETI y BDD (véase [29] y [52]).

17.4 Análisis de Rendimiento para Sistemas de Ecuaciones

En esta sección se muestra como usar el esquema DVS para resolver problemas con condiciones de frontera Dirichlet donde los desplazamientos son cero sobre la frontera del cuerpo elástico que ocupa el dominio Ω del espacio físico, donde sobre cada subdominio Ω_i que forma la partición gruesa del dominio Ω es resuelto el problema local usando el Método de Elemento Finito (FEM), usando funciones lineales como base.

En el caso de sistemas de ecuaciones, se resolvió²⁵ un sistema de ecuaciones diferenciales parciales en tres dimensiones que corresponde a la ecuación de elasticidad lineal

$$(\lambda + \mu) \nabla \nabla \cdot \underline{u} + \mu \Delta \underline{u} = \underline{f}_\Omega, \text{ en } \Omega \quad (17.9)$$

la cual es sujeta a las condiciones de frontera Dirichlet

$$\underline{u} = 0, \text{ en } \partial\Omega \quad (17.10)$$

el dominio Ω para los experimentos numéricos es un cubo unitario homogéneo isotrópico lineal elástico. En todos nuestros experimentos los nodos primales fueron localizados en las aristas de los subdominios de la partición gruesa, lo cual es suficiente para que la matriz $\underline{\underline{A}}^t$ no sea singular.

Considerando λ y μ iguales a uno, La solución analítica de este problema se escribe como

$$\underline{u} = (\sin \pi x \sin \pi y \sin \pi z, \sin \pi x \sin \pi y \sin \pi z). \quad (17.11)$$

En este caso el operador es simétrico y positivo definido, por ello se usa el método iterativo de Gradiente Conjugado para resolver el sistema lineal de ecuaciones que se genera en el esquema DVS, con una tolerancia de 10^{-7} (véase [45]). Los resultados obtenidos para las distintas descomposiciones de dominio usando el cluster Olintlali se muestran en la siguiente tabla:

²⁵Ejemplo realizado conjuntamente con Iván Contreras Trejo (véase [45]).

Partición	Subdominios	DOF	PRIMAL#1	DUAL#1	PRIMAL#2	DUAL#2
$5 \times 5 \times 5$ y $5 \times 5 \times 5$	125	41472	8	7	9	9
$6 \times 6 \times 6$ y $6 \times 6 \times 6$	216	128625	8	8	10	10
$7 \times 7 \times 7$ y $7 \times 7 \times 7$	343	331776	8	8	11	11
$8 \times 8 \times 8$ y $8 \times 8 \times 8$	512	750141	8	8	12	12

Nótese que, el código desarrollado y usado para problemas escalares que originalmente se desarrolló para resolver una sola ecuación usando en la discretización al método de Diferencias Finitas, fue extendido para resolver problemas con el método de Elemento Finito para resolver sistemas de ecuaciones.

17.5 Análisis de Rendimiento en Equipos Paralelos

Para conocer el análisis de rendimiento en equipos paralelos de los métodos desarrollados de descomposición de dominio en el espacio de vectores derivados, se realizaron varias pruebas con la finalidad de conocer la eficiencia y escalabilidad de los códigos y por ende de los métodos en distintos equipos paralelos a los que se tuvo acceso, estos incluyen equipos con 8, 22, 104 y 1024 Cores.

Primeramente, es menester fundamental el encontrar la mejor descomposición de dominio para el problema a trabajar al usar la implementación secuencial y paralela acorde al equipo del que se disponga en aras de obtener la más alta eficiencia posible, después cuando el caso lo permite, se muestran las distintas métricas utilizables y sus limitaciones al aplicarlas en problemas de descomposiciones finas; por último se muestra la escalabilidad del esquema DVS usando hasta 1024 Cores.

17.5.1 Selección Óptima de una Descomposición del Dominio

Para comenzar con la selección óptima de la descomposición del dominio Ω , se toma el problema dado por la Ec.(16.1) como caso particular de la Ec.(17.1) en dos dimensiones con una descomposición fina de 1024×1024 nodos - 1,048,576 grados de libertad- del dominio Ω , donde por ejemplo se toma sin pérdida de generalidad el algoritmo PRIMAL#1, calculado los tiempos de ejecución en los cuales se usa de uno a ocho Cores de la PC Antipolis y

probando las diferentes descomposiciones²⁶ del dominio -que van desde 2×2 y 512×512 hasta 512×512 y 2×2 - se muestran en la siguiente tabla:

	1 Core	2 Cores	3 Cores	4 Cores	5 Cores	6 Cores	7 Cores	8 Cores
Partición	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo
2×2 y 512×512	16465	10659	7207	7105	4641			
4×4 y 256×256	2251	5063	2252	2103	1643	1233	1068	947
8×8 y 128×128	855	885	482	395	314	311	283	272
16×16 y 64×64	321	348	190	149	121	125	118	117
32×32 y 32×32	26	39	26	24	23	21	21	21
64×64 y 16×16	205	595	485	477	481	461	469	469
128×128 y 8×8	1026	5453	5352	5431	5633	5843	5843	5903
256×256 y 4×4	8544	26167	25892	25902	25939	25950	25969	26003
512×512 y 2×2	34845	64230	63293	63308	63389	63475	63502	63693

Por ejemplo, suponiendo que se quiere resolver una descomposición de 2×2 y 512×512 y usar la menor cantidad de Cores posible, entonces se tienen algunas opciones para mantener un buen balanceo de cargas -en este caso se tienen 4 subdominios- usando 3 ó 5 Cores:

²⁶Para las corridas secuenciales usando métodos de descomposición de dominio, el mejor tiempo de ejecución se obtuvo en una descomposición de 32×32 y 32×32 en 26 segundos -el tiempo de ejecución para el programa secuencial de elemento finito que resuelve el sistema lineal algebraico asociado mediante factorización Cholesky fue de 111 segundos-. Para las corridas en paralelo se obtuvo el mejor tiempo de ejecución en una descomposición de 32×32 y 32×32 con un tiempo de 21 segundos usando 6 Cores -uno para el maestro y 5 para los esclavos-.

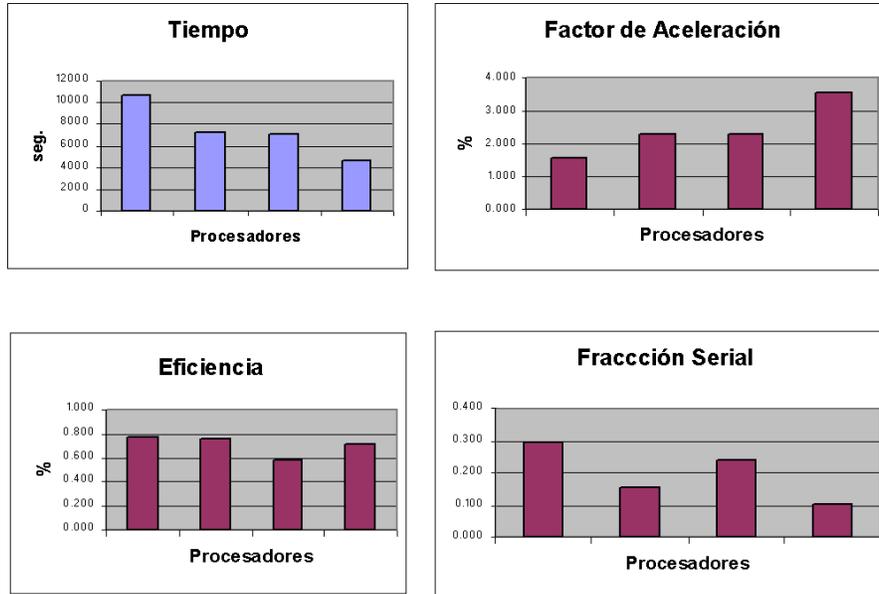


Figura 26: Métricas para la descomposición 2×2 y 512×512

- Si se desea usar 1 Core para el nodo maestro y 2 Cores para los nodos esclavos -dos subdominios por Core-, entonces el factor de aceleración²⁷ es

$$S(3) = T(1)/T(3) = 16465/7207 = 2.28,$$

la eficiencia es

$$E(3) = T(1)/(3 * T(3)) = 16465/(3 * 7207) = 0.761,$$

y la fracción serial es

$$F(3) = \frac{\frac{1}{S(3)} - \frac{1}{3}}{1 - \frac{1}{3}} = 0.158.$$

²⁷El factor de aceleración S es tal que $1 \leq S(n) \leq n$, la eficiencia E es tal que $1/n \leq E(n) \leq 1$ y la fracción serial F es tal que $0 \leq F(n) \leq 1$.

Se considera que en el caso ideal, el factor de aceleración debería aumentar linealmente al aumentar el número de procesadores $S(p) \simeq p$; por su parte la eficiencia debería de ser cercana a la unidad cuando el Hardware se está usando de forma eficiente y en caso contrario se desaprovecha este; por último la fracción serial debería tender a cero y cualquier aumento indica una sobrecarga en los procesos de comunicación.

- Si se desea usar 1 Core para el nodo maestro y 4 Cores para los nodos esclavos -un subdominio por Core-, entonces el factor de aceleración es

$$S(5) = T(1)/T(5) = 16465/4641 = 3.548,$$

la eficiencia es

$$E(5) = T(1)/(5 * T(5)) = 16465/(5 * 4641) = 0.709$$

y la fracción serial es

$$F(5) = \frac{\frac{1}{S(5)} - \frac{1}{5}}{1 - \frac{1}{5}} = 0.102.$$

En otro ejemplo, suponiendo que se quiere resolver una descomposición de 32×32 y 32×32 y usar la menor cantidad de Cores posible, entonces se tienen algunas opciones para mantener un buen balanceo de cargas -en este caso se tienen 1024 subdominios- usando 3 ó 5 Cores:

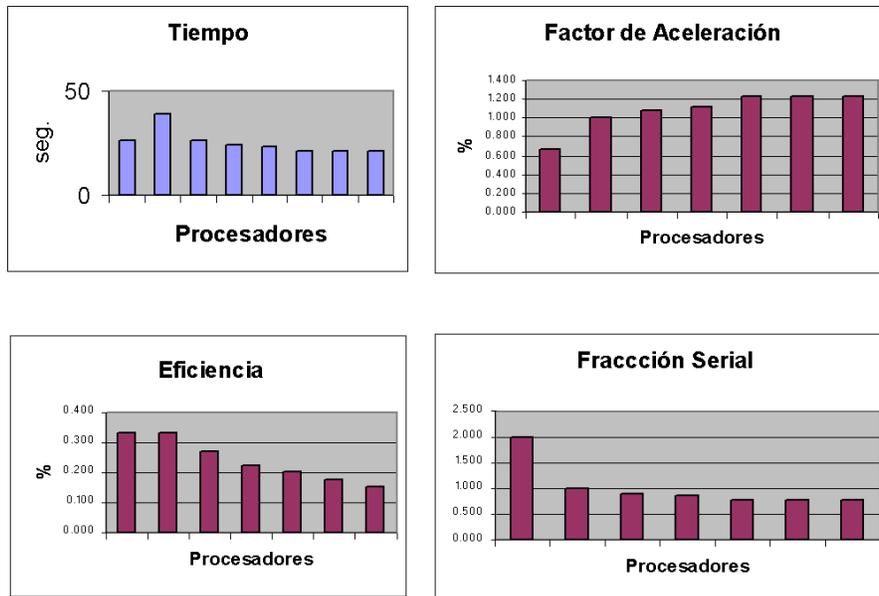


Figura 27: Métricas para la descomposición 32×32 y 32×32

- Si se desea usar 1 Core para el nodo maestro y 2 Cores para los nodos esclavos -512 subdominios por Core-, entonces el factor de aceleración es

$$S(3) = T(1)/T(3) = 26/26 = 1,$$

la eficiencia es

$$E(3) = T(1)/(3 * T(3)) = 26/(3 * 26) = 0.333$$

y la fracción serial es

$$F(3) = \frac{\frac{1}{S(3)} - \frac{1}{3}}{1 - \frac{1}{3}} = 1.$$

- Si se desea usar 1 Core para el nodo maestro y 4 Cores para los nodos esclavos -256 subdominios por Core-, entonces el factor de aceleración es

$$S(5) = T(1)/T(5) = 26/23 = 1.130,$$

la eficiencia es

$$E(5) = T(1)/(5 * T(5)) = 26/(5 * 23) = 0.377,$$

y la fracción serial es

$$F(5) = \frac{\frac{1}{S(5)} - \frac{1}{5}}{1 - \frac{1}{5}} = 0.856.$$

Nótese que la descomposición usada en el primer ejemplo dista mucho de ser la óptima²⁸, ya que la descomposición de 32×32 y 32×32 usada en el segundo ejemplo genera el mejor tiempo de ejecución tanto en secuencial como en paralelo, pero las métricas no reflejan esta mejora, aunque el tiempo de ejecución es mínimo. Por ello es necesario siempre hacer corridas de prueba buscando la descomposición que presente el menor tiempo de ejecución posible para el equipo paralelo con el que se cuente. Estas pruebas dependen fuertemente de la capacidad computacional de cada Core y de la red usada para interconectar los Cores que forman parte del equipo paralelo.

²⁸En la descomposición de 2×2 y 512×512 el tiempo de ejecución secuencial es 16,465 seg., en paralelo usando 3 Cores es de 7,207 seg. y en 5 Cores es de 4,641 seg. Mientras que para la descomposición de 32×32 y 32×32 , el tiempo de ejecución secuencial es de 26 seg., en paralelo usando 3 Cores es de 26 seg. y en 5 Cores es de 23 seg.

Observación 7 *Nótese que esta forma de medir la eficiencia, el factor de aceleración y la fracción serial tiene un detalle fino, ya que el tiempo de ejecución tomado en un procesador -para este caso es de 16465 segundos en la descomposición 2×2 y 512×512 - dista mucho de ser el mejor tiempo posible para el problema global de 1024 nodos -26 segundos-. Esto puede ser una limitante para obtener valores adecuados en las métricas; y medir la eficiencia al usar equipo paralelo cuando se trabaja con la resolución de dominios en los cuales se realiza una descomposición fina; particularmente cuando las descomposiciones son adecuadas para cientos de Cores, pues las pruebas en un Core o en pocos Cores no son posibles de realizar por el consumo excesivo de recursos computacionales y por que no son conmensurables con las corridas secuenciales.*

Además, de los datos de las corridas mostradas en la tabla, es notorio el efecto del mal balanceo de carga, nótese que:

- Para la malla de 32×32 y 32×32 el mejor tiempo de ejecución se obtiene en 6 Cores -21 segundos- y al aumentar el número de Cores en la corrida, no hay disminución del tiempo de cálculo.
- Para la malla de 512×512 y 2×2 el aumento en el número de procesadores sólo incide en un aumento en el tiempo de ejecución.
- En particular, para la malla de 2×2 y 512×512 al usar 3 Cores -sólo dos son usados realmente para el cálculo ya que el tercer Core se usa para la asignación de tareas y el control de los nodos esclavos- el factor de aceleración 2.28 es el esperado para el esquema Maestro-Eslavo.

17.5.2 Análisis de Rendimiento Usando Métricas

En esta sección se muestra mediante particiones relativamente pequeñas del dominio, el uso de las métricas -aceleración, eficiencias y fracción serial- en las cuales es posible obtener una alta eficiencia computacional cuando se proporciona una descomposición del dominio adecuada para el equipo paralelo con el que se cuenta.

Haciendo uso del Cluster Pohualli de 104 Cores, a continuación se presentan varias tablas en las cuales se muestran las métricas para diferentes descomposiciones del dominio Ω .

1) Para una descomposición de 4×4 y 150×150 -360,000 grados de libertad- se obtiene

Cores	Tiempo	Aceleración	Eficiencia	Frac. Ser.
1	267			
3	146	1.82	0.60	0.32
5	85	3.14	0.62	0.14
9	56	4.76	0.52	0.11
17	33	8.09	0.47	0.06

2) Para una descomposición de 4×4 y 200×200 -640,000 grados de libertad- se obtiene

Cores	Tiempo	Aceleración	Eficiencia	Frac. Ser.
1	1082			
3	391	2.76	0.92	0.04
5	216	5.0	1.00	0.00
9	146	7.41	0.82	0.02
17	82	13.19	0.77	0.01

3) Para una descomposición de 4×4 y 250×250 -1,000,000 grados de libertad- se obtiene

Cores	Tiempo	Aceleración	Eficiencia	Frac. Ser.
1	2628			
3	946	2.77	0.92	0.039
5	539	4.87	0.97	0.006
9	329	7.98	0.88	0.015
17	184	14.20	0.83	0.012

4) Para una descomposición de 4×4 y 300×300 -1,440,000 grados de libertad- se obtiene

Cores	Tiempo	Aceleración	Eficiencia	Frac. Ser.
1	5295			
3	2538	2.08	0.69	0.218
5	1391	3.80	0.76	0.078
9	804	6.58	0.73	0.045
17	441	12.00	0.70	0.025

De estas tablas se desprende que seleccionando la descomposición adecuada se pueden tener excelentes resultados en la eficiencia como es el caso

de la descomposición 4×4 y 200×200 . Además se muestra una gama de otras eficiencias según el número de procesadores usados y la descomposición seleccionada. Nótese que en todos los casos la fracción serial disminuye sustancialmente con el aumento del número de procesadores.

Otros ejemplos interesantes en los cuales se muestra el efecto de mandar descomposiciones no adecuadas y que se reflejan en una baja eficiencia computacional sin importar el aumento del número de procesadores, pero en todos los casos el tiempo de ejecución siempre disminuye:

i) Para una descomposición de 8×8 y 250×250 -4,000,000 grados de libertad- en el Cluster Kanbalam se obtiene

Cores	Tiempo	Aceleración	Eficiencia	Frac. Ser.
1	11366			
3	5541	2.05	0.68	0.23
5	3011	3.77	0.75	0.08
9	1855	6.12	0.68	0.05
17	1031	11.02	0.64	0.03
33	595	19.10	0.57	0.02
65	375	30.30	0.46	0.01

ii) Para una descomposición de 10×9 y 250×250 -5,625,000 grados de libertad- en el Cluster Pohualli se obtiene

Cores	Tiempo	Aceleración	Eficiencia	Frac. Ser.
1	19387			
6	4777	4.05	0.67	0.09
11	2702	7.17	0.65	0.05
46	801	24.20	0.52	0.02
91	509	38.08	0.41	0.01

De todos estos ejemplos se desprende que buscando la adecuada descomposición es posible encontrar eficiencias altas para el esquema DVS, pero siempre se tiene que tener en cuenta el buscar el menor tiempo de ejecución -véase sección anterior- antes que una alta eficiencia con un mayor tiempo de ejecución.

Por otro lado, pese a que Kanbalam es más eficiente²⁹ que los otros Clusters a los que se tuvo acceso -en particular Pohualli-, es posible encontrar una descomposición del dominio que mejore el tiempo de ejecución, aún en equipos con recursos inferiores, para ello es necesario aprovechar las características propias del Hardware del Cluster haciendo una adecuada selección de la descomposición del dominio. Para mostrar esto, se toma una descomposición de 32×32 y 150×150 -23,040,000 grados de libertad- en ambos Clusters con los siguientes resultados:

Cores	Pohualli	Kanbalam
16	9158 seg	ND
32	5178 seg	5937 seg
64	3647 seg	4326 seg
100	2661 seg	
128		2818 seg

Como se muestra en la tabla, en todos los casos el Cluster Pohualli usando como máximo 100 Cores obtiene un tiempo de cálculo inferior al que requiere Kanbalam usando a lo más los 128 Cores.

Haciendo uso de las métricas de aceleración y eficiencia relativa³⁰ se tiene que para el Cluster Kanbalam $S_{128}^{32} = 5937/2818 = 2.10$ donde lo esperado sería $S_{128}^{32} = 32/128 = 4.00$, para el caso de la eficiencia $E_{128}^{32} = (32/128) * (5937/2818) = 0.52$.

En el caso del Cluster Pohualli se tiene que $S_{100}^{16} = 9158/2661 = 3.44$ dónde lo esperado sería $S_{100}^{16} = 16/100 = 6.35$, para el caso de la eficiencia $E_{100}^{16} = (16/100) * (9158/2661) = 0.55$.

Haciendo uso del mismo número de Cores base para Pohualli que para Kanbalam, se tiene que $S_{100}^{32} = 5178/2661 = 1.94$ dónde lo esperado sería $S_{100}^{16} = 32/100 = 3.12$, para el caso de la eficiencia $E_{100}^{16} = (32/100) * (5178/2661) = 0.62$;

²⁹El Cluster Kanbalam esta formado de procesadores AMD Opteron a 2.6 GHz de 64 bits, cada 4 Cores con 8 GB de RAM interconectados con un switch de 10 Gbps de baja latencia.

El Cluster Pohualli esta formado de procesadores Intel Xeon a 2.33 GHz de 64 bits, cada 8 Cores cuentan con 32 GB de RAM interconectados con un switch de 1 Gbps.

³⁰Aceleración relativa es $S_p^{p'} = \frac{T_p}{T_{p'}}$ para $p \geq p'$, en la cual se espera que $S_p^{p'} \simeq \frac{p}{p'}$ y eficiencia relativa es $E_p^{p'} = \frac{p'}{p} S_p^{p'} = \frac{p'}{p} \frac{T_p}{T_{p'}}$.

De todo lo anterior, se desprende que el Cluster Pohualli obtiene valores de una aceleración y eficiencias relativas ligeramente mejores que el Cluster Kanbalam, pero esto no se refleja en la disminución de casi 6% del tiempo de ejecución y del uso de 28 Cores menos.

Además, el costo computacional³¹ $C_p = P * T_p$, que para el caso del cluster Kanbalam es $C_{128} = 360,704$ y en Pohualli es $C_{100} = 266,100$ que representa una disminución de 27%; además de un factor muy importante, el Cluster Pohualli tuvo un costo monetario mucho menor con respecto del Cluster Kanbalam.

17.5.3 Escalabilidad del Esquema DVS

Por último, se realizaron pruebas³² con el Cluster Kanbalam, mediante una petición especial para tener acceso a los 1024 Cores del Cluster, a la cual el Comité Técnico del mismo dio acceso después de concluir un reparación mayor del equipo que obligó a apagar el Cluster, este acceso sólo se dio por unas horas y de forma exclusiva, lo cual agradezco enormemente, ya que el Cluster tiene una gran demanda dentro y fuera de la UNAM.

Las pruebas realizadas se hicieron usando desde 32 hasta 1024 Cores, para las descomposiciones de 31×33 y 150×150 -23,017,500 grados de libertad-, 31×33 y 200×200 -40,920,000 grados de libertad- y 31×33 y 250×250 -63,937,500 grados de libertad- se obtienen los siguientes tiempos de ejecución.

Subdominio	Cores					
	32	64	128	256	512	1024
31×33 y 150×150	7315 s	4016 s	2619 s	1941 s	1541 s	1298 s
31×33 y 200×200	ND	16037 s	4916 s	3166 s	2688 s	2295 s
31×33 y 250×250	ND	ND	26587 s	8716 s	6388 s	ND

³¹El costo o trabajo de resolver un problema en paralelo es el producto del tiempo de cálculo en paralelo T_p por el número de procesadores usado P y se representa por $C_p = P * T_p$.

³²No todas las pruebas que se plantearon fueron posibles de realizar, en algunos casos la limitante fue las características físicas de los equipos computacionales, en otros es el acceso limitado y de corta duración -para el uso de 256, 512 y 1024 Cores en el Cluster Kanbalam sólo se dispuso de unas cuantas horas de cómputo y en las cuales, varios Cores del Cluster presentaron fallas de Hardware por lo que algunas corridas no se concluyeron de forma satisfactoria-.

En donde si usamos las métricas de aceleración y eficiencia relativas obtenemos los siguientes resultados

Subdominio	Aceleración	Aceleración esperada	Eficiencia
31×33 y 150×150	$S_{512}^{32} = 4.7$	$S_{512}^{32} = 32$	$E_{512}^{32} = 0.2$
31×33 y 200×200	$S_{512}^{64} = 5.9$	$S_{512}^{32} = 8$	$E_{512}^{64} = 0.7$
31×33 y 250×250	$S_{512}^{128} = 4$	$S_{512}^{32} = 4$	$E_{512}^{128} = 1.0$

De esta última tabla $-E_{512}^{128} = 1.0$ para la descomposición 31×33 y 250×250 -, se desprende que los algoritmos desarrollados son altamente escalables en equipos paralelos, ya que es posible obtener una alta eficiencia al encontrar descomposiciones adecuadas al Hardware. Y que pueden usarse para resolver problemas que involucren una gran cantidad de grados de libertad.

17.6 Criterios Integrales para Evaluar el Esquema DVS

En el desarrollo e implementación numérica de los distintos métodos de descomposición de dominio, es necesario medir de alguna forma la eficiencia de los diversos métodos entre sí, algunos criterios comúnmente usados son:

1. Dado un dominio Ω y una descomposición fija, usar el número de iteraciones como criterio de eficiencia.
2. Dado un dominio Ω y haciendo refinamientos de la partición, usar el número de iteraciones como criterio de eficiencia.
3. Dado un dominio Ω y una descomposición del mismo, buscar aquella partición en la que el tiempo de ejecución sea mínimo al variar las particiones posibles.

En principio, estas formas de medir la eficiencia de los diversos métodos no deberían de ser excluyentes entre sí, por el contrario, juntas dan un criterio robusto de la eficiencia de un método de descomposición de dominio para un problema en particular implementado en un equipo de cómputo en las cuales ciertas descomposiciones son posibles -ya sea por limitaciones fenomenológicas o por cuestiones computacionales-.

Para mostrar las implicaciones de las distintas formas de medir la eficiencia, se hace un análisis de las diversas opciones para cada uno de los casos.

1.- Dado un dominio Ω y una descomposición fija, usar el número de iteraciones como criterio de eficiencia En este caso, se usa la Ec.(17.1) como simétrica, tomando una descomposición del dominio Ω en tres dimensiones en la cual se toma una malla gruesa $10 \times 10 \times 10$ que genera 10,000 subdominios y en la que cada subdominio es descompuesto en $10 \times 10 \times 10$ elementos, los grados de libertad asociados al sistema son 970,299, donde el número de vértices primales usados es de 22,599. Obteniendo los siguientes resultados:

	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
Iteraciones:	8	8	8	8

Aquí, lo único que se observa, es que todos los métodos obtienen la misma eficiencia global en cuanto al número de iteraciones, pero nada dice de los tiempos involucrados en la ejecución. Si ahora se toma en cuenta los tiempos de ejecución en un procesador se obtiene:

	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
Tiempo:	1,380s	1,387s	1,490s	1,520s

Y si se usan varios procesadores de un Cluster -en este ejemplo se usó el Cluster Pohnualli- se obtiene:

Cores	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
3	966s	965s	930s	953s
11	184s	186s	175s	181s
101	28s	29s	27s	27s

Esta forma integral de medir la eficiencia, da una idea más realista de la eficiencia de los métodos, pero nuevamente hay que tomar en cuenta que los tiempos de ejecución dependen directamente de la arquitectura de cómputo en la que se realicen las pruebas, en especial del balanceo de la carga de trabajo, de la infraestructura de red que interconecten los nodos del Cluster y si estos son Cores virtuales o reales.

2.- Dado un dominio Ω y haciendo refinamientos de la partición, usar el número de iteraciones como criterio de eficiencia En este caso, se usa la Ec.(17.1) como simétrica, se toma una descomposición del dominio Ω en dos dimensiones, en la primer tabla se muestra la descomposición usada, el número de subdominios, los grados de libertad asociados al sistema y el número de vértices primales usados:

Ejemplo	Partición	Subdominios	Grados Libertad	Primales
1	22×22 y 22×22	484	233,289	441
2	24×24 y 24×24	576	330,625	529
3	26×26 y 26×26	676	455,625	625
4	28×29 y 28×28	784	613,089	729
5	30×30 y 30×30	900	808,201	841

En la segunda tabla se muestra el número de iteraciones requeridas para alcanzar la tolerancia solicitada al método:

Ejemplo	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
1	13	14	15	16
2	14	14	15	15
3	14	14	15	15
4	14	14	15	15
5	15	14	15	15

En la siguiente tabla se muestra el tiempo de ejecución en un procesador para concluir las iteraciones:

Ejemplo	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
1	8s	7s	14s	27s
2	13s	13s	21s	40s
3	19s	19s	33s	61s
4	25s	27s	44s	85s
5	36s	38s	61s	116s

En la última tabla se muestra el tiempo de ejecución en 4 procesadores para concluir las iteraciones:

Ejemplo	PRIMAL#1	PRIMAL#2	DUAL#1	DUAL#2
1	2.9s	2.95s	2.93s	2.99s
2	4.80s	4.89s	4.81s	4.85s
3	7.2s	7.4s	7.3s	7.4s
4	8.92s	8.95s	8.91s	8.93s
5	13.02s	13.05s	13.02s	13.3s

Nuevamente, esta forma integral de medir la eficiencia, da una idea más realista de la eficiencia de los métodos, pero nuevamente hay que tomar en

cuenta que los tiempos de ejecución dependen directamente de la arquitectura de cómputo en la que se realicen las pruebas, en especial del balanceo de la carga de trabajo, de la infraestructura de red que interconecten los nodos del Cluster y si estos son Cores virtuales o reales.

3.- Dado un dominio Ω y una descomposición del mismo, buscar aquella partición en la que el tiempo de ejecución sea mínimo al variar las particiones posibles Por último, supóngase que de-seo resolver la Ec.(16.1) con un dominio Ω mediante una discretización de 1024×1024 nodos (1,048,576 grados de libertad) mediante el algoritmo NN-NP-PRIMAL#1 dado por la Ec.(13.21), de manera inmediata surgen las siguientes preguntas: ¿cuáles son las posibles descomposiciones validas? y ¿en cuántos procesadores se pueden resolver cada descomposición?. Para este ejemplo en particular, sin hacer la tabla exhaustiva, se tiene

Partición	Subdominios	Procesadores
2x2 y 512x512	4	2,3,5
4x4 y 256x256	16	2,3,5,9,17
8x8 y 128x128	64	2,3,5,9,17,33,65
16x16 y 64x64	256	2,3,5,9,17,33,65,129,257
32x32 y 32x32	1024	2,3,5,9,17,33,65,129,...,1025
64x64 y 16x16	4096	2,3,5,9,17,33,65,129,...,4097
128x128 y 8x8	16384	2,3,5,9,17,33,65,129,...,16385
256x256 y 4x4	65536	2,3,5,9,17,33,65,129,...,65537
512x512 y 2x2	262144	2,3,5,9,17,33,65,129,...,262145

De esta tabla es posible seleccionar las descomposiciones que se adecuen a las características del equipo paralelo con que se cuente, para evaluar el tiempo de ejecución de este ejemplo use la PC Antipolis, obteniendo resultados mostrados en la tabla de la sección (17.5.1).

De estos resultados, se desprende que, dependiendo del tamaño de la malla gruesa -número de subdominios a trabajar- y de la malla fina, es siempre posible encontrar una descomposición de dominio en que el tiempo de cálculo sea mínimo, tanto al usar un solo Core -programa secuencial 26 segundos-, como al usar múltiples Cores interconectados mediante la biblioteca de paso de mensajes MPI -el tiempo mínimo se obtuvo usando 6 Cores en 21 segundos-, pero es también notorio el efecto que genera el mal balanceo de carga, el cual se refleja en que no disminuye el tiempo de ejecución al aumentar el número

de procesadores y en algunos casos el tiempo aumenta conforme se agregan más Cores.

Nótese que conforme la partición en los subdominios se hace más fina, se incrementa notablemente el tiempo de cálculo necesario para resolver los sistemas lineales asociados a los subdominios, en particular en la resolución del sistema lineal asociado a $\left(\underline{A}_{\text{III}}\right)^{-1}$, si el número de nodos por subdominio es grande puede que exceda la cantidad de memoria que tiene a su disposición el Core y por el contrario, un número pequeño de nodos generarían una infrautilización del poder computacional de los nodos esclavos.

Por otro lado, el refinamiento de la malla gruesa, involucra un aumento considerable del número de objetos subdominio en los nodos esclavos, con los que el nodo maestro tendrá comunicación, incrementando la granularidad de las comunicaciones, con la consecuente degradación en la eficiencia.

De todo lo anterior se pueden hacer algunas observaciones importantes Para una evaluación objetiva e íntegra de la eficiencia de los diversos métodos de descomposición de dominio -en particular de los desarrollados- y su implementación computacional en una arquitectura de cómputo particular, es necesario tomar en cuenta los siguientes factores:

- Número de iteraciones para una descomposición dada.
- Número de iteraciones para diferentes particiones de una descomposición dada.
- Elección de la partición que genere el menor tiempo de ejecución para un problema en una arquitectura de cómputo específica.

Estas formas de medir la eficiencias en su conjunto, dan una idea realista de la eficiencia de los métodos, pero hay que tomar en cuenta que los tiempos de ejecución dependen directamente de la arquitectura de cómputo en la que se realicen las pruebas, en especial del balanceo de la carga de trabajo, de la infraestructura de red que interconecten los nodos del Cluster y si estos son Cores virtuales o reales.

18 Conclusiones y Trabajo Futuro

Los modelos matemáticos (véase [37] y [10]) de muchos sistemas de interés, incluyendo una gran cantidad de sistemas importantes de Ciencias de la Tierra e Ingeniería, conducen a una gran variedad de ecuaciones diferenciales parciales cuyos métodos de solución están basados en el procesamiento de sistemas algebraicos de gran escala. Además, la increíble expansión experimentada por el Hardware y Software computacional existente, ha hecho posible el efectivo tratamiento de problemas de un cada vez mayor incremento de diversidad y complejidad que poseen las aplicaciones Científicas y de Ingeniería.

El cómputo en paralelo destaca entre las nuevas herramientas computacionales y para hacer uso efectivo de los equipos de cómputo de alto desempeño más avanzados disponibles actualmente, el Software que aproveche al máximo el equipo paralelo es requerido. Los métodos de descomposición de dominio han sido desarrollados precisamente para hacer un efectivo tratamiento de ecuaciones diferenciales parciales en paralelo.

Idealmente, el objetivo principal en la investigación de los métodos de descomposición de dominio es producir algoritmos capaces de obtener *la solución global por la resolución de problemas locales exclusivamente*, pero hasta ahora, esto solo ha sido una inspiración, que es, un fuerte deseo para lograr dichas propiedades y por eso lo llamamos *el paradigma DDM*. En la anterior década, los algoritmos competitivos de descomposición de dominio desarrollados han sido los métodos preconditionados sin traslape y necesariamente incorporan restricciones, los cuales poseen un reto adicional para el paradigma DDM.

En el presente trabajo introducimos un grupo de ocho algoritmos, de los cuales, cuatro son preconditionados, a los que nos referiremos como los *algoritmos DVS* (véase [36], [38]), los cuales satisfacen el paradigma DDM. De ellos se derivan los eficientes y bien conocidos algoritmos BDDC y FETI-DP, los cuales fueron incorporados en un nuevo marco de trabajo; el método de descomposición de dominio en el espacio de vectores derivados (DVS).

Así, para poner en perspectiva nuestros desarrollos, el presente trabajo se inicio con una revisión de las formulaciones Dirichlet-Dirichlet y Neumann-Neumann a nivel continuo, para después dar paso a la derivación del método de descomposición de dominio en el espacio de vectores derivados y se muestra como este esquema es puesto dentro de las formulaciones continuas, completando así el modelo matemático del esquema DVS.

Con el modelo matemático del esquema DVS derivado, se procede ha

presentar la formulación numérica -que esta estrechamente ligada a la implementación computacional- y detallar las características de la implementación computacional tanto en su forma secuencial como paralela; y mediante ejemplos numéricos, se realiza el análisis y discusión de resultados (véase [39]) para algunos problemas emblemáticos que generan sistemas algebraicos simétricos, no simétricos e indefinidos. Finalmente, en la presente sección se dan las conclusiones de los logros alcanzados en este trabajo y se esboza lo que se considera pueden ser sus perspectivas.

18.1 Conclusiones

La formulación del método de descomposición de dominio en espacio de vectores derivados (véase [36], [38]), es un esquema que involucra 8 algoritmos, este es un marco unificador de dos de las formulaciones algebraicas más comúnmente usados en los métodos de descomposición de dominio sin traslapes, estos son: Finite Element Tearing and Interconnect Dual-Primal (FETI-DP) y Balancing Domain Decomposition by Constraints (BDDC) (véase [18], [42], [43], [57], [67], [68], [70] y [72]).

Un breve y efectivo resumen de los ocho métodos de descomposición de dominio sin traslape en el espacio de vectores derivados, de los cuales cuatro son formulaciones primales y los otros cuatro son formulaciones duales, es dado a continuación:

1. Las formulaciones no preconditionadas son:

- Formulación Dirichlet-Dirichlet

$$\left\{ \underline{a} \underline{S} u_{\Delta} = \underline{f}_{\Delta} \quad \text{y} \quad \underline{j} \underline{u}_{\Delta} = 0, \quad (\text{PRIMAL\#1}) \quad (18.1) \right.$$

- Formulación Neumann-Neumann

$$\left\{ \begin{array}{l} \underline{j} \underline{S}^{-1} \underline{\lambda}_{\Delta} = \underline{j} \underline{S}^{-1} \underline{f}_{\Delta} \quad \text{y} \quad \underline{a} \underline{\lambda}_{\Delta} = 0, \quad (\text{DUAL\#1}) \\ \underline{S}^{-1} \underline{j} \underline{v}_{\Delta} = \underline{S}^{-1} \underline{j} \underline{S}^{-1} \underline{f}_{\Delta} \quad \text{y} \quad \underline{a} \underline{S} \underline{v}_{\Delta} = 0, \quad (\text{PRIMAL\#2}) \\ \underline{S} \underline{a} \underline{\mu}_{\Delta} = \underline{S} \underline{a} \underline{j} \underline{S}^{-1} \underline{f}_{\Delta} \quad \text{y} \quad \underline{j} \underline{S}^{-1} \underline{\mu}_{\Delta} = 0, \quad (\text{DUAL\#2}) \end{array} \right. \quad (18.2)$$

2. Las formulaciones preconditionadas son:

- Formulación Dirichlet-Dirichlet DVS BDDC

$$\underline{\underline{a}}S^{-1}\underline{\underline{a}}S\underline{\underline{u}}_{\Delta} = \underline{\underline{a}}S^{-1}\underline{\underline{f}}_{\Delta} \quad \text{y} \quad \underline{\underline{j}}\underline{\underline{u}}_{\Delta} = 0, \quad (\text{PRIMAL\#1}) \quad (18.3)$$

- Formulación Neumann-Neumann DVS FETI-DP

$$\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}S^{-1}\underline{\underline{\lambda}}_{\Delta} = \underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}S^{-1}\underline{\underline{f}}_{\Delta} \quad \text{y} \quad \underline{\underline{a}}\underline{\underline{\lambda}}_{\Delta} = 0, \quad (\text{DUAL\#1})$$

donde $\underline{\underline{u}}_{\Delta} = \underline{\underline{a}}S^{-1}\left(\underline{\underline{f}}_{\Delta} - \underline{\underline{j}}\underline{\underline{\lambda}}_{\Delta}\right)$ (18.4)

- Formulación Neumann-Neumann DVS-PRIMAL

$$\underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}v_{\Delta} = \underline{\underline{S}}^{-1}\underline{\underline{j}}\underline{\underline{S}}\underline{\underline{j}}S^{-1}\underline{\underline{f}}_{\Delta} \quad \text{y} \quad \underline{\underline{a}}Sv_{\Delta} = 0, \quad (\text{PRIMAL\#2})$$

donde $\underline{\underline{u}}_{\Delta} = \underline{\underline{a}}S^{-1}\left(\underline{\underline{f}}_{\Delta} - \underline{\underline{j}}\underline{\underline{S}}v_{\Delta}\right)$ (18.5)

- Formulación Neumann-Neumann DVS-DUAL

$$\underline{\underline{S}}aS^{-1}\underline{\underline{a}}\underline{\underline{\mu}}_{\Delta} = \underline{\underline{S}}aS^{-1}\underline{\underline{a}}\underline{\underline{j}}\underline{\underline{S}}S^{-1}\underline{\underline{f}}_{\Delta} \quad \text{y} \quad \underline{\underline{j}}S^{-1}\underline{\underline{\mu}}_{\Delta} = 0, \quad (\text{DUAL\#2})$$

donde $\underline{\underline{u}}_{\Delta} = \underline{\underline{a}}S^{-1}\left(\underline{\underline{f}}_{\Delta} + \underline{\underline{\mu}}_{\Delta}\right)$ (18.6)

El paradigma DVS, constituido por 8 algoritmos con características similares, a los cuales nos hemos referido como los algoritmos DVS. Donde cada uno de los algoritmos preconditionados posee las siguientes conspicuas características:

- Las formulaciones Dual y Primal de dos de los métodos comúnmente usados -FETI-DP y BDDC- han sido derivadas de una manera unificada. El esquema desarrollado incluye formulaciones algebraicas para matrices simétricas, no simétricas e indefinidas -i.e. no positivas y no negativas definidas-. Además se detallan las condiciones que tales matrices deben de satisfacer para que los algoritmos generales sean aplicables.
- El esquema DVS permite aplicar técnicas de descomposición de dominio directamente al sistema de matrices que son obtenidas después de que la ecuación diferencial o sistema de tales ecuaciones han sido discretizadas. La aplicación de tales procedimientos no requieren del conocimiento acerca de la ecuación diferencial que originó las matrices.

- Los algoritmos tienen una aplicabilidad general, ya que ellos pueden ser aplicables a problemas de valor en la frontera asociados a una sola ecuación diferencial o a sistemas de ecuaciones.
- Para cada uno de los algoritmos, se han desarrollado formulaciones explícitas en términos de matrices; ellos están dados en las Ecs. (13.4, 13.21, 13.30, 13.43, 13.11, 13.29, 13.36 y 13.50).
- Tales formulaciones permiten desarrollar códigos que satisfacen el paradigma DDM, i.e. *en el cual la solución del problema global es obtenida exclusivamente por resolución de problemas locales*.
- Son preconditionados y con restricciones. En el caso de matrices no simétricas, la eficiencia numérica de los algoritmos preconditionados están en el mismo orden como los algoritmos de descomposición de dominio del estado del arte (véase [34] y [35]).

Además, hay varias propiedades numéricas y computacionales de los algoritmos DVS que destacan, entre las más importantes se tienen:

- El código es independiente de la geometría.
- El código que se obtiene es robusto, ya que con ligeras modificaciones es aplicado a problemas en dos y tres dimensiones; además soportan matrices simétricas, no simétricas e indefinidas
- El mismo código puede ser aplicado a una sola ecuación elíptica y a sistemas de ecuaciones de elasticidad lineal, con cambios mínimos en comportamientos específicos.
- Los algoritmos desarrollados son paralelizables y escalables con una alta eficiencia computacional.
- El código soporta diferentes métodos de solución de sistemas lineales -directos e iterativos- en los subdominios.
- El algoritmo global es débilmente acoplado a los subdominios.
- El desarrollo del código orientado a objetos simplifica la programación, permitiendo que a partir de la implementación secuencial se genere la implementación paralela con un mínimo de cambios, con la consecuente facilidad de expansión y mantenimiento del código.

- Los códigos desarrollados fueron aplicados para resolver diversos problemas con valores en la frontera que existen en el modelado de ciertos fenómenos geofísicos, tales como el transporte de solutos en fluidos libres como en fluidos en medios porosos. También presentamos resultados para problemas de elasticidad estática, de este modo ilustramos la aplicación de los algoritmos desarrollados a sistemas de ecuaciones diferenciales.

Por otro lado, las formulaciones FETI-DP y BDDC son óptimas en el sentido de que el número de condicionamiento κ de estos problemas de interfaces convergen asintóticamente como (véase [27], [46] y [49])

$$\kappa = O(1 + \log^2(H/h)) \quad (18.7)$$

-los términos H y h son tomados según definiciones de la sección (12.1)-nuestras formulaciones de los algoritmos DVS FETI-DP y DVS BDDC en el espacio de vectores derivados muestran un desempeño similar cuando usan el mismo conjunto de restricciones primales.

Estas propiedades hacen de los algoritmos DVS muy adecuados como herramientas usadas en la construcción de Software masivamente paralelizables, necesario para ser usada en la programación de las computadoras paralelas de alto desempeño disponibles actualmente³³.

18.2 Trabajo Futuro

De los algoritmos desarrollados, dos hasta donde se tiene conocimiento son totalmente diferentes a cualquiera de los reportados anteriormente y deben ser motivo de investigaciones futuras, además de que los métodos que se mostraron hasta ahora sólo se ha aplicado a problemas elípticos escalares y se inicia a problemas vectoriales, pero es posible aplicarlos a problemas parabólicos escalares y vectoriales tanto lineales como no lineales, por ello el trabajo futuro puede ser esbozado como:

- Hacer una investigación sobre nuestros métodos no reportados en la literatura.

³³Una versión de los códigos desarrollados de los algoritmos DVS está en línea en la página WEB <http://www.mmc.geofisica.unam.mx/acl/DVS/>

- Aplicar los métodos desarrollados y ampliar el código para soportar:
 1. Problemas elípticos con condiciones de frontera tipo Robin.
 2. Problemas parabólicos escalares y vectoriales.
 3. Problemas elípticos y parabólicos no lineales.
- Implementar un mecanismo computacional que permita definir ecuaciones o sistemas de ecuaciones, sus parámetros y condiciones de frontera para que el código sea independiente de ellas.

En cuanto a la implementación computacional de los métodos desarrollados, la paralelización se realiza mediante el paso de mensajes usando la biblioteca MPI en C++ para interconectar cada uno de los Cores del Cluster. Pero es posible usar eficientemente la interconexión de memoria compartida de los actuales equipos de cómputo mediante OpenMP y además usar los cada vez más numerosos Cores CUDA -que una sola tarjeta NVIDIA TESLA puede tener del orden de cientos ellos-.

Los métodos de descomposición de dominio sin traslape y en particular el esquema DVS implementado con sus ocho algoritmos, pueden hacer uso de esta forma integradora de paralelismo. Para ello, la interconexión de los equipos de memoria compartida, se realizaría mediante MPI y en cada equipo de memoria compartida se manipularían uno o más subdominios mediante OpenMP -al ser cada subdominio independiente de los demás- y la manipulación de matrices y operaciones entre matrices y vectores que requiere cada subdominio se realizarían en las tarjetas NVIDIA mediante los numerosos Cores CUDA.

Donde la integración de esta forma de paralelismo en el código, es un paso natural, que involucra hacer cambios mínimos al código, al ser necesario sólo cambiar los comportamientos locales que requieren otro tipo de paralelismo al implementado actualmente, ya que la jerarquía de clases del código desarrollado permite especializar los comportamientos que implementan las comunicaciones y esto queda de manifiesto al reutilizar toda la jerarquía de clases de la implementación secuencial en la paralela.

Esto permitiría tener códigos reutilizables en distintas arquitecturas de cómputo paralelo, además de hacer uso de equipos de cómputo cada vez más asequibles y de menor costo, pero con una creciente eficiencia computacional que pueden competir con los grandes equipos de cómputo de alto desempeño.

19 Apéndice A: Consideraciones Sobre la Formulación Numérica y su Implementación Computacional

Para realizar la implementación de cada uno de los métodos desarrollados de descomposición de dominio en el espacio de vectores derivados (DVS) -véase sección (15.2)-, es necesario trabajar con los operadores \underline{a} , \underline{j} , \underline{S} y \underline{S}^{-1} , así como realizar las operaciones involucradas entre ellos.

Normalmente los operadores \underline{S} y \underline{S}^{-1} no se construyen -son operadores virtuales- porque su implementación generaría matrices densas, las cuales consumen mucha memoria y hacen ineficiente su implementación computacional. En vez de eso, sólo se realizan las operaciones que involucra la definición del operador, por ejemplo $\underline{S}u_\Gamma$

$$\underline{S}u_\Gamma = \left(\underline{A}_{\Delta\Delta} - \underline{A}_{\Delta\Pi} \left(\underline{A}_{\Pi\Pi} \right)^{-1} \underline{A}_{\Pi\Delta} \right) u_\Gamma \quad (19.1)$$

en donde, para su evaluación sólo involucra la operación de multiplicación matriz-vector y resta de vectores, haciendo la evaluación computacional eficiente.

En el presente apéndice se desarrollan las operaciones que se requieren para implementar a cada uno los operadores involucrados en los métodos desarrollados y que matrices en cada caso son necesarias de construir, siempre teniendo en cuenta el hacer lo más eficiente posible su implementación y evaluación en equipos de cómputo de alto desempeño.

19.1 Matrices Virtuales y Susceptibles de Construir

La gran mayoría de las matrices usadas en los métodos de descomposición de dominio son operadores virtuales, por ejemplo el operador \underline{S} , que es definido como

$$\underline{S} = \underline{A}_{\Delta\Delta} - \underline{A}_{\Delta\Pi} \left(\underline{A}_{\Pi\Pi} \right)^{-1} \underline{A}_{\Pi\Delta} \quad (19.2)$$

y es formado por

$$\underline{S} = \sum_{\alpha=1}^E \underline{S}^\alpha \quad (19.3)$$

donde $\underline{\underline{S}}^\alpha$ a su vez está constituida por el complemento de Schur local al subdominio Ω_α

$$\underline{\underline{S}}^\alpha = \underline{\underline{A}}_{\Delta\Delta}^\alpha - \underline{\underline{A}}_{\Delta\Pi}^\alpha \left(\underline{\underline{A}}_{\Pi\Pi}^\alpha \right)^{-1} \underline{\underline{A}}_{\Pi\Delta}^\alpha \quad (19.4)$$

pero, de nuevo, las matrices locales $\underline{\underline{S}}^\alpha$ y $\left(\underline{\underline{A}}_{\Pi\Pi}^\alpha \right)^{-1}$ no se construyen ya que $\underline{\underline{A}}_{\Pi\Pi}^\alpha u_\Pi = \underline{\underline{S}}_\pi^\alpha u_\Pi$ donde $\underline{\underline{S}}_\pi^\alpha$ es definido como

$$\underline{\underline{S}}_\pi^\alpha = \underline{\underline{A}}_{\pi\pi}^\alpha - \underline{\underline{A}}_{\pi I}^\alpha \left(\underline{\underline{A}}_{II}^\alpha \right)^{-1} \underline{\underline{A}}_{I\pi}^\alpha. \quad (19.5)$$

Entonces, las matrices susceptibles de ser construidas en cada subdominio Ω_α son

$$\underline{\underline{A}}_{II}^\alpha, \underline{\underline{A}}_{I\pi}^\alpha, \underline{\underline{A}}_{I\Delta}^\alpha, \underline{\underline{A}}_{\pi I}^\alpha, \underline{\underline{A}}_{\pi\pi}^\alpha, \underline{\underline{A}}_{\pi\Delta}^\alpha, \underline{\underline{A}}_{\Delta I}^\alpha, \underline{\underline{A}}_{\Delta\pi}^\alpha \text{ y } \underline{\underline{A}}_{\Delta\Delta}^\alpha \quad (19.6)$$

pero $\underline{\underline{A}}_{I\pi}^\alpha = \left(\underline{\underline{A}}_{\pi I}^\alpha \right)^T$, $\underline{\underline{A}}_{I\Delta}^\alpha = \left(\underline{\underline{A}}_{\Delta I}^\alpha \right)^T$ y $\underline{\underline{A}}_{\pi\Delta}^\alpha = \left(\underline{\underline{A}}_{\Delta\pi}^\alpha \right)^T$, así, las únicas matrices susceptibles de construir son

$$\underline{\underline{A}}_{II}^\alpha, \underline{\underline{A}}_{I\pi}^\alpha, \underline{\underline{A}}_{I\Delta}^\alpha, \underline{\underline{A}}_{\pi\pi}^\alpha, \underline{\underline{A}}_{\pi\Delta}^\alpha \text{ y } \underline{\underline{A}}_{\Delta\Delta}^\alpha \quad (19.7)$$

donde todas ellas son locales a cada subdominio Ω_α , con una estructura acorde al tipo de discretización discutida en las secciones (12.5 y 15.1) y en este apéndice.

Por lo anterior, nótese que el operador $\underline{\underline{S}}^{-1}$ tampoco se construye, para evaluar $\underline{\underline{S}}^{-1} u_\Gamma$ se usa el procedimiento indicado en la sección (19.3).

Otros operadores como son las matrices $\underline{\underline{a}}$ y $\underline{\underline{j}}$ pueden ser o no construidos, pero es necesario recordar que $\underline{\underline{j}} = \underline{\underline{I}} - \underline{\underline{a}}$, así que, en principio $\underline{\underline{j}}$ no sería necesario construir, por otro lado los operadores $\underline{\underline{a}}$ y $\underline{\underline{j}}$ sólo existen en el nodo maestro -donde se controla a los nodos duales y primales y no en los subdominios-; y estas matrices en caso de ser construidas serán dispersas, con pocos valores por renglón -según el número de subdominios que compartan a cada uno de los nodos de la frontera interior- y están sujetas al tipo de triangulación usada en la descomposición de la malla gruesa del dominio.

19.2 Evaluación de la Matriz $\underline{\underline{S}}$ con Nodos Primales Definidos

En todos los casos donde aparece el operador $\underline{\underline{S}}$, ya que, sólo interesa la evaluación de $\underline{\underline{S}}y_\Gamma$, i.e. $\underline{\underline{v}}_\Gamma = \underline{\underline{S}}y_\Gamma$, entonces se reescribe al operador $\underline{\underline{S}}$ en

términos de sus componentes por subdominio

$$\underline{u}_\Gamma = \left[\sum_{\alpha=1}^E \underline{S}^\alpha \right] \underline{y}_\Gamma \quad (19.8)$$

para evaluar, el vector \underline{y}_Γ se descompone en los subvectores $\underline{y}_\Gamma^\alpha$ correspondientes a cada subdominio Ω_α . Así, para evaluar $\tilde{\underline{u}}_\Gamma^\alpha = \underline{S}^\alpha \underline{y}_\Gamma^\alpha$ se usa el hecho de que

$$\underline{S}^\alpha = \underline{A}_{\Delta\Delta}^\alpha - \underline{A}_{\Delta\Pi}^\alpha \left(\underline{A}_{\Pi\Pi}^\alpha \right)^{-1} \underline{A}_{\Pi\Delta}^\alpha \quad (19.9)$$

en donde, se tiene que evaluar

$$\tilde{\underline{u}}_\Gamma^\alpha = \left(\underline{A}_{\Delta\Delta}^\alpha - \underline{A}_{\Delta\Pi}^\alpha \left(\underline{A}_{\Pi\Pi}^\alpha \right)^{-1} \underline{A}_{\Pi\Delta}^\alpha \right) \underline{y}_\Gamma^\alpha \quad (19.10)$$

estas evaluaciones en cada subdominio Ω_α pueden realizarse en paralelo.

Para evaluar de forma eficiente esta expresión, se realizan las siguientes operaciones equivalentes

$$\begin{aligned} \underline{x1} &= \underline{A}_{\Delta\Delta}^\alpha \underline{y}_\Gamma^\alpha & (19.11) \\ \underline{x2} &= \left(\underline{A}_{\Delta\Pi}^\alpha \left(\underline{A}_{\Pi\Pi}^\alpha \right)^{-1} \underline{A}_{\Pi\Delta}^\alpha \right) \underline{y}_\Gamma^\alpha \\ \tilde{\underline{u}}_\Gamma^\alpha &= \underline{x1} - \underline{x2} \end{aligned}$$

la primera y tercera expresión no tienen ningún problema en su evaluación, para la segunda expresión se tiene que hacer

$$\underline{x3} = \underline{A}_{\Pi\Delta}^\alpha \underline{y}_\Gamma^\alpha \quad (19.12)$$

con este resultado intermedio se debe calcular

$$\underline{x4} = \left(\underline{A}_{\Pi\Pi}^\alpha \right)^{-1} \underline{x3} \quad (19.13)$$

pero como no se cuenta con $\left(\underline{A}_{\Pi\Pi}^\alpha \right)^{-1}$ ya que sería una matriz densa, entonces se multiplica la expresión por $\underline{A}_{\Pi\Pi}^\alpha$ obteniendo

$$\underline{A}_{\Pi\Pi}^\alpha \underline{x4} = \underline{A}_{\Pi\Pi}^\alpha \left(\underline{A}_{\Pi\Pi}^\alpha \right)^{-1} \underline{x3} \quad (19.14)$$

al simplificar, se obtiene

$$\underline{\underline{A}}_{\text{III}}^{\alpha} \underline{x4} = \underline{x3}. \quad (19.15)$$

Esta última expresión puede ser resuelta usando Gradiente Conjugado o alguna variante de GMRES. Una vez obtenido $\underline{x4}$, se puede calcular

$$\underline{x2} = \underline{\underline{A}}_{\Delta\text{II}}^{\alpha} \underline{x4} \quad (19.16)$$

así

$$\underline{\tilde{u}}_{\Gamma}^{\alpha} = \underline{x1} - \underline{x2} \quad (19.17)$$

completando la secuencia de operaciones necesaria para obtener $\underline{\tilde{u}}_{\Gamma}^{\alpha} = \underline{\underline{S}}^{\alpha} \underline{y}_{\Gamma}^{\alpha}$.

Observación 8 En el caso de la expresión dada por la Ec.(19.15) al aplicar un método iterativo, sólo interesará realizar el producto $\underline{\underline{A}}_{\text{III}}^{\alpha} \underline{x_{\text{II}}}$, o más precisamente $\underline{\underline{S}}_{\pi}^i \underline{x_{\pi}}$, donde

$$\underline{\underline{S}}_{\pi}^i = \left(\underline{\underline{A}}_{\pi\pi}^i - \underline{\underline{A}}_{\pi I}^i \left(\underline{\underline{A}}_{II}^i \right)^{-1} \underline{\underline{A}}_{I\pi}^i \right) \quad (19.18)$$

entonces si se llama $\underline{x_{\pi}}^i$ al vector correspondiente al subdominio i , se tiene

$$\underline{\tilde{u}}_{\pi}^i = \left(\underline{\underline{A}}_{\pi\pi}^i - \underline{\underline{A}}_{\pi I}^i \left(\underline{\underline{A}}_{II}^i \right)^{-1} \underline{\underline{A}}_{I\pi}^i \right) \underline{x_{\pi}}^i \quad (19.19)$$

para evaluar de forma eficiente esta expresión, se realizan las siguientes operaciones equivalentes

$$\underline{u1} = \underline{\underline{A}}_{\pi\pi}^i \underline{x_i} \quad (19.20)$$

$$\underline{u2} = \left(\underline{\underline{A}}_{\pi I}^i \left(\underline{\underline{A}}_{II}^i \right)^{-1} \underline{\underline{A}}_{I\pi}^i \right) \underline{x_i}$$

$$\underline{\tilde{u}}_{\Gamma}^i = \underline{u1} - \underline{u2}$$

la primera y tercera expresión no tienen ningún problema en su evaluación, para la segunda expresión se tiene que hacer

$$\underline{u3} = \underline{\underline{A}}_{I\pi}^i \underline{x_i} \quad (19.21)$$

con este resultado intermedio se debe calcular

$$\underline{u3} = \left(\underline{\underline{A}}_{II}^i \right)^{-1} \underline{u4} \quad (19.22)$$

pero como no se cuenta con $\left(\underline{\underline{A}}_{II}^i\right)^{-1}$, entonces se multiplica la expresión por $\underline{\underline{A}}_{II}^i$ obteniendo

$$\underline{\underline{A}}_{II}^i \underline{u3} = \underline{\underline{A}}_{II}^i \left(\underline{\underline{A}}_{II}^i\right)^{-1} \underline{u4} \quad (19.23)$$

al simplificar, se obtiene

$$\underline{\underline{A}}_{II}^i \underline{u4} = \underline{u3}. \quad (19.24)$$

Esta última expresión puede ser resuelta usando métodos directos -Factorización LU o Cholesky- o iterativos -Gradiente Conjugado o alguna variante de GMRES- cada una de estas opciones tiene ventajas y desventajas desde el punto de vista computacional -como el consumo de memoria adicional o el aumento de tiempo de ejecución por las operaciones involucradas en cada caso- que deben ser evaluadas al momento de implementar el código para un problema particular. Una vez obtenido $\underline{u4}$, se puede calcular

$$\underline{u2} = \underline{\underline{A}}_{\pi I}^i \underline{u4} \quad (19.25)$$

así

$$\tilde{u}_{\Gamma}^i = \underline{u1} - \underline{u2} \quad (19.26)$$

completando la secuencia de operaciones necesaria para obtener $\underline{\underline{S}}_{\pi}^i x_i$.

19.3 Evaluación de la Matriz $\underline{\underline{S}}^{-1}$ con Nodos Primitives Definidos

En los algoritmos desarrollados interviene el cálculo de $\underline{\underline{S}}^{-1}$, dado que la matriz $\underline{\underline{S}}$ no se construye, entonces la matriz $\underline{\underline{S}}^{-1}$ tampoco se construye. En lugar de ello, se procede de la siguiente manera: Se asume que en las operaciones anteriores al producto de $\underline{\underline{S}}^{-1}$, se ha obtenido un vector. Supóngase que es \underline{v}_{Γ} , entonces para hacer

$$\underline{u}_{\Gamma} = \underline{\underline{S}}^{-1} \underline{v}_{\Gamma} \quad (19.27)$$

se procede a multiplicar por $\underline{\underline{S}}$ a la ecuación anterior

$$\underline{\underline{S}} \underline{u}_{\Gamma} = \underline{\underline{S}} \underline{\underline{S}}^{-1} \underline{v}_{\Gamma} \quad (19.28)$$

obteniendo

$$\underline{\underline{S}} \underline{u}_{\Gamma} = \underline{v}_{\Gamma} \quad (19.29)$$

es decir, usando algún proceso iterativo -como CGM, GMRES- se resuelve el sistema anterior, de tal forma que en cada iteración de \underline{u}_Γ^i se procede como se indicó en la sección del cálculo de \underline{S} , resolviendo $\underline{S}\underline{u}_\Gamma = \underline{v}_\Gamma$ mediante iteraciones de $\underline{u}_\Gamma^{i+1} = \underline{S}\underline{u}_\Gamma^i$.

19.4 Cálculo de los Nodos Interiores

La evaluación de

$$\underline{u}_\Pi = - \left(\underline{A}_{\Pi\Pi} \right)^{-1} \underline{A}_{\Pi\Delta} \underline{u}_\Delta \quad (19.30)$$

involucra de nuevo cálculos locales de la expresión

$$\underline{u}_I^\alpha = - \left(\underline{A}_{\Pi\Pi}^\alpha \right)^{-1} \underline{A}_{\Pi\Delta}^\alpha \underline{u}_\Gamma^\alpha \quad (19.31)$$

aquí otra vez está involucrado $\left(\underline{A}_{\Pi\Pi}^\alpha \right)^{-1}$, por ello se debe usar el siguiente procedimiento para evaluar de forma eficiente esta expresión, realizando las operaciones equivalentes

$$\begin{aligned} \underline{x}_4 &= \underline{A}_{\Pi\Delta}^\alpha \underline{u}_\Gamma^\alpha \\ \underline{u}_I^\alpha &= \left(\underline{A}_{\Pi\Pi}^\alpha \right)^{-1} \underline{x}_4 \end{aligned} \quad (19.32)$$

multiplicando por $\underline{A}_{\Pi\Pi}^\alpha$ la última expresión, se obtiene

$$\underline{A}_{\Pi\Pi}^\alpha \underline{u}_I^\alpha = \underline{A}_{\Pi\Pi}^\alpha \left(\underline{A}_{\Pi\Pi}^\alpha \right)^{-1} \underline{x}_4 \quad (19.33)$$

simplificando, se obtiene

$$\underline{A}_{\Pi\Pi}^\alpha \underline{u}_I^\alpha = \underline{x}_4 \quad (19.34)$$

esta última expresión puede ser resuelta usando métodos directos -como Factorización LU o Cholesky- o mediante métodos iterativos -como Gradiente Conjugado o alguna variante de GMRES- como se indica en la observación (8).

19.5 Descomposición de Schur sin Nodos Primitives

En el caso de que en la descomposición no se usen nodos primales, la matriz virtual global \underline{A} queda como

$$\underline{\underline{A}} = \begin{pmatrix} \underline{\underline{A}}_{II}^1 & \underline{\underline{0}} & \cdots & \underline{\underline{0}} & \underline{\underline{A}}_{I\Delta}^1 \\ \underline{\underline{0}} & \underline{\underline{A}}_{II}^2 & \cdots & \underline{\underline{0}} & \underline{\underline{A}}_{I\Delta}^2 \\ \underline{\underline{0}} & \underline{\underline{0}} & \ddots & \underline{\underline{0}} & \vdots \\ \underline{\underline{0}} & \underline{\underline{0}} & \cdots & \underline{\underline{A}}_{II}^E & \underline{\underline{A}}_{I\Delta}^E \\ \underline{\underline{A}}_{\Delta I}^1 & \underline{\underline{A}}_{\Delta I}^2 & \cdots & \underline{\underline{A}}_{\Delta I}^E & \underline{\underline{A}}_{\Delta\Delta} \end{pmatrix} \quad (19.35)$$

de donde $\underline{\underline{A}}x = \underline{b}$ se implementa como

$$\underline{\underline{A}} \begin{pmatrix} \underline{x}_I \\ \underline{x}_\Delta \end{pmatrix} = \begin{pmatrix} \underline{b}_I \\ \underline{b}_\Delta \end{pmatrix} \quad (19.36)$$

i.e.

$$\left(\sum_{\alpha=1}^E \left(\underline{\underline{A}}_{\Delta\Delta}^\alpha - \underline{\underline{A}}_{\Delta I}^\alpha \left(\underline{\underline{A}}_{II}^\alpha \right)^{-1} \underline{\underline{A}}_{I\Delta}^\alpha \right) \right) \underline{x}_\Delta = \underline{b}_\Delta - \sum_{\alpha=1}^E \left(\underline{\underline{A}}_{\Delta I}^\alpha \left(\underline{\underline{A}}_{II}^\alpha \right)^{-1} \underline{b}_I^\alpha \right) \quad (19.37)$$

una vez encontrado \underline{x}_Δ , se encuentra \underline{x}_I mediante

$$\underline{x}_I^\alpha = \left(\underline{\underline{A}}_{II}^\alpha \right)^{-1} \left(\underline{b}_I^\alpha - \underline{\underline{A}}_{I\Delta}^\alpha \underline{x}_\Delta \right). \quad (19.38)$$

19.6 DVS para Ecuaciones Escalares y Vectoriales

Con el fin de ejemplificación, se supone una ecuación escalar en dos dimensiones definida en un dominio Ω , mediante una descomposición en subdominios usando una malla estructurada cartesiana como se muestra en la figura:

En este caso, sería una descomposición del dominio Ω en 3×3 y 6×5 , la malla gruesa sería descompuesta en $3 \times 3 = 9$ subdominios Ω_α y donde cada uno de ellos tiene una partición fina de 6×5 i.e. $42 = (6 + 1) * (5 + 1)$ grados de libertad por subdominio. En el caso vectorial, en el cual cada grado de libertad contiene C componentes, el número de grados de libertad total será igual a $((6 + 1) * (5 + 1) * C)$. Por ejemplo, en el caso de $C = 3$ se tienen 126 grados de libertad por subdominio (véase [45]).

EDP Vectoriales en Dos y Tres Dimensiones En el caso de una ecuación vectorial en dos -tres- dimensiones, si el dominio Ω es descompuesto en una malla estructurada cartesiana, donde la partición gruesa de $n \times m$ -ó $n \times m \times o$ - subdominios Ω_α y donde cada uno es descompuesto en una

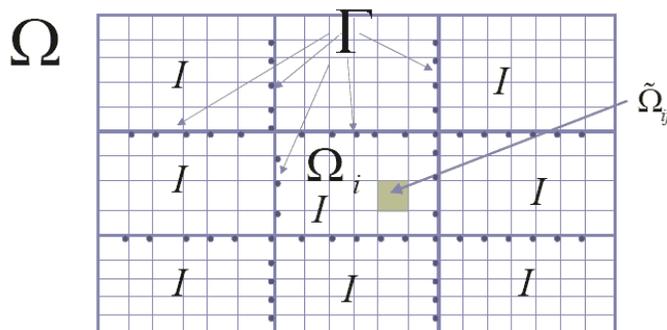


Figura 28: Dominio Ω descompuesto en una partición gruesa de 3×3 y cada subdominio Ω_i en una partición fina de 6×5 .

partición fina de $r \times s$ -ó $r \times s \times t$ -, en el cual cada grado de libertad contiene C componentes, el número de grados de libertad por subdominio es $(r + 1) * (s + 1) * C$ -ó $(r + 1) * (s + 1) * (t + 1) * C$ -.

A partir de la formulación de los métodos desarrollados, se generan las matrices locales en cada subdominio, de esta forma se obtiene la descomposición fina del dominio, generándose de manera virtual el sistema lineal definido por el método DVS que se este implementando.

La implementación computacional que se desarrolló tiene una jerarquía de clases en donde la clase *DPMethod* realiza la partición gruesa del dominio usando la clase *Interchange* y controla la partición de cada subdominio mediante un objeto de la clase de *RectSub* generando la partición fina del dominio. La resolución de los nodos de la frontera interior se hace mediante el método de CGM o alguna variante de GMRES.

El método de descomposición de dominio en el espacio de vectores derivados se implementó realizando las siguientes tareas:

- A) La clase *DPMethod* genera la descomposición gruesa del dominio mediante la agregación de un objeto de la clase *Interchange*, se supone que se tiene particionado en $n \times m$ -ó $n \times m \times o$ - subdominios.
- B) Con esa geometría se construyen los objetos de *RectSub* -uno por cada subdominio Ω_α -, donde cada subdominio es particionado

en $r \times s$ -ó $r \times s \times t$ - subdominios y se regresan las coordenadas de los nodos de frontera del subdominio correspondiente a la clase *DPMethod*.

C) Con estas coordenadas, la clase *DPMethod* conoce a los nodos de la frontera interior -son estos los que resuelve el método de descomposición de dominio-. Las coordenadas de los nodos de la frontera interior se dan a conocer a los objetos *RectSub*, transmitiendo sólo aquellos que están en su subdominio.

D) Después de conocer los nodos de la frontera interior, cada objeto *RectSub* calcula las matrices locales sin realizar comunicación alguna. Al terminar de calcular las matrices se avisa a la clase *DPMethod* de la finalización de los cálculos.

E) Mediante la comunicación de vectores del tamaño del número de nodos de la frontera interior entre la clase *DPMethod* y los objetos *RectSub*, se prepara todo lo necesario para empezar el método de CGM o GMRES y resolver el sistema lineal virtual.

F) Para aplicar el método de CGM o GMRES, en cada iteración se transmite un vector del tamaño del número de nodos de la frontera interior para que en cada objeto se realicen las operaciones pertinentes y resolver así el sistema algebraico asociado, esta comunicación se realiza de ida y vuelta entre la clase *DPMethod* y los objetos *RectSub* tantas veces como iteraciones haga el método. Resolviendo con esto los nodos de la frontera interior \underline{u}_{Γ_i} .

G) Al término de las iteraciones, se pasa la solución \underline{u}_{Γ_i} de los nodos de la frontera interior que pertenecen a cada subdominio dentro de cada objeto *RectSub* para que se resuelvan los nodos locales al subdominio $\underline{u}_{\pi}^{\alpha} = - \left(\underline{A}_{\text{III}}^{\alpha} \right)^{-1} \underline{A}_{\text{II}\Delta}^{\alpha} \underline{u}_{\Gamma}^{\alpha}$, sin realizar comunicación alguna en el proceso, al concluir se avisa a la clase *DDM* de ello.

I) La clase *DPMethod* mediante un último mensaje avisa que se concluya el programa, terminado así el esquema Maestro-Esclavo.

Análisis de Comunicaciones Para hacer un análisis de las comunicaciones en una ecuación vectorial con C componentes, entre el nodo principal

y los nodos esclavos en el método DVS es necesario conocer qué se trasmite y su tamaño, es por ello que en la medida de lo posible se detallan las comunicaciones existentes -hay que hacer mención que entre los nodos esclavos no hay comunicación alguna-.

Tomando la descripción del algoritmo detallado anteriormente, en donde se supuso una partición del dominio Ω con una malla estructurada cartesiana en $n \times m$ -ó $n \times m \times o$ en tres dimensiones- para la malla gruesa y $r \times s$ -ó $r \times s \times t$ - para la malla fina, las comunicaciones correspondientes a cada inciso son:

- A) El nodo maestro transmite 2 coordenadas en dos -en tres- dimensiones correspondientes a la delimitación del subdominio.
- B) $2 * (r + 1) * (s + 1) * C$ -ó $2 * (r + 1) * (s + 1) * (t + 1) * C$ - coordenadas transmite cada subdominio al nodo maestro.
- C) A lo más $n * m * 2 * (r + 1) * (s + 1) * C$ -ó $n * m * o * 2 * (r + 1) * (s + 1) * (t + 1) * C$ - coordenadas son las de los nodos de la frontera interior, y sólo aquellas correspondientes a cada subdominio son transmitidas por el nodo maestro a los subdominios en los esclavos siendo estas a lo más $2 * (n * m) * (r * s) * C$ -ó $2 * (n * m * o) * (r * s * t) * C$ - coordenadas.
- D) Sólo se envía un aviso de la conclusión del cálculo de las matrices.
- E) A lo más $2 * (r + 1) * (s + 1) * C$ -ó $2 * (r + 1) * (s + 1) * (t + 1) * C$ - coordenadas son transmitidas a los subdominios en los nodos esclavos desde el nodo maestro y los nodos esclavos transmiten al nodo maestro esa misma cantidad información.
- F) A lo más $2 * (r + 1) * (s + 1) * C$ -ó $2 * (r + 1) * (s + 1) * (t + 1) * C$ - coordenadas son transmitidas a los subdominios en los nodos esclavos y estos retornan un número igual al nodo maestro por iteración del método de CGM o alguna variante de GMRES.
- G) A lo más $2 * (r + 1) * (s + 1) * C$ -ó $2 * (r + 1) * (s + 1) * (t + 1) * C$ - valores de la solución de la frontera interior son transmitidas a los subdominios en los nodos esclavos desde el nodo maestro y cada objeto transmite un único aviso de terminación.
- I) El nodo maestro manda un aviso a cada subdominio en los nodos esclavos para concluir con el esquema.

En todos los casos, la transmisión se realiza mediante paso de arreglos de enteros y números de punto flotante que varían de longitud pero siempre son cantidades pequeñas de estos y se transmiten en forma de bloque, por ello las comunicaciones son eficientes.

EDP Escalares como un Caso Particular de EDP Vectoriales En el caso de trabajar con ecuaciones escalares en dos o tres dimensiones con una malla estructurada cartesiana, en vez de ecuaciones vectoriales, todo el análisis anterior continua siendo válido y sólo es necesario tomar el número de componentes C igual a uno, manteniéndose la estructura del código intacta.

Esto le da robustez al código, pues permite trabajar con problemas escalares y vectoriales con un pequeño cambio en la estructura del programa, permitiendo resolver una gama más grande de problemas.

Tamaño de las Matrices Locales. Ahora, si se supone que el dominio $\Omega \subset \mathbb{R}^3$ es descompuesto con una malla estructurada cartesiana en una partición gruesa de $n \times m \times o$ subdominios Ω_α y cada subdominio Ω_α es descompuesto en una partición fina de $r \times s \times t$, con número de componentes igual a C , entonces el número de grados de libertad por subdominio es $(r + 1) * (s + 1) * (t + 1) * C$. En la cual se usa un ordenamiento estándar en la numeración de los nodos dentro de cada subdominio

El número de nodos interiores es

$$N^I = (r - 1) * (s - 1) * (t - 1) \quad (19.39)$$

el número de nodos primales, suponiendo que se usa restricción en los vértices es

$$N^\pi = (n - 1) * (m - 1) * (o - 1) \quad (19.40)$$

y el número de nodos duales es a lo más

$$N^\Delta = 2 * [(r - 1) + (s - 1)] * (t - 1) \quad (19.41)$$

entonces se tiene que el tamaño y la estructura de cada una de las matrices locales

$$\underline{\underline{A}}_{II}^i, \underline{\underline{A}}_{I\pi}^i, \underline{\underline{A}}_{I\Delta}^i, \underline{\underline{A}}_{\pi I}^i, \underline{\underline{A}}_{\pi\pi}^i, \underline{\underline{A}}_{\pi\Delta}^i, \underline{\underline{A}}_{\Delta I}^i, \underline{\underline{A}}_{\Delta\pi}^i \text{ y } \underline{\underline{A}}_{\Delta\Delta}^i$$

generadas por la descomposición fina del subdominio será:

- $\underline{\underline{A}}_{II}^i$ es una matriz cuadrada de $N^I \times N^I$ nodos, con una estructura bandada

- $\underline{\underline{A}}_{I\pi}^i$ es una matriz rectangular de $N^I \times N^\pi$ nodos, con una estructura dispersa
- $\underline{\underline{A}}_{\pi I}^i$ es una matriz rectangular de $N^\pi \times N^I$ nodos, con una estructura dispersa y transpuesta de $\underline{\underline{A}}_{I\pi}^i$
- $\underline{\underline{A}}_{I\Delta}^i$ es una matriz rectangular de $N^I \times N^\Delta$ nodos, con una estructura dispersa
- $\underline{\underline{A}}_{\Delta I}^i$ es una matriz rectangular de $N^\Delta \times N^I$ nodos, con una estructura dispersa y transpuesta de $\underline{\underline{A}}_{I\Delta}^i$
- $\underline{\underline{A}}_{\pi\Delta}^i$ es una matriz rectangular de $N^\pi \times N^\Delta$ nodos, con una estructura dispersa
- $\underline{\underline{A}}_{\Delta\pi}^i$ es una matriz rectangular de $N^\pi \times N^\Delta$ nodos, con una estructura dispersa y transpuesta de $\underline{\underline{A}}_{\pi\Delta}^i$
- $\underline{\underline{A}}_{\pi\pi}^i$ es una matriz cuadrada de $N^\pi \times N^\pi$ nodos, con una estructura bandada
- $\underline{\underline{A}}_{\Delta\Delta}^i$ es una matriz cuadrada de $N^\Delta \times N^\Delta$ nodos, con una estructura bandada

Para información sobre la estructura de las matrices bandadas véase la sección (20.2) y para matrices dispersas véase la sección (20.3).

19.7 Método de Descomposición de Dominio de Subestructuración

La solución numérica por los esquemas tradicionales de discretización -tipo Elemento Finito y Diferencias Finitas- generan una discretización del problema, la cual es usada para generar un sistema de ecuaciones algebraicos $\underline{\underline{A}}u = \underline{\underline{b}}$. Este sistema algebraico en general es de gran tamaño para problemas reales.

En esta sección y sin pérdida de generalidad se considerarán problemas con valor en la frontera (VBVP) de la forma

$$\begin{aligned} \mathcal{L}u &= f \quad \text{en } \Omega \\ u &= g \quad \text{sobre } \partial\Omega \end{aligned} \tag{19.42}$$

donde

$$\mathcal{L}u = -\nabla \cdot \underline{\underline{a}} \cdot \nabla u + cu \quad (19.43)$$

con $\underline{\underline{a}}$ una matriz positiva definida, simétrica y $c \geq 0$, como un caso particular del operador elíptico de orden 2 y para ejemplificar se toma un dominio $\Omega \subset R^2$ con fronteras poligonales, es decir, Ω es un conjunto abierto acotado y conexo tal que su frontera $\partial\Omega$ es la unión de un número finito de polígonos.

Si multiplico a la ecuación $-\nabla \cdot \underline{\underline{a}} \cdot \nabla u + cu = f_\Omega$ por $v \in V = H_0^1(\Omega)$, se obtiene

$$-v(\nabla \cdot \underline{\underline{a}} \cdot \nabla u + cu) = vf_\Omega \quad (19.44)$$

aplicando el teorema de Green se obtiene

$$\int_{\Omega} (\nabla v \cdot \underline{\underline{a}} \cdot \nabla u + cuv) d\mathbf{x} = \int_{\Omega} vf_\Omega d\mathbf{x}. \quad (19.45)$$

Definiendo el operador bilineal

$$a(u, v) = \int_{\Omega} (\nabla v \cdot \underline{\underline{a}} \cdot \nabla u + cuv) d\mathbf{x} \quad (19.46)$$

y la funcional lineal

$$l(v) = \langle f, v \rangle = \int_{\Omega} vf_\Omega d\mathbf{x} \quad (19.47)$$

entonces se puede reescribir el problema en forma variacional, haciendo uso de la forma bilineal $a(\cdot, \cdot)$ y la funcional lineal $l(\cdot)$.

Donde las funciones lineales definidas por tramos en Ω_e en este caso serán polinomios de orden uno en cada variable de forma separada y cuya restricción de ϕ_i a Ω_e es $\phi_i^{(e)}$. Para simplificar los cálculos en esta etapa, se supone que la matriz $\underline{\underline{a}} = a \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, entonces se tiene que la integral del lado izquierdo de la Ec.(5.32) queda escrita como

$$\int_{\Omega} (a\nabla\phi_i \cdot \nabla\phi_j + c\phi_i\phi_j) dx dy = \int_{\Omega} f_\Omega\phi_j dx dy \quad (19.48)$$

donde

$$\begin{aligned}
 K_{ij} &= \int_{\Omega} (a \nabla \phi_i \cdot \nabla \phi_j + c \phi_i \phi_j) \, dx dy & (19.49) \\
 &= \sum_{e=1}^E \int_{\Omega_e} (a \nabla \phi_i^{(e)} \cdot \nabla \phi_j^{(e)} + c \phi_i^{(e)} \phi_j^{(e)}) \, dx dy \\
 &= \sum_{e=1}^E \int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) \, dx dy
 \end{aligned}$$

y el lado derecho como

$$\begin{aligned}
 F_j &= \int_{\Omega} f_{\Omega} \phi_j \, dx dy & (19.50) \\
 &= \sum_{e=1}^E \int_{\Omega_e} f_{\Omega} \phi_j^{(e)} \, dx dy.
 \end{aligned}$$

Se considera el problema dado por la Ec.(19.42) en el dominio Ω , el cual es subdividido en E subdominios Ω_i , $i = 1, 2, \dots, E$ sin traslape, también conocida como malla gruesa \mathcal{T}_H , es decir

$$\Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j \quad \text{y} \quad \bar{\Omega} = \bigcup_{i=1}^E \bar{\Omega}_i, \quad (19.51)$$

y al conjunto

$$\Gamma = \bigcup_{i=1}^E \Gamma_i, \quad \text{si } \Gamma_i = \partial \Omega_i \setminus \partial \Omega \quad (19.52)$$

lo llamo la frontera interior del dominio Ω .

Un ejemplo de un dominio Ω y su descomposición en subdominios Ω_i y cada Ω_i a su vez descompuesto en Ω_e subdominios. Sin pérdida de generalidad se toma $g = 0$ en $\partial \Omega$, nótese que siempre es posible poner el problema de la Ec.(19.42) como uno con condiciones de frontera Dirichlet que se nulifiquen mediante la adecuada manipulación del término del lado derecho de la ecuación.

Primeramente sea $D \subset H_0^1(\Omega)$ un espacio lineal de funciones de dimensión finita N , en el cual esté definido un producto interior denotado para cada $u, v \in D$ por

$$u \cdot v = \langle u, v \rangle. \quad (19.53)$$

Considerando la existencia de los subconjuntos linealmente independientes

$$\begin{aligned} \mathcal{B} &\subset \tilde{D}, \mathcal{B}_I \subset \tilde{D}_I, \mathcal{B}_\Gamma \subset \tilde{D}_\Gamma \\ \mathcal{B}_\Gamma &\subset \tilde{D}_\Gamma, \mathcal{B}_{\Gamma J} \subset \tilde{D}_{\Gamma 1}, \mathcal{B}_{\Gamma M} \subset \tilde{D}_{\Gamma 2} \end{aligned} \quad (19.54)$$

los cuales satisfacen

$$\mathcal{B} = \mathcal{B}_I \cup \mathcal{B}_\Gamma \text{ y } \bar{\mathcal{B}}_\Gamma = \mathcal{B}_{\Gamma J} \cup \mathcal{B}_{\Gamma M} \quad (19.55)$$

el espacio generado por cada uno de los subconjuntos \mathcal{B}_Γ y $\bar{\mathcal{B}}_\Gamma$ es \tilde{D}_Γ , sin embargo, nótese la propiedad de que los miembros de \mathcal{B}_Γ tienen soporte local.

Se definen las bases

$$\mathcal{B}_I = \{w_I^1, \dots, w_I^{\bar{N}_I}\}, \mathcal{B}_{\Gamma M} = \{w_M^1, \dots, w_M^{\bar{N}_M}\} \text{ y } \mathcal{B}_{\Gamma J} = \{w_J^1, \dots, w_J^{\bar{N}_J}\} \quad (19.56)$$

de las funcionales lineales ϕ_i en Ω .

Entonces definiendo para toda $\delta = 1, \dots, K$, la matriz de $N_\delta \times N_\delta$

$$\underline{\underline{A}}_{II}^\delta \equiv [\langle w_I^i, w_I^j \rangle] \quad (19.57)$$

que sólo esta definida en cada subespacio (subdominio Ω_δ). Entonces, la matriz virtual $\underline{\underline{A}}_{II}$ es dada por la matriz diagonal de la forma

$$\underline{\underline{A}}_{II} \equiv \begin{bmatrix} \underline{\underline{A}}_{II}^1 & & & \\ & \underline{\underline{A}}_{II}^2 & & \\ & & \ddots & \\ & & & \underline{\underline{A}}_{II}^E \end{bmatrix} \quad (19.58)$$

donde el resto de la matriz fuera de la diagonal en bloques es cero.

De forma similar se define

$$\underline{\underline{A}}_{I\Gamma}^\delta \equiv [\langle w_I^i, w_\Gamma^\alpha \rangle], \quad \underline{\underline{A}}_{\Gamma I}^\delta \equiv [\langle w_\Gamma^\alpha, w_I^i \rangle] \quad (19.59)$$

y

$$\underline{\underline{A}}_{\Gamma\Gamma}^\delta \equiv [\langle w_\Gamma^\alpha, w_\Gamma^\alpha \rangle] \quad (19.60)$$

para toda $\delta = 1, \dots, E$, obsérvese que como $\bar{\mathcal{B}}_\Gamma = \mathcal{B}_{\Gamma J} \cup \mathcal{B}_{\Gamma M}$ entonces

$$\underline{\underline{A}}_{\Gamma\Gamma}^\delta = [\langle w_\Gamma^\alpha, w_\Gamma^\alpha \rangle] = [\langle w_{\Gamma J}^\alpha, w_{\Gamma J}^\alpha \rangle] + [\langle w_{\Gamma M}^\alpha, w_{\Gamma M}^\alpha \rangle] \quad (19.61)$$

también que $\underline{A}_{\Gamma I}^\delta = \left(\underline{A}_{\Gamma I}^\delta \right)^T$. Entonces las matrices virtuales $\underline{A}_{\Gamma I}$, $\underline{A}_{\Gamma \Gamma}$ y $\underline{A}_{\Gamma \Gamma}$ quedarán definidas como

$$\underline{A}_{\Gamma I} \equiv \begin{bmatrix} \underline{A}_{\Gamma I}^1 \\ \underline{A}_{\Gamma I}^2 \\ \vdots \\ \underline{A}_{\Gamma I}^E \end{bmatrix} \quad (19.62)$$

$$\underline{A}_{\Gamma I} \equiv [\underline{A}_{\Gamma I}^1 \quad \underline{A}_{\Gamma I}^2 \quad \cdots \quad \underline{A}_{\Gamma I}^E] \quad (19.63)$$

y

$$\underline{A}_{\Gamma \Gamma} \equiv \left[\sum_{i=1}^E \underline{A}_{\Gamma \Gamma}^i \right] \quad (19.64)$$

donde $\left[\sum_{i=1}^E \underline{A}_{\Gamma \Gamma}^i \right]$ es construida sumando las $\underline{A}_{\Gamma \Gamma}^i$ según el orden de los nodos globales versus los nodos locales.

También se considera al vector $\underline{u} \equiv (u_1, \dots, u_E)$ el cual se escribe como $\underline{u} = (\underline{u}_I, \underline{u}_\Gamma)$ donde $\underline{u}_I = (u_1, \dots, u_{N_I})$ y $\underline{u}_\Gamma = (u_1, \dots, u_{N_\Gamma})$.

Así, el sistema virtual

$$\begin{aligned} \underline{A}_{II} \underline{u}_I + \underline{A}_{I\Gamma} \underline{u}_\Gamma &= \underline{b}_I \\ \underline{A}_{\Gamma I} \underline{u}_I + \underline{A}_{\Gamma \Gamma} \underline{u}_\Gamma &= \underline{b}_\Gamma \end{aligned} \quad (19.65)$$

quedando expresado como

$$\begin{bmatrix} \underline{A}_{II}^1 & & \\ & \ddots & \\ & & \underline{A}_{II}^E \end{bmatrix} \begin{bmatrix} \underline{u}_{I1} \\ \vdots \\ \underline{u}_{IE} \end{bmatrix} + \begin{bmatrix} \underline{A}_{I\Gamma}^1 \\ \vdots \\ \underline{A}_{I\Gamma}^E \end{bmatrix} \begin{bmatrix} \underline{u}_{\Gamma 1} \\ \vdots \\ \underline{u}_{\Gamma E} \end{bmatrix} = \begin{bmatrix} \underline{b}_{I1} \\ \vdots \\ \underline{b}_{IE} \end{bmatrix}$$

$$\begin{bmatrix} \underline{A}_{\Gamma I}^1 & \cdots & \underline{A}_{\Gamma I}^E \end{bmatrix} \begin{bmatrix} \underline{u}_{I1} \\ \vdots \\ \underline{u}_{IE} \end{bmatrix} + \begin{bmatrix} \underline{A}_{\Gamma \Gamma} \end{bmatrix} \begin{bmatrix} \underline{u}_{\Gamma 1} \\ \vdots \\ \underline{u}_{\Gamma E} \end{bmatrix} = \begin{bmatrix} \underline{b}_{\Gamma 1} \\ \vdots \\ \underline{b}_{\Gamma E} \end{bmatrix}$$

o más compactamente como $\underline{A}\underline{u} = \underline{b}$, nótese que las matrices $\underline{A}_{\Gamma \Gamma}^i, \underline{A}_{\Gamma I}^i, \underline{A}_{II}^i$ y \underline{A}_{II}^i son matrices bandadas.

Si ahora despejo \underline{u}_I de la primera ecuación del sistema dado por la Ec.(19.65), se obtiene

$$\underline{u}_I = \left(\underline{A}_{II} \right)^{-1} \left(\underline{b}_I - \underline{A}_{I\Gamma} \underline{u}_\Gamma \right) \quad (19.66)$$

si sustituyo \underline{u}_I en la segunda ecuación del sistema dado por la Ec.(19.65) entonces se obtiene

$$\left(\underline{A}_{\Gamma\Gamma} - \underline{A}_{\Gamma I} \left(\underline{A}_{II} \right)^{-1} \underline{A}_{I\Gamma} \right) \underline{u}_\Gamma = \underline{b}_\Gamma - \underline{A}_{\Gamma I} \left(\underline{A}_{II} \right)^{-1} \underline{b}_I \quad (19.67)$$

en la cual los nodos interiores no figuran en la ecuación y todo queda en función de los nodos de la frontera interior \underline{u}_Γ .

A la matriz formada por $\underline{A}_{\Gamma\Gamma} - \underline{A}_{\Gamma I} \left(\underline{A}_{II} \right)^{-1} \underline{A}_{I\Gamma}$ se le conoce como el complemento de Schur global y se le denota como

$$\underline{S} = \underline{A}_{\Gamma\Gamma} - \underline{A}_{\Gamma I} \left(\underline{A}_{II} \right)^{-1} \underline{A}_{I\Gamma}. \quad (19.68)$$

En este caso, como estoy planteando todo en términos de subdominios Ω_i , con $i = 1, \dots, E$, entonces las matrices $\underline{A}_{\Gamma\Gamma}^i, \underline{A}_{\Gamma I}^i, \underline{A}_{I\Gamma}^i$ y \underline{A}_{II}^i quedan definidas de manera local, así que se procede a definir el complemento de Schur local como

$$\underline{S}_i = \underline{A}_{\Gamma\Gamma}^i - \underline{A}_{\Gamma I}^i \left(\underline{A}_{II}^i \right)^{-1} \underline{A}_{I\Gamma}^i \quad (19.69)$$

de forma adicional se define

$$\underline{b}_i = \underline{b}_{\Gamma_i} - \underline{A}_{\Gamma I}^i \left(\underline{A}_{II}^i \right)^{-1} \underline{b}_{I_i}. \quad (19.70)$$

El sistema dado por la Ec.(19.67) se escribe como

$$\underline{S} \underline{u}_\Gamma = \underline{b} \quad (19.71)$$

y queda definido de manera virtual a partir de

$$\left[\sum_{i=1}^E \underline{S}_i \right] \underline{u}_\Gamma = \left[\sum_{i=1}^E \underline{b}_i \right] \quad (19.72)$$

donde $\left[\sum_{i=1}^E \underline{S}_i \right]$ y $\left[\sum_{i=1}^E \underline{b}_i \right]$ podrían ser construida sumando las S_i y b_i respectivamente según el orden de los nodos globales versus los nodos locales.

El sistema lineal virtual obtenido de la Ec.(19.71) se resuelve -dependiendo de si es o no simétrico- eficientemente usando el método de Gradiente Conjugado o alguna variante de GMRES, para ello no es necesario construir la

matriz \underline{S} con las contribuciones de cada S_i correspondientes al subdominio i , lo que se hace es pasar a cada subdominio el vector \underline{u}_{Γ_i} correspondiente a la i -ésima iteración del método iterativo usado, para que en cada subdominio se evalúe $\tilde{\underline{u}}_{\Gamma}^i = \underline{S}_i \underline{u}_{\Gamma_i}$ localmente y con el resultado se forma el vector $\tilde{\underline{u}}_{\Gamma} = \sum_{i=1}^E \tilde{\underline{u}}_{\Gamma_i}$ y se continué con los demás pasos del método. Esto es ideal para una implementación en paralelo del método de Gradiente Conjugado o variante de GMRES.

Una vez resuelto el sistema de la Ec.(19.72) en el que se ha encontrado la solución para los nodos de la frontera interior \underline{u}_{Γ} , entonces debo resolver localmente los \underline{u}_{I_i} correspondientes a los nodos interiores para cada subespacio Ω_i , para esto empleo

$$\underline{u}_{I_i} = \left(\underline{A}_{II}^i \right)^{-1} \left(\underline{b}_{I_i} - \underline{A}_{I\Gamma}^i \underline{u}_{\Gamma_i} \right) \quad (19.73)$$

para cada $i = 1, 2, \dots, E$, quedando así resuelto el problema $\underline{A}u = \underline{b}$ tanto en los nodos interiores \underline{u}_{I_i} como en los de la frontera interior \underline{u}_{Γ_i} correspondientes a cada subespacio Ω_i .

Observación 9 *Nótese que normalmente las matrices locales \underline{S}_i y $\left(\underline{A}_{II}^i \right)^{-1}$ no se construyen, ya que estas serían matrices densas y su construcción es computacionalmente costosa, y como sólo interesa el producto $\underline{S} \underline{y}_{\Gamma}$, o más precisamente $\left[\sum_{i=1}^E \underline{S}_i \right] \underline{y}_{\Gamma}$, entonces si llamo \underline{y}_{Γ_i} al vector correspondiente al subdominio i , entonces se obtiene*

$$\tilde{\underline{u}}_{\Gamma}^i = \left(\underline{A}_{\Gamma\Gamma}^i - \underline{A}_{\Gamma I}^i \left(\underline{A}_{II}^i \right)^{-1} \underline{A}_{I\Gamma}^i \right) \underline{y}_{\Gamma_i}. \quad (19.74)$$

Para evaluar de forma eficiente esta expresión, se realizan las siguientes operaciones equivalentes

$$\begin{aligned} \underline{x1} &= \underline{A}_{\Gamma\Gamma}^i \underline{y}_{\Gamma_i} & (19.75) \\ \underline{x2} &= \left(\underline{A}_{\Gamma I}^i \left(\underline{A}_{II}^i \right)^{-1} \underline{A}_{I\Gamma}^i \right) \underline{y}_{\Gamma_i} \\ \tilde{\underline{u}}_{\Gamma}^i &= \underline{x1} - \underline{x2} \end{aligned}$$

la primera y tercera expresión no tienen ningún problema en su evaluación, para la segunda expresión se debe evaluar

$$\underline{x3} = \underline{A}_{I\Gamma}^i \underline{y}_{\Gamma_i} \quad (19.76)$$

con este resultado intermedio se debe calcular

$$\underline{x4} = \left(\underline{A}_{II}^i \right)^{-1} \underline{x3} \quad (19.77)$$

pero como no se cuenta con $\left(\underline{A}_{II}^i \right)^{-1}$, entonces multiplico la expresión por \underline{A}_{II}^i obteniendo

$$\underline{A}_{II}^i \underline{x4} = \underline{A}_{II}^i \left(\underline{A}_{II}^i \right)^{-1} \underline{x3} \quad (19.78)$$

al simplificar, se obtiene

$$\underline{A}_{II}^i \underline{x4} = \underline{x3}. \quad (19.79)$$

Esta última expresión puede ser resuelta usando Factorización LU, Gradiente Conjugado o alguna variante de GMRES -cada una de estas opciones tiene ventajas y desventajas computacionales que deben ser evaluadas al momento de implementar el código para un problema particular-. Una vez obtenido $\underline{x4}$, se puede calcular

$$\underline{x2} = \underline{A}_{\Gamma I}^i \underline{x4} \quad (19.80)$$

así

$$\underline{\tilde{u}}_{\Gamma}^i = \underline{x1} - \underline{x2} \quad (19.81)$$

completando la secuencia de operaciones necesarias para obtener $\underline{S}_i \underline{y}_{\Gamma_i}$.

Observación 10 En el caso del cálculo de

$$\underline{b}_i = \underline{b}_{\Gamma_i} - \underline{A}_{\Gamma I}^i \left(\underline{A}_{II}^i \right)^{-1} \underline{b}_{L_i} \quad (19.82)$$

algo análogo al comentario anterior deberá de hacerse, ya que de nuevo está involucrado $\left(\underline{A}_{II}^i \right)^{-1}$, por ello se debe usar el siguiente procedimiento para evaluar de forma eficiente esta expresión, realizando las operaciones equivalentes

$$\underline{y1} = \left(\underline{A}_{II}^i \right)^{-1} \underline{b}_{L_i} \quad (19.83)$$

multiplicando por \underline{A}_{II}^i a la última expresión, se obtiene

$$\underline{A}_{II}^i \underline{y1} = \underline{A}_{II}^i \left(\underline{A}_{II}^i \right)^{-1} \underline{b}_{L_i} \quad (19.84)$$

simplificando, se obtiene

$$\left(\underline{\underline{A}}_{II}^i\right) \underline{y1} = \underline{b}_{I_i} \quad (19.85)$$

donde esta última expresión puede ser resuelta usando Factorización LU, Gradiente Conjugado o alguna variante de GMRES, luego se hace

$$\underline{y2} = \underline{\underline{A}}_{\Gamma I}^i \underline{y1} \quad (19.86)$$

y para finalizar el cálculo, se obtiene

$$\underline{b}_i = \underline{b}_{\Gamma_i} - \underline{y2}. \quad (19.87)$$

Observación 11 *En la evaluación de*

$$\underline{u}_{I_i} = \left(\underline{\underline{A}}_{II}^i\right)^{-1} \left(\underline{b}_{I_i} - \underline{\underline{A}}_{\Gamma I}^i \underline{u}_{\Gamma_i}\right) \quad (19.88)$$

esta de nuevo involucrado $\left(\underline{\underline{A}}_{II}^i\right)^{-1}$, por ello debo de usar el siguiente procedimiento para evaluar de forma eficiente esta expresión, realizando las operaciones equivalentes

$$\begin{aligned} \underline{x4} &= \underline{b}_{I_i} - \underline{\underline{A}}_{\Gamma I}^i \underline{u}_{\Gamma_i} \\ \underline{u}_{I_i} &= \left(\underline{\underline{A}}_{II}^i\right)^{-1} \underline{x4} \end{aligned} \quad (19.89)$$

multiplicando por $\underline{\underline{A}}_{II}^i$ a la última expresión, se obtiene

$$\underline{\underline{A}}_{II}^i \underline{u}_{I_i} = \underline{\underline{A}}_{II}^i \left(\underline{\underline{A}}_{II}^i\right)^{-1} \underline{x4} \quad (19.90)$$

simplificando, se obtiene

$$\underline{\underline{A}}_{II}^i \underline{u}_{I_i} = \underline{x4} \quad (19.91)$$

esta última expresión puede ser resuelta usando Factorización LU, Cholesky, Gradiente Conjugado o alguna variante de GMRES.

Como se indico en las últimas observaciones, para resolver el sistema $\underline{\underline{A}}_{II}^i \underline{x} = \underline{b}$ se puede usar Factorización LU, Factorización Cholesky, Gradiente Conjugado, alguna variante de GMRES o cualquier otro método para resolver sistemas lineales, pero deberá de usarse aquel que proporcione la mayor velocidad en el cálculo o que consuma la menor cantidad de memoria

-ambas condicionantes son mutuamente excluyentes-, por ello la decisión de que método usar deberá de tomarse al momento de tener que resolver un problema particular en un equipo dado y básicamente el condicionante es el tamaño de la matriz $\underline{\underline{A}}_{II}^i$.

Para usar el método de Factorización LU, se deberá primeramente de factorizar la matriz bandada $\underline{\underline{A}}_{II}^i$ en una matriz $\underline{\underline{LU}}$, la cual es bandada pero incrementa el tamaño de la banda a más del doble, pero esta operación sólo se deberá de realizar una vez en cada subdominio, y para solucionar los diversos sistemas lineales $\underline{\underline{A}}_{II}^i x = \underline{b}$ sólo será necesario evaluar los sistemas

$$\begin{aligned}\underline{L}y &= \underline{b} \\ \underline{U}x &= \underline{y}\end{aligned}\tag{19.92}$$

en donde \underline{y} es un vector auxiliar. Esto proporciona una manera eficiente de evaluar el sistema lineal pero el consumo en memoria para un problema particular puede ser excesivo.

Por ello, si el problema involucra una gran cantidad de nodos interiores y el equipo en el que se implantará la ejecución del programa tiene una cantidad de memoria limitada, es recomendable usar el método de Gradiente Conjugado o alguna variante de GMRES, este consume una cantidad de memoria adicional pequeña, pero puede consumir muchas iteraciones con el consecuente aumento de tiempo de cómputo para alcanzar la precisión de la Factorización LU.

De esta forma, es posible adaptar el código para tomar en cuenta desde la implementación al equipo de cómputo al que se tenga acceso y poder sacar el máximo provecho al método de Subestructuración en la resolución de problemas elípticos de gran envergadura.

El número de condicionamiento del complemento de Schur sin preconditionamiento puede ser estimado, para ello:

Definición 120 *Introduzco una norma- L^2 equivalente sobre Γ mediante*

$$\|\underline{u}_\Gamma\|_\Gamma^2 = \sum_{i=1}^E \|\underline{u}_\Gamma\|_{L^2(\partial\Omega_i)}^2.\tag{19.93}$$

Teorema 121 *Sea \underline{u}_Γ la traza de funciones de elemento finito en V^h sobre Γ , asumo que los coeficientes de la ecuación diferencial parcial $\rho_i = 1$, $i = 1, 2, \dots, E$, y que la malla fina \mathcal{T}_h y la malla gruesa \mathcal{T}_H sea cuasi-uniforme.*

Entonces existen dos constantes positivas c y C , independientes de h y H , tal que

$$cH \| \underline{u}_\Gamma \|_\Gamma^2 \leq s(\underline{u}_\Gamma, \underline{u}_\Gamma) \leq Ch^{-1} \| \underline{u}_\Gamma \|_\Gamma^2 \quad (19.94)$$

de este modo

$$\kappa = \text{cond}(\underline{\underline{S}}) \leq \frac{C}{Hh}. \quad (19.95)$$

Por analogía al método de subestructuración desarrollado anteriormente, dado un sistema lineal $\underline{\underline{M}}x = \underline{\underline{f}}$ que proviene de la discretización de algún método tipo Diferencias Finitas, Elemento Finito o Volumen Finito, siempre es posible reacomodarlo como (véase [7])

$$\begin{pmatrix} \underline{\underline{A}} & \underline{\underline{B}} \\ \underline{\underline{C}} & \underline{\underline{D}} \end{pmatrix} \begin{pmatrix} \underline{u} \\ \underline{v} \end{pmatrix} = \begin{pmatrix} \underline{a} \\ \underline{b} \end{pmatrix} \quad (19.96)$$

con $\underline{\underline{M}} = \begin{pmatrix} \underline{\underline{A}} & \underline{\underline{B}} \\ \underline{\underline{C}} & \underline{\underline{D}} \end{pmatrix}$, $\underline{x} = \begin{pmatrix} \underline{u} \\ \underline{v} \end{pmatrix}$ y $\underline{\underline{f}} = \begin{pmatrix} \underline{a} \\ \underline{b} \end{pmatrix}$, en la cual la matriz $\underline{\underline{A}}$ sea invertible, entonces

$$(\underline{\underline{D}} - \underline{\underline{C}}\underline{\underline{A}}^{-1}\underline{\underline{B}}) \underline{v} = \underline{b} - \underline{\underline{C}}\underline{\underline{A}}^{-1}\underline{a} \quad (19.97)$$

y donde

$$\underline{u} = \underline{\underline{A}}^{-1} (\underline{a} - \underline{\underline{B}}\underline{v}). \quad (19.98)$$

Así, he transformado el sistema $\underline{\underline{M}}x = \underline{\underline{f}}$, en otro equivalente

$$\underline{\underline{N}}\underline{v} = \underline{g} \quad (19.99)$$

pero con un menor número de grados de libertad, donde

$$\underline{\underline{N}} = (\underline{\underline{D}} - \underline{\underline{C}}\underline{\underline{A}}^{-1}\underline{\underline{B}}), \quad \underline{g} = \underline{b} - \underline{\underline{C}}\underline{\underline{A}}^{-1}\underline{a} \quad (19.100)$$

a esta descomposición matricial se le conoce como la descomposición de Schur.

20 Apéndice B: Estructura Óptima de las Matrices en su Implementación Computacional

Una parte fundamental de la implementación computacional de los métodos numéricos de resolución de sistemas algebraicos, es utilizar una forma óptima de almacenar, recuperar y operar las matrices, tal que, facilite los cálculos que involucra la resolución de grandes sistemas de ecuaciones lineales cuya implementación puede ser secuencial o paralela (véase [13]).

El sistema lineal puede ser expresado en la forma matricial $\underline{A}u = f$, donde la matriz \underline{A} -que puede ser real o virtual- es de tamaño $n \times n$ con banda b , pero el número total de datos almacenados en ella es a los más $n * b$ números de doble precisión, en el caso de ser simétrica la matriz, el número de datos almacenados es menor a $(n * b)/2$. Además si el problema que la originó es de coeficientes constantes el número de valores almacenados se reduce drásticamente a sólo el tamaño de la banda b .

En el caso de que el método para la resolución del sistema lineal a usar sea del tipo Factorización LU o Cholesky, la estructura de la matriz cambia, ampliándose el tamaño de la banda de b a $2 * b + 1$ en la factorización, en el caso de usar métodos iterativos tipo CGM o GMRES la matriz se mantiene intacta con una banda b .

Para la resolución del sistema lineal virtual asociada a los métodos de descomposición de dominio, la operación básica que se realiza de manera reiterada, es la multiplicación de una matriz por un vector $\underline{v} = \underline{C}u$, la cual es necesario realizar de la forma más eficiente posible.

Un factor determinante en la implementación computacional, para que esta resulte eficiente, es la forma de almacenar, recuperar y realizar las operaciones que involucren matrices y vectores, de tal forma que la multiplicación se realice en la menor cantidad de operaciones y que los valores necesarios para realizar dichas operaciones queden en la medida de lo posible contiguos para ser almacenados en el Cache³⁴ del procesador.

³⁴Nótese que la velocidad de acceso a la memoria principal (RAM) es relativamente lenta con respecto al Cache, este generalmente está dividido en sub-Caches L1 -de menor tamaño y el más rápido-, L2 y hasta L3 -el más lento y de mayor tamaño- los cuales son de tamaño muy reducido con respecto a la RAM.

Por ello, cada vez que las unidades funcionales de la Unidad de Aritmética y Lógica requieren un conjunto de datos para implementar una determinada operación en los registros, solicitan los datos primeramente a los Caches, estos consumen diversa cantidad de ciclos de reloj para entregar el dato si lo tienen -pero siempre el tiempo es menor que

20.1 Almacenamiento en la Memoria RAM

La memoria RAM (Random Access Memory) o memoria de acceso aleatorio es un componente físico de nuestro ordenador, generalmente instalado sobre la misma placa base. La memoria RAM es extraíble y se puede ampliar mediante módulos de distintas capacidades.

La función de la memoria RAM es la de cargar los datos e instrucciones que se ejecutan en el procesador. Estas instrucciones y datos provienen del sistema operativo, dispositivos de entrada y salida, de discos duros y todo lo que está instalado en el equipo.

En la memoria RAM se almacenan todos los datos e instrucciones de los programas que se están ejecutando, estas son enviadas desde las unidades de almacenamiento antes de su ejecución. De esta forma podremos tener disponibles todos los programas que ejecutamos. Se llama memoria de acceso aleatorio porque se puede leer y escribir en cualquiera de sus posiciones de memoria sin necesidad de respetar un orden secuencial para su acceso.

La RAM dinámica cuenta con un reloj interno capaz de sincronizar esta con el procesador. De esta forma se mejoran notablemente los tiempos de acceso y la eficiencia de comunicación entre ambos elementos. Actualmente todas nuestras computadoras cuentan con este tipo de memorias operando en ellos. Las principales tipos de memoria son: DDR, DDR2, DDR3, DDR4 y la nueva DDR5. Donde las tasas de transferencia (GB/s) son: DDR (2.1 - 3.2), DDR2 (4.2 - 6.4), DDR3 (8.5 - 14.9), DDR4 (17 - 25.6) y DDR5 (38.4 - 51.2). La característica más importante es que, por ejemplo, en la memoria DDR4 cuatro cores pueden acceder simultáneamente a ella y en la DDR5 serán cinco cores.

Caché L1, L2 y L3 La memoria Caché es otra de las especificaciones importantes de los procesadores, y sirve de manera esencial de la misma manera que la memoria RAM: como almacenamiento temporal de datos. No obstante, dado que la memoria Caché está en el procesador en sí, es mucho más rápida y el procesador puede acceder a ella de manera más eficiente, así que el tamaño de esta memoria puede tener un impacto bastante notable en el rendimiento, especialmente cuando se realizan tareas que demandan un uso intensivo del CPU como en el cómputo de alto desempeño o cómputo científico.

solicitarle el dato a la memoria principal-; en caso de no tenerlo, se solicitan a la RAM para ser cargados a los caches y poder implementar la operación solicitada.

La Caché se divide en diferentes jerarquías de acceso:

- La Caché L1 es el primer sitio donde la CPU buscará información, pero también es la más pequeña y la más rápida, a veces para mayor eficiencia, la Caché L1 se subdivide en L1d (datos) y L1i (instrucciones), actualmente los procesadores modernos en cada core tiene su propio cache de datos e instrucciones.
- La Caché L2 suele ser más grande que la L1 pero es algo más lenta. Sin embargo, por norma general es la que mayor impacto tiene en el rendimiento, este también está incluido en cada core.
- La Caché L3 es mucho más grande que las anteriores, y generalmente se comparte entre todos los núcleos del procesador (a diferencia de las anteriores, que normalmente van ligadas a cada core). Este tercer nivel es en el que buscará el procesador la información tras no encontrarla en la L1 y L2, por lo que su tiempo de acceso es todavía mayor.

Para poner en contexto la relevancia de la memoria Caché, supongamos que el acceso a los datos de la memoria Caché L1 por el procesador es de dos ciclos de reloj, el acceso a la memoria Caché L2 es de 6 ciclos de reloj, el acceso a la memoria Caché L3 es de 12 ciclos y el acceso a la RAM es de 32 ciclos de reloj.

Además supongamos que la operación suma y resta necesitan de 2 ciclos de reloj para completar la operación una vez que cuente con los datos involucrados en dicha operación, que la multiplicación requiere 4 ciclos de reloj para completar la operación, la división necesita 6 ciclos de reloj para completar la operación y estamos despreciando el tiempo necesario para poner los datos del Caché L1 a los registros del procesador para poder iniciar el cálculo, así también despreciamos el tiempo requerido para sacar el resultado de los registros del procesador al Caché L1.

Esto nos da una idea del número máximo teórico de operaciones básicas que un procesador puede realizar por segundo dependiendo de la velocidad de reloj de la CPU³⁵.

³⁵Por ejemplo en un procesador AMD Ryzen 9 3900X con 12 Cores (2 Threads por Core) por procesador emulando un total de 24 Cores, corre a una frecuencia base de 3,340 MHz, con una frecuencia mínima de 2,200 MHz y máxima de 4,917 Mhz, con Caché L1d de 384 KiB, L1i de 384 KiB, Caché L2 de 6 MiB y Caché L3 de 64 MiB.

Si nosotros necesitamos hacer la multiplicación de una matriz \underline{A} es de tamaño $n \times n$ por un vector \underline{u} de tamaño n y guardar el resultado en el vector \underline{f} de tamaño n . Entonces algunos escenarios son posibles:

1. Si el código del programa cabe en el Caché L1 de instrucciones y la matriz \underline{A} , los vectores \underline{u} y \underline{f} caben íntegramente en el Caché L1 de datos, entonces el procesador estará siendo utilizado de forma óptima al hacer los cálculos pues no tendrá tiempos muertos por espera de datos.
2. Si el código del programa cabe en el Caché L1 de instrucciones y los vectores \underline{u} y \underline{f} caben íntegramente en el Caché L1 de datos pero la matriz \underline{A} está dispersa entre los Cachés L1 y L2, entonces el procesador estará teniendo algunos tiempos muertos mientras carga la parte que necesita de la matriz del Caché L2 a L1 para hacer los cálculos y utilizado de forma óptima el procesador mientras no salga del Caché L1.
3. Si el código del programa cabe en el Caché L1 de instrucciones y los vectores \underline{u} y \underline{f} caben íntegramente en el Caché L1 de datos pero la matriz \underline{A} está dispersa entre los Cachés L1, L2 y L3, entonces el procesador estará teniendo muchos tiempos muertos mientras carga la parte que necesita de la matriz del Caché L3 y L2 a L1 para hacer los cálculos resultando en mediana eficiencia en el uso del procesador.
4. Si el código del programa cabe en el Caché L1 de instrucciones y los vectores \underline{u} y \underline{f} caben íntegramente en los Cachés L3, L2 y L1 pero los datos de la matriz \underline{A} está dispersa entre la RAM y los Cachés L3, L2 y L1, entonces el procesador estará teniendo un exceso de tiempos muertos mientras carga la parte que necesita de la matriz de la RAM a los Cachés L3, L2 y L1 para hacer los cálculos resultando en una gran pérdida de eficiencia en el uso del procesador.

Además, debemos recordar que la computadora moderna nunca dedica el cien por ciento del CPU a un solo programa, ya que los equipos son mul-

titarea³⁶ y multiusuario³⁷ por lo que la conmutación de procesos (que se realiza cada cierta cantidad de milisegundos) degrada aún más la eficiencia computacional de los procesos que demandan un uso intensivo de CPU³⁸.

Dado que la multiplicación de una matriz \underline{C} por un vector \underline{u} , dejando el resultado en \underline{v} se realiza mediante el algoritmo

```
for (i=0; i<ren; i++)
{
    s = 0.0;
    for (j=0; j < col; j++)
    {
        s += C[i][j]*u[j];
    }
    v[i] = s;
}
```

³⁶Cuentan con la capacidad para ejecutar varios procesos simultáneamente en uno o más procesadores, para ello necesitan hacer uso de la conmutación de tareas, es decir, cada cierto tiempo detiene el programa que está corriendo y guardan sus datos, para poder cargar en memoria otro programa y sus respectivos datos y así reiniciar su ejecución por un período determinado de tiempo, una vez concluido su tiempo de ejecución se reinicia la conmutación de tareas con otro proceso.

³⁷Se refiere a todos aquellos sistemas operativos que permiten el empleo de sus procesamientos y servicios al mismo tiempo. Así, el sistema operativo cuenta con la capacidad de satisfacer las necesidades de varios usuarios al mismo tiempo, siendo capaz de gestionar y compartir sus recursos en función del número de usuarios que estén conectados a la vez.

³⁸Actualmente existen una gran cantidad de distribuciones de GNU/Linux que vienen muy optimizadas intentando conseguir la mejor desenvolvura de su arquitectura y configuraciones de serie. En el caso de la configuración por omisión de Debian GNU/Linux y Ubuntu, están pensadas para que sean lo más robusta posible y que se use en todas las circunstancias imaginables, por ello están optimizadas de forma muy conservadora para tener un equilibrio entre eficiencia y consumo de energía. Pero es posible agregar uno o más Kernels GNU/Linux generados por terceros que contenga las optimizaciones necesarias para hacer más eficiente y competitivo en cuestiones de gestión y ahorro de recursos del sistema.

Hay varias opciones del Kernel GNU/Linux optimizado ([Liquorix](#) viene optimizado para multimedia y Juegos, por otro lado [XanMod](#) tiene uno para propósito general, otro aplicaciones críticas en tiempo real y otro más para cálculos intensivos) de las últimas versiones estable del Kernel.

Para lograr una eficiente implementación del algoritmo anterior, es necesario que el gran volumen de datos desplazados de la memoria al Cache y viceversa sea mínimo. Por ello, los datos se deben agrupar para que la operación más usada -en este caso multiplicación matriz por vector- se realice con la menor solicitud de datos a la memoria principal, si los datos usados -renglón de la matriz- se ponen contiguos minimizará los accesos a la memoria principal, pues es más probable que estos estarán contiguos en el Cache al momento de realizar la multiplicación.

Por ejemplo, en el caso de matrices bandadas de tamaño de banda b , el algoritmo anterior se simplifica a

```

for (i=0; i<ren; i++)
{
    s= 0.0;
    for (k=0; k < ban; k++)
    {
        if ((Ind[k] + i) >= 0 && (Ind[k]+i) < ren)
            s += Dat[i][k]*u[Ind[k]+i];
    }
    v[i]=s;
}

```

Si, la solicitud de memoria para $\text{Dat}[i]$ se hace de tal forma que los datos del renglón estén continuos -son b números de punto flotante-, esto minimizará los accesos a la memoria principal en cada una de las operaciones involucradas en el producto, como se explica en las siguientes secciones.

20.2 Matrices Bandadas

En el caso de las matrices bandadas de banda b -sin pérdida de generalidad y para propósitos de ejemplificación se supone pentadiagonal- típicamente

Para el primer caso, al ser la matriz simétrica, sólo es necesario almacenar la parte con índices mayores o iguales a cero, de tal forma que se buscará el índice que satisfaga $ind = |j - i|$, reduciendo el tamaño de la banda a $b/2$ en la matriz \underline{A} .

Para el segundo caso, al tener coeficientes constantes el operador diferencial, los valores de los renglones dentro de cada columna de la matriz son iguales, y sólo es necesario almacenarlos una sola vez, reduciendo drásticamente el tamaño de la matriz de datos.

Implementación de Matrices Bandadas Orientada a Objetos en C++ Una forma de implementar la matriz bandada \underline{A} de banda b en C++ es mediante:

```
// Creacion
double **Dat;
Dat = new double*[ren];
for (i = 0; i < ren; i++) Dat[i] = new double[b];
int *Ind;
Ind = new int[b];
// Inicializacion
for (i = 0; i < ren; i++)
    for (j = 0; j < b; j++) Dat[i][j] = 0.0;
for (i = 0; i < b; i++) Ind[i] = 0;
```

20.3 Matrices Dispersas

Las matrices dispersas de a lo más b valores distintos por renglón -sin pérdida de generalidad y para propósitos de ejemplificación se supone $b = 3$ - que surgen en métodos de descomposición de dominio para almacenar algunas

matrices, típicamente tienen la siguiente forma

$$\underline{\underline{A}} = \begin{bmatrix} a_1 & & & b_1 & & c_1 \\ & a_2 & & b_2 & & c_2 \\ & & & a_3 & & b_3 & c_3 \\ a_4 & & & b_4 & & & \\ & & a_5 & & b_5 & & c_5 \\ a_6 & b_6 & c_6 & & & & \\ & & & & & a_7 & b_7 & c_7 \\ & & & a_8 & & b_8 & c_8 \\ & & & & & a_9 & b_9 & \end{bmatrix} \quad (20.4)$$

la cual puede ser almacenada usando el algoritmo (véase [13]) Jagged Diagonal Storage (JDC), optimizado para ser usado en C++. Para este ejemplo en particular, se hará uso de una matriz de índices

$$\underline{\underline{Ind}} = \begin{bmatrix} 1 & 6 & 9 \\ 2 & 5 & 8 \\ 5 & 8 & 9 \\ 1 & 4 & 0 \\ 3 & 6 & 9 \\ 1 & 2 & 3 \\ 7 & 8 & 9 \\ 4 & 7 & 8 \\ 7 & 8 & 0 \end{bmatrix} \quad (20.5)$$

y los datos serán almacenados usando la estructura

$$\underline{\underline{Dat}} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \\ a_4 & b_4 & 0 \\ a_5 & b_5 & c_5 \\ a_6 & b_6 & c_6 \\ a_7 & b_7 & c_7 \\ a_8 & b_8 & c_8 \\ a_9 & b_9 & 0 \end{bmatrix} \quad (20.6)$$

de tal forma que la matriz $\underline{\underline{A}}$ puede ser reconstruida de forma eficiente. Para obtener el valor $A_{i,j}$, busco el valor j en la lista de índices $\underline{\underline{Ind}}$ dentro del renglón i , si lo encuentro en la posición k , entonces $A_{i,j} = \underline{\underline{Dat}}_{ik}$, en otro caso $A_{i,j} = 0$.

Casos Particulares de la Matriz Dispersa $\underline{\underline{A}}$ Si la matriz $\underline{\underline{A}}$, que al ser almacenada, se observa que existen a lo más r diferentes renglones con valores distintos de los n con que cuenta la matriz y si $r \ll n$, entonces es posible sólo guardar los r renglones distintos y llevar un arreglo que contenga la referencia al renglón almacenado.

Implementación de Matrices Dispersas Orientada a Objetos en C++ Una forma de implementar la matriz bandada $\underline{\underline{A}}$ de banda b en C++ es mediante:

```
// Creación
double **Dat;
Dat = new double*[ren];
for (i = 0; i < ren; i++) Dat[i] = new double[b];
int **Ind;
Ind = new int*[ren];
for (i = 0; i < ren; i++) Ind[i] = new int[b];
// Inicialización
for (i = 0; i < ren; i++)
    for (j = 0; j < b; j++) Dat[i][j] = 0.0, Ind[i][j] = -1;
```

20.4 Multiplicación Matriz-Vector

Los métodos de descomposición de dominio requieren por un lado la resolución de al menos un sistema lineal y por el otro lado requieren realizar la operación de multiplicación de matriz por vector, i.e. $\underline{\underline{C}}\underline{u}$ de la forma más eficiente posible, por ello los datos se almacenan de tal forma que la multiplicación se realice en la menor cantidad de operaciones.

Dado que la multiplicación de una matriz $\underline{\underline{C}}$ por un vector \underline{u} , dejando el resultado en \underline{v} se realiza mediante el algoritmo:

```
for (i=0; i<ren; i++)
{
    s = 0.0;
    for (j=0; j < col; j++)
    {
        s += C[i][j]*u[j];
    }
}
```

```

        v[i] = s;
    }

```

En el caso de matrices bandadas, se simplifica a:

```

for (i=0; i<ren; i++)
{
    s= 0.0;
    for (k=0; k < ban; k++)
    {
        if ((Ind[k] + i) >= 0 && (Ind[k]+i) < ren)
            s += Dat[i][k]*u[Ind[k]+i];
    }
    v[i]=s;
}

```

De forma similar, en el caso de matrices dispersas, se simplifica a:

```

for (i=0; i<ren; i++)
{
    s = 0.0, k = 0
    while (Ind[i][k] != -1)
    {
        s += Dat[i][k]*u[Ind[i][k]];
        k++;
        if (k >= b) break;
    }
    v[i] = s;
}

```

De esta forma, al tomar en cuenta la operación de multiplicación de una matriz por un vector, donde el renglón de la matriz involucrado en la multiplicación queda generalmente en una región contigua del Cache, se hace óptima la operación de multiplicación de matriz por vector.

21 Apéndice C: Solución de Grandes Sistemas de Ecuaciones Lineales

Los modelos matemáticos de muchos sistemas en Ciencia e Ingeniería requieren el procesamiento de sistemas algebraicos de gran escala. La solución de este sistema lineal puede ser expresado en la forma matricial siguiente

$$\underline{A}u = \underline{f} \quad (21.1)$$

donde la matriz \underline{A} es de tamaño $n \times n$, está puede ser densa, bandada (de banda es b), dispersa (de máximo número de columnas ocupadas es b) o rara.

Los métodos de resolución del sistema algebraico de ecuaciones $\underline{A}u = \underline{f}$ se clasifican en dos grandes grupos (véase [10], [11], [12] y [14]):

- En los métodos directos la solución u se obtiene en un número fijo de pasos y sólo están sujetos a los errores de redondeo.
- En los métodos iterativos, se realizan iteraciones para aproximarse a la solución u aprovechando las características propias de la matriz \underline{A} , tratando de usar un menor número de pasos que en un método directo.

Por lo general, es conveniente usar bibliotecas³⁹ para implementar de forma eficiente a los vectores, matrices -bandadas, dispersas o raras- y resolver el sistemas lineal.

Los métodos iterativos rara vez se usan para resolver sistemas lineales de dimensión pequeña (el concepto de dimensión pequeña es muy relativo), ya que el tiempo necesario para conseguir una exactitud satisfactoria rebasa el que requieren los métodos directos. Sin embargo, en el caso de sistemas grandes con un alto porcentaje de elementos cero, son eficientes tanto en el almacenamiento en la computadora como en el tiempo que se invierte en su solución. Por ésta razón al resolver éstos sistemas algebraicos de ecuaciones es preferible aplicar métodos iterativos tal como Gradiente Conjugado -Conjugate Gradient Method (CGM)- o Residual Mínimo Generalizado -Generalized Minimum Residual Method (GMRES).

³⁹Algunas de las bibliotecas más usadas para resolver sistemas lineales usando matrices bandadas y dispersas son PETCS, HYPRE, ATLAS, LAPACK++, LAPACK, EISPACK, LINPACK, BLAS, entre muchas otras alternativas, tanto para implementaciones secuenciales como paralelas y más recientemente para hacer uso de los procesadores CUDA en las GPU de nVidia.

21.1 Métodos Directos

En los métodos directos (véase [10] y [13]), la solución \underline{u} se obtiene en un número fijo de pasos y sólo están sujetos a errores de redondeo. Entre los métodos más importantes se puede considerar: Factorización LU -para matrices simétricas y no simétricas- y Factorización Cholesky -para matrices simétricas-. En todos los casos la matriz original \underline{A} es modificada y en caso de usar la Factorización LU el tamaño de la banda b crece a $2b + 1$ si la factorización se realiza en la misma matriz. Los métodos aquí mencionados, se colocaron en orden descendente en cuanto al consumo de recursos computacionales y ascendente en cuanto al aumento en su eficiencia.

21.1.1 Eliminación Gausiana

Tal vez es el método más utilizado para encontrar la solución usando métodos directos. Este algoritmo sin embargo no es eficiente, ya que en general, un sistema de N ecuaciones requiere para su almacenaje en memoria de N^2 entradas para la matriz \underline{A} , pero cerca de $N^3/3 + O(N^2)$ multiplicaciones y $N^3/3 + O(N^2)$ adiciones para encontrar la solución siendo muy costoso computacionalmente.

La eliminación Gausiana se basa en la aplicación de operaciones elementales a renglones o columnas de tal forma que es posible obtener matrices equivalentes.

Escribiendo el sistema de N ecuaciones lineales con N incógnitas como

$$\sum_{j=1}^N a_{ij}^{(0)} x_j = a_{i,n+1}^{(0)}, \quad i = 1, 2, \dots, N \quad (21.2)$$

y si $a_{11}^{(0)} \neq 0$ y los pivotes $a_{ii}^{(i-1)}, i = 2, 3, \dots, N$ de las demás filas, que se obtienen en el curso de los cálculos, son distintos de cero, entonces, el sistema lineal anterior se reduce a la forma triangular superior (eliminación hacia adelante)

$$x_i + \sum_{j=i+1}^N a_{ij}^{(i)} x_j = a_{i,n+1}^{(i)}, \quad i = 1, 2, \dots, N \quad (21.3)$$

donde

$$\begin{aligned}
 k &= 1, 2, \dots, N; \{j = k + 1, \dots, N\} \\
 a_{kj}^{(k)} &= \frac{a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}; \\
 i &= k + 1, \dots, N + 1\{ \\
 a_{ij}^{(k)} &= a_{ij}^{(k-1)} - a_{kj}^{(k)} a_{ik}^{(k-1)} \} \} \}
 \end{aligned}$$

y las incógnitas se calculan por sustitución hacia atrás, usando las fórmulas

$$\begin{aligned}
 x_N &= a_{N,N+1}^{(N)}; & (21.4) \\
 i &= N - 1, N - 2, \dots, 1 \\
 x_i &= a_{i,N+1}^{(i)} - \sum_{j=i+1}^N a_{ij}^{(i)} x_j.
 \end{aligned}$$

En algunos casos nos interesa conocer $\underline{\underline{A}}^{-1}$, por ello si la eliminación se aplica a la matriz aumentada $\underline{\underline{A}} \mid \underline{\underline{I}}$ entonces la matriz $\underline{\underline{A}}$ de la matriz aumentada se convertirá en la matriz $\underline{\underline{I}}$ y la matriz $\underline{\underline{I}}$ de la matriz aumentada será $\underline{\underline{A}}^{-1}$. Así, el sistema $\underline{\underline{A}}\underline{\underline{u}} = \underline{\underline{b}}$ se transformará en $\underline{\underline{u}} = \underline{\underline{A}}^{-1}\underline{\underline{b}}$ obteniendo la solución de $\underline{\underline{u}}$.

21.1.2 Factorización LU

Sea $\underline{\underline{U}}$ una matriz triangular superior obtenida de $\underline{\underline{A}}$ por eliminación bandada. Entonces $\underline{\underline{U}} = \underline{\underline{L}}^{-1}\underline{\underline{A}}$, donde $\underline{\underline{L}}$ es una matriz triangular inferior con unos en la diagonal. Las entradas de $\underline{\underline{L}}^{-1}$ pueden obtenerse de los coeficientes $\underline{\underline{L}}_{ij}$ y pueden ser almacenados estrictamente en las entradas de la diagonal inferior de $\underline{\underline{A}}$ ya que estas ya fueron eliminadas. Esto proporciona una Factorización $\underline{\underline{LU}}$ de $\underline{\underline{A}}$ en la misma matriz $\underline{\underline{A}}$ ahorrando espacio de memoria, donde el ancho de banda cambia de b a $2b + 1$.

En el algoritmo de Factorización LU, se toma como datos de entrada del sistema $\underline{\underline{A}}\underline{\underline{u}} = \underline{\underline{f}}$, a la matriz $\underline{\underline{A}}$, la cual será factorizada en la misma matriz, está contendrá a las matrices $\underline{\underline{L}}$ y $\underline{\underline{U}}$ producto de la factorización, quedando

En el algoritmo de Factorización Cholesky, se toma como datos de entrada del sistema $\underline{A}u = \underline{f}$, a la matriz \underline{A} , la cual será factorizada en la misma matriz y contendrá a las matrices \underline{L} y \underline{L}^T producto de la factorización, quedando el método numérico esquemáticamente como:

$$\begin{aligned} &\text{para } i = 1, 2, \dots, n \text{ y } j = i + 1, \dots, n \\ &A_{ii} = \sqrt{\left(A_{ii} - \sum_{k=1}^{i-1} A_{ik}^2 \right)} \\ &A_{ji} = \left(A_{ji} - \sum_{k=1}^{i-1} A_{jk}A_{ik} \right) / A_{ii} \end{aligned} \quad (21.9)$$

El problema original $\underline{A}u = \underline{f}$ se escribe como $\underline{L}\underline{L}^T u = \underline{b}$, donde la búsqueda de la solución \underline{u} se reduce a la solución sucesiva de los sistemas lineales triangulares

$$\underline{L}y = \underline{f} \quad \text{y} \quad \underline{L}^T u = y \quad (21.10)$$

usando la formulación equivalente dada por las Ec.(21.7) y (21.8) para la des-composición LU.

La mayor ventaja de esta descomposición es que, en el caso en que es aplicable, el costo de cómputo es sustancialmente reducido, ya que requiere de $N^3/6$ multiplicaciones y $N^3/6$ sumas.

21.2 Métodos Iterativos

En los métodos iterativos, se realizan iteraciones para aproximarse a la solución \underline{u} aprovechando las características propias de la matriz \underline{A} , tratando de usar un menor número de pasos que en un método directo (véase [10] y [13]).

En los métodos iterativos tales como Jacobi, Gauss-Seidel y de Relajación Sucesiva (SOR) en el cual se resuelve el sistema lineal

$$\underline{A}u = \underline{f} \quad (21.11)$$

comienza con una aproximación inicial \underline{u}^0 a la solución \underline{u} y genera una sucesión de vectores $\{u^k\}_{k=1}^{\infty}$ que converge a \underline{u} . Los métodos iterativos traen consigo un proceso que convierte el sistema $\underline{A}u = \underline{f}$ en otro equivalente mediante la iteración de punto fijo de la forma $\underline{u} = \underline{T}u + \underline{c}$ para alguna matriz

fija \underline{T} y un vector \underline{c} . Luego de seleccionar el vector inicial \underline{u}^0 la sucesión de los vectores de la solución aproximada se genera calculando

$$\underline{u}^k = \underline{T}\underline{u}^{k-1} + \underline{c} \quad \forall k = 1, 2, 3, \dots \quad (21.12)$$

La convergencia a la solución la garantiza el siguiente teorema (véase [14]).

Teorema 122 *Si $\|\underline{T}\| < 1$, entonces el sistema lineal $\underline{u} = \underline{T}\underline{u} + \underline{c}$ tiene una solución única \underline{u}^* y las iteraciones \underline{u}^k definidas por la fórmula $\underline{u}^k = \underline{T}\underline{u}^{k-1} + \underline{c} \quad \forall k = 1, 2, 3, \dots$ convergen hacia la solución exacta \underline{u}^* para cualquier aproximación inicial \underline{u}^0 .*

Nótese que, mientras menor sea la norma de la matriz \underline{T} , más rápida es la convergencia, en el caso cuando $\|\underline{T}\|$ es menor que uno, pero cercano a uno, la convergencia es lenta y el número de iteraciones necesario para disminuir el error depende significativamente del error inicial. En este caso, es deseable proponer al vector inicial \underline{u}^0 de forma tal que sea mínimo el error inicial. Sin embargo, la elección de dicho vector no tiene importancia si la $\|\underline{T}\|$ es pequeña, ya que la convergencia es rápida.

Como es conocido, la velocidad de convergencia de los métodos iterativos dependen de las propiedades espectrales de la matriz de coeficientes del sistema de ecuaciones, cuando el operador diferencial \mathcal{L} de la ecuación del problema a resolver es auto-adjunto se obtiene una matriz simétrica y positivo definida y el número de condicionamiento de la matriz \underline{A} , es por definición

$$cond(\underline{A}) = \frac{\lambda_{\max}}{\lambda_{\min}} \geq 1 \quad (21.13)$$

donde λ_{\max} y λ_{\min} es el máximo y mínimo de los eigen-valores de la matriz \underline{A} . Si el número de condicionamiento es cercano a 1 los métodos numéricos al solucionar el problema convergerá en pocas iteraciones, en caso contrario se requerirán muchas iteraciones.

Frecuentemente al usar el método de Elemento Finito, Diferencias Finitas, entre otros, se tiene una velocidad de convergencia de $O\left(\frac{1}{h^2}\right)$ y en el caso de métodos de descomposición de dominio sin preconditionar se tiene una velocidad de convergencia de $O\left(\frac{1}{h}\right)$, donde h es la máxima distancia de separación entre nodos continuos de la partición, es decir, que poseen una pobre velocidad de convergencia cuando $h \rightarrow 0$ (véase [8], [9], [10] y [14]).

Los métodos aquí mencionados se colocaron en orden descendente en cuanto al consumo de recursos computacionales y ascendente en cuanto al aumento en la eficiencia en su desempeño, describiéndose a continuación:

21.2.1 Jacobi

Si todos los elementos de la diagonal principal de la matriz \underline{A} son diferentes de cero $a_{ii} \neq 0$ para $i = 1, 2, \dots, n$. Podemos dividir la i -ésima ecuación del sistema lineal (21.11) por a_{ii} para $i = 1, 2, \dots, n$, y después trasladamos todas las incógnitas, excepto x_i , a la derecha, se obtiene el sistema equivalente

$$\underline{u} = \underline{B}\underline{u} + \underline{d} \quad (21.14)$$

donde

$$d_i = \frac{b_i}{a_{ii}} \quad \text{y} \quad B = \{b_{ij}\} = \begin{cases} -\frac{a_{ij}}{a_{ii}} & \text{si } j \neq i \\ 0 & \text{si } j = i \end{cases}.$$

Las iteraciones del método de Jacobi están definidas por la fórmula

$$x_i = \sum_{j=1}^n b_{ij}x_j^{(k-1)} + d_i \quad (21.15)$$

donde $x_i^{(0)}$ son arbitrarias ($i = 1, 2, \dots, n; k = 1, 2, \dots$).

También el método de Jacobi se puede expresar en términos de matrices. Supongamos por un momento que la matriz \underline{A} tiene la diagonal unitaria, esto es $\text{diag}(\underline{A}) = \underline{I}$. Si descomponemos $\underline{A} = \underline{I} - \underline{B}$, entonces el sistema dado por la Ecs. (21.11) se puede reescribir como

$$(\underline{I} - \underline{B})\underline{u} = \underline{b}. \quad (21.16)$$

Para la primera iteración asumimos que $\underline{k} = \underline{b}$; entonces la última ecuación se escribe como $\underline{u} = \underline{B}\underline{u} + \underline{k}$. Tomando una aproximación inicial \underline{u}^0 , podemos obtener una mejor aproximación reemplazando \underline{u} por la más reciente aproximación de \underline{u}^m . Esta es la idea que subyace en el método Jacobi. El proceso iterativo queda como

$$\underline{u}^{m+1} = \underline{B}\underline{u}^m + \underline{k}. \quad (21.17)$$

La aplicación del método a la ecuación de la forma $\underline{A}\underline{u} = \underline{b}$, con la matriz \underline{A} no cero en los elementos diagonales, se obtiene multiplicando la Ec. (21.11) por $D^{-1} = [\text{diag}(\underline{A})]^{-1}$ obteniendo

$$\underline{B} = \underline{I} - \underline{D}^{-1}\underline{A}, \quad \underline{k} = \underline{D}^{-1}\underline{b}. \quad (21.18)$$

21.2.2 Gauss-Seidel

Este método es una modificación del método Jacobi, en el cual una vez obtenido algún valor de \underline{u}^{m+1} , este es usado para obtener el resto de los valores utilizando los valores más actualizados de \underline{u}^{m+1} . Así, la Ec. (21.17) puede ser escrita como

$$u_i^{m+1} = \sum_{j<i} b_{ij}u_j^{m+1} + \sum_{j>i} b_{ij}u_j^m + k_i. \quad (21.19)$$

Notemos que el método Gauss-Seidel requiere el mismo número de operaciones aritméticas por iteración que el método de Jacobi. Este método se escribe en forma matricial como

$$\underline{u}^{m+1} = \underline{\underline{E}}\underline{u}^{m+1} + \underline{\underline{F}}\underline{u}^m + \underline{k} \quad (21.20)$$

donde $\underline{\underline{E}}$ y $\underline{\underline{F}}$ son las matrices triangular superior e inferior respectivamente. Este método mejora la convergencia con respecto al método de Jacobi en un factor aproximado de 2.

21.2.3 Richardson

Escribiendo el método de Jacobi como

$$\underline{u}^{m+1} - \underline{u}^m = \underline{b} - \underline{\underline{A}}\underline{u}^m \quad (21.21)$$

entonces el método Richardson se genera al incorporar la estrategia de sobre-relajación de la forma siguiente

$$\underline{u}^{m+1} = \underline{u}^m + \omega (\underline{b} - \underline{\underline{A}}\underline{u}^m). \quad (21.22)$$

El método de Richardson se define como

$$\underline{u}^{m+1} = (\underline{\underline{I}} - \omega \underline{\underline{A}}) \underline{u}^m + \omega \underline{b} \quad (21.23)$$

en la práctica encontrar el valor de ω puede resultar muy costoso computacionalmente y las diversas estrategias para encontrar ω dependen de las características propias del problema, pero este método con un valor ω óptimo resulta mejor que el método de Gauss-Seidel.

21.2.4 Relajación Sucesiva

Partiendo del método de Gauss-Seidel y sobrerrelajando este esquema, obtenemos

$$u_i^{m+1} = (1 - \omega) u_i^m + \omega \left[\sum_{j=1}^{i-1} b_{ij} u_j^{m+1} + \sum_{j=i+1}^N b_{ij} u_j^m + k_i \right] \quad (21.24)$$

y cuando la matriz $\underline{\underline{A}}$ es simétrica con entradas en la diagonal positivas, éste método converge si y sólo si $\underline{\underline{A}}$ es definida positiva y $\omega \in (0, 2)$. En la práctica encontrar el valor de ω puede resultar muy costoso computacionalmente y las diversas estrategias para encontrar ω dependen de las características propias del problema.

Espacio de Krylov Los métodos Jacobi, Gauss-Seidel y de Relajación Sucesiva (SOR) son usualmente menos eficientes que los métodos discutidos en el resto de esta sección basados en el espacio de Krylov (véase [53] y [13]). Estos métodos minimizan, en la k -ésima iteración alguna medida de error sobre el espacio afín $\underline{x}_0 + \mathcal{K}_k$, donde \underline{x}_0 es la iteración inicial y \mathcal{K}_k es el k -ésimo subespacio de Krylov

$$\mathcal{K}_k = \text{Generado} \{ \underline{r}_0, \underline{\underline{A}} \underline{r}_0, \dots, \underline{\underline{A}}^{k-1} \underline{r}_0 \} \text{ para } k \geq 1. \quad (21.25)$$

El residual es $\underline{r} = \underline{b} - \underline{\underline{A}} \underline{x}$, tal $\{ \underline{r}_k \}_{k \geq 0}$ denota la sucesión de residuales

$$\underline{r}_k = \underline{b} - \underline{\underline{A}} \underline{x}_k. \quad (21.26)$$

Entre los métodos más usados definidos en el espacio de Krylov para el tipo de problemas tratados en el presente trabajo se puede considerar: Método de Gradiente Conjugado -para matrices simétricas- y GMRES -para matrices no simétricas-.

El método del Gradiente Conjugado ha recibido mucha atención en su uso al resolver ecuaciones diferenciales parciales y ha sido ampliamente utilizado en años recientes por la notoria eficiencia al reducir considerablemente en número de iteraciones necesarias para resolver el sistema algebraico de ecuaciones. Aunque los pioneros de este método fueron Hestenes y Stiefel (1952), el interés actual arranca a partir de que Reid (1971) lo planteara como un método iterativo, que es la forma en que se le usa con mayor frecuencia en la actualidad, esta versión está basada en el desarrollo hecho en [?].

La idea básica en que descansa el método del Gradiente Conjugado consiste en construir una base de vectores ortogonales y utilizarla para realizar la búsqueda de la solución en forma más eficiente. Tal forma de proceder generalmente no sería aconsejable porque la construcción de una base ortogonal utilizando el procedimiento de Gram-Schmidt requiere, al seleccionar cada nuevo elemento de la base, asegurar su ortogonalidad con respecto a cada uno de los vectores construidos previamente. La gran ventaja del método de Gradiente Conjugado radica en que cuando se utiliza este procedimiento, basta con asegurar la ortogonalidad de un nuevo miembro con respecto al último que se ha construido, para que automáticamente esta condición se cumpla con respecto a todos los anteriores.

Definición 123 Una matriz $\underline{\underline{A}}$ es llamada positiva definida si todos sus eigen-valores tienen parte real positiva o equivalentemente, si $\underline{u}^T \underline{\underline{A}} \underline{u}$ tiene parte real positiva para $\underline{u} \in \mathbb{C} \setminus \{0\}$. Notemos en este caso que

$$\underline{u}^T \underline{\underline{A}} \underline{u} = \underline{u}^T \frac{\underline{\underline{A}} + \underline{\underline{A}}^T}{2} \underline{u} > 0, \text{ con } \underline{u} \in \mathbb{R}^n \setminus \{0\}.$$

21.2.5 Método de Gradiente Conjugado

Si la matriz generada por la discretización es simétrica $\underline{\underline{A}} = \underline{\underline{A}}^T$ y definida positiva $-\underline{u}^T \underline{\underline{A}} \underline{u} > 0$ para todo $\underline{u} \neq 0$, entonces es aplicable el método de Gradiente Conjugado -Conjugate Gradient Method (CGM)-. La idea básica en que descansa el método del Gradiente Conjugado consiste en construir una base de vectores ortogonales espacio de Krylov $\mathcal{K}_n(\underline{\underline{A}}, \underline{v}^n)$ y utilizarla para realizar la búsqueda de la solución en forma lo más eficiente posible.

Tal forma de proceder generalmente no sería aconsejable porque la construcción de una base ortogonal utilizando el procedimiento de Gram-Schmidt requiere, al seleccionar cada nuevo elemento de la base, asegurar su ortogonalidad con respecto a cada uno de los vectores construidos previamente. La gran ventaja del método de Gradiente Conjugado radica en que cuando se utiliza este procedimiento, basta con asegurar la ortogonalidad de un nuevo miembro con respecto al último que se ha construido, para que automáticamente esta condición se cumpla con respecto a todos los anteriores.

En el algoritmo de Gradiente Conjugado, se toma a la matriz $\underline{\underline{A}}$ como simétrica y positiva definida, y como datos de entrada del sistema

$$\underline{\underline{A}} \underline{u} = \underline{f} \tag{21.27}$$

el vector de búsqueda inicial \underline{u}^0 y se calcula $\underline{r}^0 = \underline{f} - \underline{A}\underline{u}^0$, $\underline{p}^0 = \underline{r}^0$, quedando el método numérico esquemáticamente como:

$$\begin{aligned}
 \alpha^n &= \frac{\langle \underline{p}^n, \underline{p}^n \rangle}{\langle \underline{p}^n, \underline{A}\underline{p}^n \rangle} \\
 \underline{u}^{n+1} &= \underline{u}^n + \alpha^n \underline{p}^n \\
 \underline{r}^{n+1} &= \underline{r}^n - \alpha^n \underline{A}\underline{p}^n \\
 &\text{Prueba de convergencia} \\
 \beta^n &= \frac{\langle \underline{r}^{n+1}, \underline{r}^{n+1} \rangle}{\langle \underline{r}^n, \underline{r}^n \rangle} \\
 \underline{p}^{n+1} &= \underline{r}^{n+1} + \beta^n \underline{p}^n \\
 n &= n + 1
 \end{aligned} \tag{21.28}$$

donde $\langle \cdot, \cdot \rangle = (\cdot, \cdot)$ será el producto interior adecuado al sistema lineal en particular, la solución aproximada será \underline{u}^{n+1} y el vector residual será \underline{r}^{n+1} .

En la implementación numérica y computacional del método es necesario realizar la menor cantidad de operaciones posibles por iteración, en particular en $\underline{A}\underline{p}^n$, una manera de hacerlo queda esquemáticamente como:

Dado el vector de búsqueda inicial \underline{u} , calcula $\underline{r} = \underline{f} - \underline{A}\underline{u}$, $\underline{p} = \underline{r}$ y $\mu = \underline{r} \cdot \underline{r}$.

$$\begin{aligned}
 &\text{Para } n = 1, 2, \dots, \text{Mientras } (\mu < \varepsilon) \{ \\
 &\quad \underline{v} = \underline{A}\underline{p} \\
 &\quad \alpha = \frac{\mu}{\underline{p} \cdot \underline{v}} \\
 &\quad \underline{u} = \underline{u} + \alpha \underline{p} \\
 &\quad \underline{r} = \underline{r} - \alpha \underline{v} \\
 &\quad \mu' = \underline{r} \cdot \underline{r} \\
 &\quad \beta = \frac{\mu'}{\mu} \\
 &\quad \underline{p} = \underline{r} + \beta \underline{p} \\
 &\quad \mu = \mu' \\
 &\}
 \end{aligned}$$

La solución aproximada será \underline{u} y el vector residual será \underline{r} .

Si se denota con $\{\lambda_i, V_i\}_{i=1}^N$ las eigen-soluciones de \underline{A} , i.e. $\underline{A}V_i = \lambda_i V_i$, $i = 0, 1, 2, \dots, N$. Ya que la matriz \underline{A} es simétrica, los eigen-valores son reales y se pueden ordenar $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_N$. Se define el número de condición por $Cond(\underline{A}) = \lambda_N / \lambda_1$ y la norma de la energía asociada a \underline{A} por $\|\underline{u}\|_{\underline{A}}^2 = \underline{u} \cdot \underline{A}\underline{u}$ entonces

$$\|\underline{u} - \underline{u}^k\|_{\underline{A}} \leq \|\underline{u} - \underline{u}^0\|_{\underline{A}} \left[\frac{1 - \sqrt{Cond(\underline{A})}}{1 + \sqrt{Cond(\underline{A})}} \right]^{2k}. \tag{21.29}$$

El siguiente teorema da idea del espectro de convergencia del sistema $\underline{\underline{A}}u = \underline{\underline{b}}$ para el método de Gradiente Conjugado.

Teorema 124 Sea $\kappa = \text{cond}(\underline{\underline{A}}) = \frac{\lambda_{\max}}{\lambda_{\min}} \geq 1$, entonces el método de Gradiente Conjugado satisface la $\underline{\underline{A}}$ -norma del error dado por

$$\frac{\|e^n\|}{\|e^0\|} \leq \frac{2}{\left[\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1} \right)^n + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1} \right)^{-n} \right]} \leq 2 \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \right)^n \quad (21.30)$$

donde $\underline{\underline{e}}^m = \underline{\underline{u}} - \underline{\underline{u}}^m$ del sistema $\underline{\underline{A}}u = \underline{\underline{b}}$.

Nótese que para κ grande se tiene que

$$\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \simeq 1 - \frac{2}{\sqrt{\kappa}} \quad (21.31)$$

tal que

$$\|\underline{\underline{e}}^n\|_{\underline{\underline{A}}} \simeq \|\underline{\underline{e}}^0\|_{\underline{\underline{A}}} \exp\left(-2 \frac{n}{\sqrt{\kappa}}\right) \quad (21.32)$$

de lo anterior se puede esperar un espectro de convergencia del orden de $O(\sqrt{\kappa})$ iteraciones (véase [14] y [53]).

21.2.6 Gradiente Conjugado Precondicionado

Cuando la matriz $\underline{\underline{A}}$ es simétrica y definida positiva se puede escribir como

$$\lambda_1 \leq \frac{\underline{\underline{u}} \underline{\underline{A}} \cdot \underline{\underline{u}}}{\underline{\underline{u}} \cdot \underline{\underline{u}}} \leq \lambda_n \quad (21.33)$$

y tomando la matriz $\underline{\underline{C}}^{-1}$ como un preconditionador de $\underline{\underline{A}}$ con la condición de que

$$\lambda_1 \leq \frac{\underline{\underline{u}} \underline{\underline{C}}^{-1} \underline{\underline{A}} \cdot \underline{\underline{u}}}{\underline{\underline{u}} \cdot \underline{\underline{u}}} \leq \lambda_n \quad (21.34)$$

entonces la Ec. (21.27) se puede escribir como

$$\underline{\underline{C}}^{-1} \underline{\underline{A}}u = \underline{\underline{C}}^{-1}b \quad (21.35)$$

donde $\underline{\underline{C}}^{-1}\underline{\underline{A}}$ es también simétrica y definida positiva en el producto interior $\langle \underline{u}, \underline{v} \rangle = \underline{u} \cdot \underline{\underline{C}}\underline{v}$, porque

$$\begin{aligned} \langle \underline{u}, \underline{\underline{C}}^{-1}\underline{\underline{A}}\underline{v} \rangle &= \underline{u} \cdot \underline{\underline{C}} (\underline{\underline{C}}^{-1}\underline{\underline{A}}\underline{v}) \\ &= \underline{u} \cdot \underline{\underline{A}}\underline{v} \end{aligned} \quad (21.36)$$

que por hipótesis es simétrica y definida positiva en ese producto interior.

La elección del producto interior $\langle \cdot, \cdot \rangle$ quedará definido como

$$\langle \underline{u}, \underline{v} \rangle = \underline{u} \cdot \underline{\underline{C}}^{-1}\underline{\underline{A}}\underline{v} \quad (21.37)$$

por ello las Ecs. (21.28[1]) y (21.28[3]), se convierten en

$$\alpha^{k+1} = \frac{\underline{r}^k \cdot \underline{r}^k}{\underline{p}^{k+1} \cdot \underline{\underline{C}}^{-1}\underline{p}^{k+1}} \quad (21.38)$$

y

$$\beta^{k+1} = \frac{\underline{p}^k \cdot \underline{\underline{C}}^{-1}\underline{r}^k}{\underline{p}^k \cdot \underline{\underline{A}}\underline{p}^k} \quad (21.39)$$

generando el método de Gradiente Conjugado preconditionado con preconditionador $\underline{\underline{C}}^{-1}$. Es necesario hacer notar que los métodos Gradiente Conjugado y Gradiente Conjugado Precondicionado sólo difieren en la elección del producto interior.

Para el método de Gradiente Conjugado Precondicionado, los datos de entrada son un vector de búsqueda inicial \underline{u}^0 y el preconditionador $\underline{\underline{C}}^{-1}$. Calculandose $\underline{r}^0 = \underline{b} - \underline{\underline{A}}\underline{u}^0$, $\underline{p} = \underline{\underline{C}}^{-1}\underline{r}^0$, quedando el método esquemáticamente como:

$$\begin{aligned} \beta^{k+1} &= \frac{\underline{p}^k \cdot \underline{\underline{C}}^{-1}\underline{r}^k}{\underline{p}^k \cdot \underline{\underline{A}}\underline{p}^k} \\ \underline{p}^{k+1} &= \underline{r}^k - \beta^{k+1}\underline{p}^k \\ \alpha^{k+1} &= \frac{\underline{r}^k \cdot \underline{r}^k}{\underline{p}^{k+1} \cdot \underline{\underline{C}}^{-1}\underline{p}^{k+1}} \\ \underline{u}^{k+1} &= \underline{u}^k + \alpha^{k+1}\underline{p}^{k+1} \\ \underline{r}^{k+1} &= \underline{\underline{C}}^{-1}\underline{r}^k - \alpha^{k+1}\underline{\underline{A}}\underline{p}^{k+1}. \end{aligned} \quad (21.40)$$

Algoritmo Computacional del Método Dado el sistema $\underline{A}u = \underline{b}$, con la matriz \underline{A} simétrica y definida positiva de dimensión $n \times n$. La entrada al método será una elección de \underline{u}^0 como condición inicial, $\varepsilon > 0$ como la tolerancia del método, N como el número máximo de iteraciones y la matriz de preconditionamiento \underline{C}^{-1} de dimensión $n \times n$, el algoritmo del método de Gradiente Conjugado Precondicionado queda como:

$$\begin{aligned} \underline{r} &= \underline{b} - \underline{A}u \\ \underline{w} &= \underline{C}^{-1}\underline{r} \\ \underline{v} &= (\underline{C}^{-1})^T \underline{w} \\ \alpha &= \sum_{j=1}^n w_j^2 \\ k &= 1 \end{aligned}$$

Mientras que $k \leq N$

Si $\|\underline{v}\|_\infty < \varepsilon$ Salir

$$\underline{x} = \underline{A}v$$

$$t = \frac{\alpha}{\sum_{j=1}^n v_j x_j}$$

$$\underline{u} = \underline{u} + tv$$

$$\underline{r} = \underline{r} - t\underline{x}$$

$$\underline{w} = \underline{C}^{-1}\underline{r}$$

$$\beta = \sum_{j=1}^n w_j^2$$

Si $\|\underline{r}\|_\infty < \varepsilon$ Salir

$$s = \frac{\beta}{\alpha}$$

$$\underline{v} = (\underline{C}^{-1})^T \underline{w} + s\underline{v}$$

$$\alpha = \beta$$

$$k = k + 1$$

La salida del método será la solución aproximada $\underline{u} = (u_1, \dots, u_n)$ y el residual $\underline{r} = (r_1, \dots, r_n)$.

En el caso del método sin preconditionamiento, \underline{C}^{-1} es la matriz identidad, que para propósitos de optimización sólo es necesario hacer la asignación de vectores correspondiente en lugar del producto de la matriz por el vector. En el caso de que la matriz \underline{A} no sea simétrica, el método de Gradiente

Conjugado puede extenderse para soportarlas, para más información sobre pruebas de convergencia, resultados numéricos entre los distintos métodos de solución del sistema algebraico $\underline{A}u = \underline{b}$ generada por la discretización de un problema elíptico y cómo extender estos para matrices no simétricas ver [?] y [?].

Teorema 125 Sean $\underline{A}, \underline{B}$ y \underline{C} tres matrices simétricas y positivas definidas entonces

$$\kappa(\underline{C}^{-1}\underline{A}) \leq \kappa(\underline{C}^{-1}\underline{B}) \kappa(\underline{B}^{-1}\underline{A}).$$

21.2.7 Método Residual Mínimo Generalizado

Si la matriz generada por la discretización es no simétrica, entonces una opción, es el método Residual Mínimo Generalizado -Generalized Minimum Residual Method (GMRES)-, este representa una formulación iterativa común satisfaciendo una condición de optimización. La idea básica detrás del método se basa en construir una base ortonormal

$$\{\underline{v}^1, \underline{v}^2, \dots, \underline{v}^n\} \quad (21.41)$$

para el espacio de Krylov $\mathcal{K}_n(\underline{A}, \underline{v}^n)$. Para hacer \underline{v}^{n+1} ortogonal a $\mathcal{K}_n(\underline{A}, \underline{v}^n)$, es necesario usar todos los vectores previamente construidos $\{\underline{v}^{n+1j}\}_{j=1}^n$ -en la práctica sólo se guardan algunos vectores anteriores- en los cálculos. Y el algoritmo se basa en una modificación del método de Gram-Schmidt para la generación de una base ortonormal. Sea $\underline{V}_n = [\underline{v}^1, \underline{v}^2, \dots, \underline{v}^n]$ la cual denota la matriz conteniendo \underline{v}^j en la j -ésima columna, para $j = 1, 2, \dots, n$, y sea $\underline{H}_n = [h_{i,j}]$, $1 \leq i, j \leq n$, donde las entradas de \underline{H}_n no especificadas en el algoritmo son cero. Entonces, \underline{H}_n es una matriz superior de Hessenberg. i.e. $h_{ij} = 0$ para $j < i - 1$, y

$$\begin{aligned} \underline{A}\underline{V}_n &= \underline{V}_n\underline{H}_n + h_{n+1,n} [0, \dots, 0, \underline{v}^{n+1}] \\ \underline{H}_n &= \underline{H}_n^T \underline{A}\underline{V}_n. \end{aligned} \quad (21.42)$$

En el algoritmo del método Residual Mínimo Generalizado, la matriz \underline{A} es tomada como no simétrica, y como datos de entrada del sistema

$$\underline{A}u = \underline{f} \quad (21.43)$$

el vector de búsqueda inicial \underline{u}^0 y se calcula $\underline{r}^0 = \underline{f} - \underline{A}\underline{u}^0$, $\beta^0 = \|\underline{r}^0\|$, $\underline{v}^1 = \underline{r}^0/\beta^0$, quedando el método esquemáticamente como:

$$\begin{aligned}
 &\text{Para } n = 1, 2, \dots, \text{Mientras } \beta^n < \tau\beta^0 \{ \\
 &\quad \underline{w}_0^{n+1} = \underline{A}\underline{v}^n \\
 &\quad \text{Para } l = 1 \text{ hasta } n \{ \\
 &\quad\quad h_{l,n} = \langle \underline{w}_l^{n+1}, \underline{v}^l \rangle \\
 &\quad\quad \underline{w}_{l+1}^{n+1} = \underline{w}_l^{n+1} - h_{l,n}\underline{v}^l \\
 &\quad\quad \} \\
 &\quad h_{n+1,n} = \|\underline{w}_{n+1}^{n+1}\| \\
 &\quad \underline{v}^{n+1} = \underline{w}_{n+1}^{n+1}/h_{n+1,n} \\
 &\quad \text{Calcular } \underline{y}^n \text{ tal que } \beta^n = \|\beta^0 \underline{e}_1 - \hat{\underline{H}}_n \underline{y}^n\| \text{ es mínima} \\
 &\quad \}
 \end{aligned} \tag{21.44}$$

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

$$\underline{r}^n = \underline{r}^0 - \underline{A}\underline{V}_n \underline{y}^n = \underline{V}_{n+1} \left(\beta^0 \underline{e}_1 - \hat{\underline{H}}_n \underline{y}^n \right). \tag{21.45}$$

Teorema 126 Sea \underline{u}^k la iteración generada después de k iteraciones de GM-RES, con residual \underline{r}^k . Si la matriz \underline{A} es diagonalizable, i.e. $\underline{A} = \underline{V}\underline{\Lambda}\underline{V}^{-1}$ donde $\underline{\Lambda}$ es una matriz diagonal de eigen-valores de \underline{A} , y \underline{V} es la matriz cuyas columnas son los eigen-vectores, entonces

$$\frac{\|\underline{r}^k\|}{\|\underline{r}^0\|} \leq \kappa(V) \min_{p_\kappa \in \Pi_\kappa, p_\kappa(0)=1} \max_{\lambda_j} |p_\kappa(\lambda_j)| \tag{21.46}$$

donde $\kappa(V) = \frac{\|\underline{V}\|}{\|\underline{V}^{-1}\|}$ es el número de condicionamiento de \underline{V} .

21.3 Algunos Ejemplos

Sea el sistema lineal $\underline{A}u = \underline{f}$ dado por

$$\begin{bmatrix} 4 & 3 & 0 \\ 3 & 4 & 1 \\ 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 24 \\ 20 \\ -24 \end{bmatrix}$$

cuya solución es $\begin{bmatrix} 3 \\ 4 \\ -5 \end{bmatrix}$, si usamos métodos directos obtenemos los siguientes resultados:

- Solución usando el método Factorización LU:

$$\begin{bmatrix} +3.0000000000e + 00 \\ +4.0000000000e + 00 \\ -5.0000000000e + 00 \end{bmatrix}$$

- Solución usando el método Tridiagonal:

$$\begin{bmatrix} +3.0000000000e + 00 \\ +4.0000000000e + 00 \\ -5.0000000000e + 00 \end{bmatrix}$$

- Solución usando el método Choleski:

$$\begin{bmatrix} +3.0000000000e + 00 \\ +4.0000000000e + 00 \\ -5.0000000000e + 00 \end{bmatrix}$$

- Solución usando el método de la Inversa:

$$\begin{bmatrix} +3.0000000000e + 00 \\ +4.0000000000e + 00 \\ -5.0000000000e + 00 \end{bmatrix}$$

Si ahora usamos métodos iterativos (con tolerancia $1e - 6$) obtenemos:

- Solución usando el método Jacobi, iteraciones necesarias para resolver el sistema lineal: 59

$$\begin{bmatrix} +3.0000036111e + 00 \\ +4.0000042130e + 00 \\ -5.0000012037e + 00 \end{bmatrix}$$

- Solución usando el método Gauss-Seidel, iteraciones necesarias para resolver el sistema lineal: 25

$$\begin{bmatrix} +3.0000151461e + 00 \\ +3.9999873782e + 00 \\ -5.0000031554e + 00 \end{bmatrix}$$

- Solución usando el método CGM, iteraciones necesarias para resolver el sistema lineal: 3

$$\begin{bmatrix} +3.0000000000e + 00 \\ +4.0000000000e + 00 \\ -5.0000000000e + 00 \end{bmatrix}$$

Esto nos muestra que para ciertos sistemas lineales (en apariencia inofensivo) algunos métodos iterativos pueden requerir una gran cantidad de iteraciones para converger y en algunos casos, esto ni siquiera está garantizado, es por ello que debemos elegir el método más adecuado para el tipo de sistema lineal con el que se trabaje:

- En los métodos directos la solución \underline{u} se obtiene en un número fijo de pasos y sólo están sujetos a los errores de redondeo.
- En los métodos iterativos, se realizan iteraciones para aproximarse a la solución \underline{u} aprovechando las características propias de la matriz \underline{A} , tratando de usar un menor número de pasos que en un método directo.

Por lo general, es conveniente usar bibliotecas⁴⁰ para implementar de forma eficiente a los vectores, matrices -bandadas, dispersas o ralas- y resolver el sistemas lineal.

⁴⁰ Algunas de las bibliotecas más usadas para resolver sistemas lineales usando matrices bandadas y dispersas son PETCS, HYPRE, ATLAS, LAPACK++, LAPACK, EISPACK, LINPACK, BLAS, entre muchas otras alternativas, tanto para implementaciones secuenciales como paralelas y más recientemente para hacer uso de los procesadores CUDA en las GPU de nVidia.

Los métodos iterativos rara vez se usan para resolver sistemas lineales de dimensión pequeña (el concepto de dimensión pequeña es muy relativo), ya que el tiempo necesario para conseguir una exactitud satisfactoria rebasa el que requieren los métodos directos. Sin embargo, en el caso de sistemas grandes con un alto porcentaje de elementos cero, son eficientes tanto en el almacenamiento en la computadora como en el tiempo que se invierte en su solución. Por ésta razón al resolver éstos sistemas algebraicos de ecuaciones es preferible aplicar métodos iterativos tal como Gradiente Conjugado -Conjugate Gradient Method (CGM)- o Residual Mínimo Generalizado -Generalized Minimum Residual Method (GMRES) según sea el tipo de matriz generada.

21.3.1 Usando Python

En Python es común usar *numpy* para definir un matrices y vectores además de sus operaciones relacionadas, para ello podemos usar:

```
import numpy as np
A = np.matrix([[4, 3, 0], [3, 4, 1], [0, -1, 4]])
b = np.matrix([[24], [20], [-24]])
```

Antes de resolver un sistema lineal, es conveniente validar que una matriz tenga inversa. Para que tenga inversa, la matriz no tiene que tener vectores linealmente independientes, es decir, no debe haber filas o columnas que puedan ser escritas como la combinación de otras filas o columnas. Para ello podemos usar:

```
import numpy as np
A = np.array(
[[0,1,0,0],
[0,0,1,0],
[0,1,1,0],
[1,0,0,1]]
)
lambdas, V = np.linalg.eig(A.T)
print(A[lambdas == 0, :])
```

visualizando las entradas que son linealmente independientes y cuáles no:

```
[[0 1 1 0]]
```

En Python hay diversas formas de resolver un sistema lineal, por ejemplo, si usamos el cálculo de la inversa A^{-1} :

```
import numpy as np
A = np.matrix([[4, 3, 0], [3, 4, 1], [0, -1, 4]])
b = np.matrix([[24], [20], [-24]])
if np.linalg.det(A) == 0:
    x = None
    print("No se puede resolver")
else:
    x = (A**-1)*b
    print(x)
    print(np.dot(A, x))
    print((np.dot(A, x) == b).all())
```

que nos entregará la siguiente salida:

```
[[3.]
 [4.]
 [-5.]]
[[24.]
 [20.]
 [-24]]
True
```

El primer vector es la solución del sistema lineal, la comprobación y la revisión de la igualdad entrada a entrada entre la solución y el vector b ⁴¹.

Otro ejemplo usando *linalg.solve*:

```
import numpy as np
A = np.array([[4, 3, 0], [3, 4, 1], [0, -1, 4]])
b = np.array([[24], [20], [-24]])
x = np.linalg.solve(A, b)
print(x)
```

⁴¹En este caso las soluciones son idénticas, pero en general pueden diferir en algunos dígitos por la pérdida de precisión, por lo que este ejemplo es solo es ilustrativo.

Otro ejemplo usando *linalg.inv*:

```
import numpy as np
A = np.array([[4, 3, 0], [3, 4, 1], [0, -1, 4]])
b = np.array([[24], [20], [-24]])
x = np.linalg.inv(A).dot(b)
print(x)
```

Otro ejemplo usando *linalg.pinv*:

```
import numpy as np
A = np.array([[4, 3, 0], [3, 4, 1], [0, -1, 4]])
b = np.array([[24], [20], [-24]])
x = np.linalg.pinv(A).dot(b)
print(x)
```

Otro ejemplo usando *linalg.qr*:

```
import numpy as np
A = np.array([[4, 3, 0], [3, 4, 1], [0, -1, 4]])
b = np.array([[24], [20], [-24]])
Q, R = np.linalg.qr(A)
y = np.dot(Q.T, b)
x = np.linalg.solve(R, y)
print(x)
```

Otro ejemplo usando *sympy*:

```
from sympy import symbols, Matrix, linsolve
x, y, z = symbols('x, y, z')
A = Matrix([[4, 3, 0], [3, 4, 1], [0, -1, 4]])
b = Matrix([[24], [20], [-24]])
xx = linsolve((A,b),[x, y, z])
print(xx)
```

21.3.2 Usando C++

GMM++⁴² es una librería para C++ que permite definir diversos tipos de matrices y vectores además operaciones básicas de álgebra lineal. La facilidad de uso y la gran cantidad de opciones hacen que GMM++ sea una buena opción para trabajar con operaciones elementales de álgebra lineal.

Se instala en Debian Linux y/o Ubuntu como:

```
# apt install libgmm++-dev
```

Para compilar el ejemplo usar:

```
$ g++ ejemplito.cpp
```

Para ejecutar usar:

```
$ ./a.out
```

Ejemplo 1 *Un sencillo ejemplo de manejo de matrices y vectores en C++:*

```
#include <gmm/gmm.h>
#include <math.h>
int main(void)
{
    int N = 100;
    // Matriz densa
    gmm::dense_matrix<double> AA(N, N);
    // Matriz dispersa
    gmm::row_matrix< gmm::rsvector<double> > A(N, N);
    // Vectores
    std::vector<double> x(N), b(N);
    int i;
    double P = -2 ;
    double Q = 1;
    double R = 1;
    A(0, 0) = P; // Primer renglon de la matriz A y vector b
    A(0, 1) = Q;
    b[0] = P;
    // Renglones intermedios de la matriz A y vector b
    for(i = 1; i < N - 1; i++)
    {
        A(i, i - 1) = R;
        A(i, i) = P;
    }
}
```

⁴²GMM++ [<http://getfem.org/gmm.html>]

```

    A(i, i + 1) = Q;
    b[i] = P;
}
A(N - 1, N - 2) = R; // Renglon final de la matriz A y vector b
A(N - 1, N - 1) = P;
b[N - 1] = P;
// Copia la matriz dispersa a la densa para usarla en LU
gmm::copy(A,AA);
// Visualiza la matriz y el vector
std::cout << "Matriz A" << AA << gmm::endl;
std::cout << "Vector b" << b << gmm::endl;
return 0;
}

```

21.4 Cómputo Paralelo

Los modelos matemáticos de muchos sistemas en Ciencia e Ingeniería requieren el procesamiento de sistemas algebraicos de gran escala. Pero cuando los tiempos de solución del sistema algebraico es excesivo o cuando un solo equipo de cómputo no puede albergar nuestro problema, entonces pensamos en el cómputo en paralelo.

La computación paralela es el uso de múltiples recursos computacionales para resolver un problema. Se distingue de la computación secuencial en que varias operaciones pueden ocurrir simultáneamente. Los sistemas de cómputo con procesamiento en paralelo surgen de la necesidad de resolver problemas complejos en un tiempo razonable, utilizando las ventajas de memoria, velocidad de los procesadores, formas de interconexión de estos y distribución de la tarea, a los que en su conjunto denominamos arquitectura en paralelo. Entenderemos por una arquitectura en paralelo a un conjunto de procesadores interconectados capaces de cooperar en la solución de un problema.

Así, para resolver un problema en particular, se usa una arquitectura o combinación de múltiples arquitecturas (topologías), ya que cada una ofrece ventajas y desventajas que tienen que ser sopesadas antes de implementar la solución del problema en una arquitectura en particular. También es necesario conocer los problemas a los que se enfrenta un desarrollador de programas que se desean correr en paralelo, como son: el partir eficientemente

un problema en múltiples tareas y cómo distribuir estas según la arquitectura en particular con que se trabaje.

El paralelismo clásico, o puesto de otra manera, el clásico uso del paralelismo, es el de diseño de programas eficientes en el ámbito científico. La simulación de problemas científicos es un área de gran importancia, los cuales requieren de una gran capacidad de procesamiento y de espacio de memoria, debido a las complejas operaciones que se deben realizar.

Tradicionalmente, los programas informáticos se han escrito para el cómputo en serie. Para resolver un problema, se construye un algoritmo y se implementa como un flujo en serie de instrucciones. Estas instrucciones se ejecutan en una unidad central de procesamiento en un ordenador. Sólo puede ejecutarse una instrucción a la vez y un tiempo después de que la instrucción ha terminado, se ejecuta la siguiente.

La computación en paralelo, por el contrario, utiliza simultáneamente múltiples elementos de procesamiento para resolver un problema. Esto se logra mediante la división del problema en partes independientes de modo que cada elemento de procesamiento pueda ejecutar su parte del algoritmo de manera simultánea con los otros. Los elementos de procesamiento son diversos e incluyen recursos tales como una computadora con múltiples procesadores, varios ordenadores en red, Hardware especializado, o cualquier combinación de los anteriores.

Los programas informáticos paralelos son más difíciles de escribir que los secuenciales, porque la concurrencia introduce nuevos tipos de errores de Software, siendo las condiciones de carrera las más comunes. La comunicación y sincronización entre diferentes subtareas son algunos de los mayores obstáculos para obtener un buen rendimiento del programa paralelo.

Equipo Paralelo de Memoria Compartida un multiprocesador puede verse como una computadora paralela compuesta por varios procesadores interconectados que comparten un mismo sistema de memoria (todos los equipos de cómputo actuales son de este tipo). Tienen un único espacio de direcciones para todos los procesadores. Para hacer uso de la memoria compartida (que puede ser de hasta Terabytes) por más de un procesador, se requiere hacer uso de técnicas de semáforos que mantienen la integridad de la memoria; esta arquitectura no puede crecer mucho en el número de procesadores interconectados por la saturación rápida del bus.

Equipo Paralelo de Memoria Distribuida los sistemas multicomputadoras se pueden ver como una computadora paralela en el cual cada procesador tiene su propia memoria local. En estos sistemas la memoria se encuentra distribuida y no compartida como en los sistemas multiprocesador. Los procesadores se comunican a través de paso de mensajes, ya que éstos sólo tienen acceso directo a su memoria local y no a las memorias del resto de los procesadores.

La transferencia de los datos se realiza a través de la red de interconexión que conecta un subconjunto de procesadores con otro subconjunto. La transferencia de unos procesadores a otros se realiza por múltiples transferencias entre procesadores conectados dependiendo del establecimiento de dicha red.

Dado que la memoria está distribuida entre los diferentes elementos de proceso, estos sistemas reciben el nombre de distribuidos. Por otra parte, estos sistemas son débilmente acoplados, ya que los módulos funcionan de forma casi independiente unos de otros. Este tipo de memoria distribuida es de acceso lento por ser peticiones a través de la red, pero es una forma muy efectiva de tener acceso a un gran volumen de memoria.

Equipo Paralelo de Memoria Compartida-Distribuida La tendencia actual en las máquinas paralelas es de aprovechar las facilidades de programación que ofrecen los ambientes de memoria compartida y la escalabilidad de los ambientes de memoria distribuida. En este modelo se conectan entre sí módulos de multiprocesadores, pero se mantiene la visión global de la memoria a pesar de que es distribuida. Ejemplo de este tipo de equipo son los Clusters. El desarrollo de sistemas operativos y compiladores del dominio público (Linux y Software GNU), estándares para interfaz de paso de mensajes (Message Passing Interface MPI), conexión universal a periféricos (Peripheral Component Interconnect PCI), entre otros, han hecho posible tomar ventaja de los recursos económicos computacionales de producción masiva (procesadores, discos, redes).

La principal desventaja que presenta a los proveedores de multicomputadoras es que deben satisfacer una amplia gama de usuarios, es decir, deben ser generales. Esto aumenta los costos de diseños y producción de equipos, así como los costos de desarrollo de Software que va con ellos: sistema operativo, compiladores y aplicaciones. Todos estos costos deben ser añadidos cuando se hace una venta. Por supuesto alguien que sólo necesita procesadores y un mecanismo de pase de mensajes no debería pagar por todos estos añadi-

dos que nunca usará. Estos usuarios son los que están impulsando el uso de Clusters principalmente de computadoras personales (PC)

¿Cómo Paralelizo mi Programa? El problema de todos los usuarios de métodos numéricos para solucionar sistemas lineales y su implementación computacional es: ¿cómo paralelizo mi programa?, esta pregunta no tiene una respuesta simple, depende de muchos factores, por ejemplo: en que lenguaje o paquete está desarrollado, el algoritmo usado para solucionar el problema, a que equipos paralelo tengo acceso, etc.

Algunas respuestas ingenuas son:

- Si uso lenguajes de programación compilables, puedo conseguir un compilador que permita usar directivas de compilación en equipos de memoria compartida sobre programas escritos de forma secuencial, con la esperanza que sean puestos por el compilador como programas paralelos haciendo uso de hilos, OpenMP y optimización del código generado.
- Si uso paquetes como MatLab, Julia o Python, es posible conseguir una versión del paquete que implemente bibliotecas que usen CUDAs, OpenMP o MPI para muchos de los algoritmos más usados en la programación.
- Si mi problema cabe en un sólo equipo, entonces puedo usar dos o más cores para resolver mi problema usando memoria compartida (puedo usar OpenMP, trabajar con hilos o usando paquetes como MatLab o lenguajes como Python, Fortran, C y C++, etc.), pero sólo puedo escalar para usar el máximo número de cores de un equipo (que en la actualidad puede ser del orden de 128 cores, hasta Terabytes de RAM y con almacenamiento de algunas decenas de Terabytes).
- Si mi problema cabe en la GRAM de una GPU, entonces es posible usar una tarjeta gráfica (usando paquetes como MatLab o lenguajes como Python, Fortran, C y C++, etc.), pero sólo puedo escalar a la capacidad de dichas tarjetas gráficas.
- Puedo usar un cluster que usa memoria distribuida-compartida en conjunto con tarjetas gráficas, este tipo de programación requiere una nueva expertes y generalmente implica el uso de paso de mensajes como MPI y rediseño de los algoritmos usados en nuestro programa.

Notemos primero que no todos los algoritmos son paralelizables. En cualquier caso se tienen que ver los pros y contras de la paralelización para cada caso particular. Pero es importante destacar que existen una gran cantidad de bibliotecas y paquetes que ya paralelizan la resolución de sistemas lineales⁴³ y no lineales.

21.5 Nuestra Implementación

Los modelos matemáticos de muchos sistemas en Ciencia e Ingeniería y en particular una gran cantidad de sistemas continuos geofísicos requieren el procesamiento de sistemas algebraicos de gran escala. Es este trabajo se muestra como proceder para transformar un problema de ecuaciones diferenciales parciales en un sistema algebraico virtual de ecuaciones lineales; y así, poder hallar la solución a dicho problema al resolver el sistema lineal asociado al esquema DVS. La solución de este sistema virtual, involucra la solución acoplada de muchos sistemas lineales locales -uno por cada subdominio-, cada uno de estos sistemas lineales puede ser expresado en la forma matricial siguiente $\underline{A}u = f$ donde la matriz \underline{A} es de tamaño $n \times n$ y generalmente bandada, cuyo tamaño de banda es b .

Los métodos de resolución del sistema algebraico de ecuaciones $\underline{A}u = f$ se clasifican en dos grandes grupos (véase [14]): los métodos directos y los métodos iterativos. En los métodos directos la solución u se obtiene en un número fijo de pasos y sólo están sujetos a los errores de redondeo. En los métodos iterativos, se realizan iteraciones para aproximarse a la solución u aprovechando las características propias de la matriz \underline{A} , tratando de usar un menor número de pasos que en un método directo (véase [10], [11], [12] y [14]).

Por lo general, es conveniente usar librerías⁴⁴ para implementar de forma

⁴³Algunas de las bibliotecas más usadas para resolver sistemas lineales usando matrices bandadas y dispersas son PETCS, HYPRE, ATLAS, LAPACK++, LAPACK, EISPACK, LINPACK, BLAS, entre muchas otras alternativas, tanto para implementaciones secuenciales como paralelas y más recientemente para hacer uso de los procesadores CUDA en las GPU de nVidia.

⁴⁴Algunas de las librerías más usadas para resolver sistemas lineales usando matrices bandadas y dispersas son PETCS, HYPRE, ATLAS, LAPACK++, LAPACK, EISPACK, LINPACK, BLAS, entre muchas otras alternativas, tanto para implementaciones secuenciales como paralelas y más recientemente para hacer uso de los procesadores CUDA en las GPU de nVidia.

eficiente a los vectores, matrices -bandadas y dispersas- y resolver los sistemas lineales locales asociados al método DVS, pero se decidió⁴⁵ hacer una jerarquía de clases propia, que implementa los algoritmos necesarios para este trabajo, los cuales corren en cualquier equipo de cómputo que tenga un compilador de C++ y la librería de paso de mensajes MPI; y en caso de querer usar librerías para la manipulación de sistemas lineales, sólo es necesario especializar las clases desarrolladas con las implementaciones particulares de estas; y así, ocultar el uso de dichas librerías sin afectar al resto del código. Siendo sencillo, adaptar el código para usar una o más librerías o versiones de estas, según sea necesario para correr el programa en un equipo de cómputo particular.

Así, para poder operar con los diversos métodos numéricos directos o iterativos que resuelven sistemas lineales reales y virtuales, se implemento una jerarquía de clases, la cual se muestra a continuación:

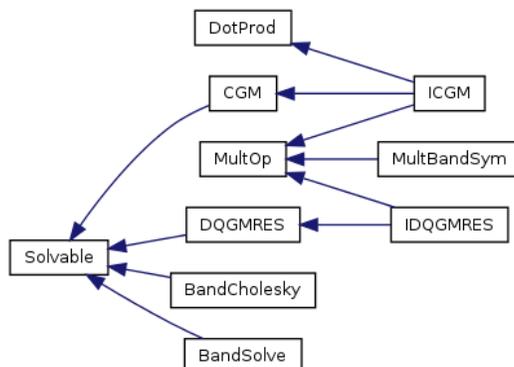


Figura 29: Jerarquía de clases para la resolución de sistemas lineales

Esta jerarquía permite que los métodos DVS le soliciten a la clase abstracta Solvable⁴⁶ que resuelva un determinado sistema lineal sin importar el

⁴⁵La variedad de librerías y las diferentes versiones que existen -muchas de ellas, difieren en la forma de programar entre versiones sucesivas- y pueden usarse es grande, pero cada una de ellas requiere de una implementación específica en el programa y una configuración particular del equipo. Esto restringe al código desarrollado a una plataforma y versión particular de la librería seleccionada.

⁴⁶En general los comportamientos virtuales de las clases abstractas, pueden no ser eficientes si son llamadas una gran cantidad de veces durante la ejecución del programa. Para el caso del esquema DVS, en el cual se usa CGM o GMRES para resolver el sistema

método numérico subyacente -directos o iterativos-, si la matriz es real -existe como tal- o es virtual -esta dispersa en los distintos procesadores del Cluster- y es reutilizable tanto en la implementación secuencial como paralela. La clase Solvable tiene la siguiente estructura:

```
class Solvable
{
    private:
        int iter;
    public:
        virtual int getIter(void)=0;
        virtual void solve(double *x, double *y)=0;
};
```

lineal virtual, este sólo se llama una sola vez; y el proceso de solución del sistema lineal asociado consume la mayoría del tiempo de ejecución, por eso se considera eficiente.

22 Apéndice D: El Cómputo en Paralelo

La computación paralela es el uso de múltiples recursos computacionales para resolver un problema. Se distingue de la computación secuencial en que varias operaciones pueden ocurrir simultáneamente. Los sistemas de cómputo con procesamiento en paralelo surgen de la necesidad de resolver problemas complejos en un tiempo razonable, utilizando las ventajas de memoria, velocidad de los procesadores, formas de interconexión de estos y distribución de la tarea, a los que en su conjunto denominamos arquitectura en paralelo. Entenderemos por una arquitectura en paralelo a un conjunto de procesadores interconectados capaces de cooperar en la solución de un problema.

Así, para resolver un problema en particular, se usa una arquitectura o combinación de múltiples arquitecturas (topologías), ya que cada una ofrece ventajas y desventajas que tienen que ser sopesadas antes de implementar la solución del problema en una arquitectura en particular. También es necesario conocer los problemas a los que se enfrenta un programador que desea que su programa corra en paralelo, como son: el partir eficientemente⁴⁷ un problema en múltiples subtareas y cómo distribuir eficazmente⁴⁸ estas según la arquitectura en particular con que se trabaje. La eficacia difiere de la eficiencia en el sentido de que la eficiencia hace referencia en la mejor utilización de los recursos computacionales (procesadores), en tanto que la eficacia hace referencia en la capacidad para alcanzar un objetivo, aunque en el proceso no se haya hecho el mejor uso de los recursos computacionales.

El cómputo de alto rendimiento (HPC por High-Performance Computing) está formado por un conjunto de computadoras unidas entre sí en forma de Cluster⁴⁹, para aumentar su potencia de trabajo y rendimiento. En 2019 la supercomputadora *SUMMIT* de IBM funcionaban a más de 148 peta-Flops⁵⁰ (cada uno de ellos equivale a la realización de más de 1000 billones

⁴⁷Es un indicador de la utilización efectiva de los diferentes recursos computacionales (principalmente del uso de los procesadores) en relación al algoritmo dado. Se entiende que la eficacia se da cuando se utilizan menos recursos para lograr un mismo objetivo. O al contrario, cuando se logran más objetivos con los mismos o menos recursos.

⁴⁸La podemos definir como el nivel de consecución de metas y objetivos. La eficacia hace referencia a nuestra capacidad de lograr lo que no proponemos.

⁴⁹Existe el Ranking de las 500 supercomputadoras más poderosas del mundo (esta se actualiza cada seis meses en junio y noviembre) y puede ser consultada en:

<https://top500.org>

⁵⁰“FLOPS” operaciones de punto flotante por segundo. Describe una velocidad de procesamiento teórica: para hacer posible esa velocidad es necesario enviar datos a los

de operaciones por segundo de las pruebas de rendimiento del HPC-AI), mientras que en junio de 2020 *FUGAKU* de Japón superaba los 415.5 peta-Flops y en noviembre del mismo año alcanzó los 2.0 exa-Flops, siendo este el primer equipo en alcanzar valores por arriba de un exa-Flops para cualquier precisión sobre cualquier tipo de Hardware, manteniéndose en el primer lugar del top 500 durante el 2021

El clásico uso del paralelismo, es el de diseño de programas eficientes en el ámbito científico. La simulación de problemas científicos es un área de gran importancia, los cuales requieren de una gran capacidad de procesamiento y de espacio de memoria, debido a las complejas operaciones que se deben realizar.

Otro uso clásico es el de las gráficas y vídeos de simulaciones generadas por computadora. La generación de fotogramas y videos requiere de una gran cantidad de cálculos matemáticos. Esto supone una tarea muy compleja para un solo procesador, luego es necesario que haya algún tipo de paralelismo, para distribuir la tarea para que esta sea realizada eficiente y eficazmente.

Tradicionalmente, los programas informáticos se han escrito para el cómputo en secuencial. Para resolver un problema, se construye un algoritmo y se implementa como un flujo en serie de instrucciones. Estas instrucciones se ejecutan en una unidad central de procesamiento en un ordenador. Sólo puede ejecutarse una instrucción a la vez y un tiempo después de que la instrucción ha terminado, se ejecuta la siguiente.

Actualmente, es una práctica común usar directivas de compilación en equipos paralelos sobre programas escritos de forma secuencial, con la esperanza que sean puestos por el compilador como programas paralelos. Esto en la gran mayoría de los casos genera códigos poco eficientes, pese a que corren en equipos paralelos y pueden usar toda la memoria compartida de dichos equipos, el algoritmo ejecutado continúa siendo secuencial en una gran parte del código.

La computación en paralelo, por el contrario, utiliza simultáneamente múltiples elementos de procesamiento para resolver un problema. Esto se logra mediante la división del problema en partes independientes de modo que cada elemento de procesamiento pueda ejecutar su parte del algoritmo de manera simultánea con los otros. Los elementos de procesamiento son di-

procesadores de forma continua. Por lo tanto, el procesamiento de los datos se debe tener en cuenta en el diseño del sistema. La memoria del sistema, junto con las interconexiones que unen los nodos de procesamiento entre sí, impactan en la rapidez con la que los datos llegan a los procesadores.

versos e incluyen recursos tales como una computadora con múltiples procesadores, varios ordenadores en red, Hardware especializado, o cualquier combinación de los anteriores.

Los programas informáticos paralelos son más difíciles de escribir que los secuenciales, porque la concurrencia introduce nuevos tipos de errores de Software, siendo las condiciones de sincronización las más comunes. La comunicación y sincronización entre diferentes subtareas son algunos de los mayores obstáculos para obtener un buen rendimiento del programa paralelo.

¿Cómo Paralelizo mi programa? El problema de todos los usuarios de Ciencias e Ingeniería que hacen uso de métodos numéricos y su implementación computacional es: ¿cómo paralelizo mi programa?, esta pregunta no tiene una respuesta simple, depende de muchos factores, por ejemplo: en que lenguaje o paquete está desarrollado el programa, el algoritmo usado para solucionar el problema, a que equipos paralelo tengo acceso y un largo etc. Algunas respuestas ingenuas podrían ser:

- Si uso lenguajes de programación compilables, puedo conseguir un compilador que permita usar directivas de compilación en equipos de memoria compartida sobre programas escritos de forma secuencial, con la esperanza que sean puestos por el compilador como programas paralelos haciendo uso de hilos, OpenMP y optimización del código generado.
- Si uso paquetes como MatLab, Julia o Python, es posible conseguir una versión del paquete que implemente bibliotecas que usen CUDA, OpenMP o MPI para muchos de los algoritmos más usados en la programación.
- Si mi problema cabe en un sólo equipo, entonces puedo usar dos o más cores para resolver mi problema usando memoria compartida (puedo usar OpenMP, trabajar con hilos o usando paquetes como MatLab o lenguajes como Python, Julia, Fortran, C y C++, etc.), pero sólo puedo escalar para usar el máximo número de cores de un equipo (que en la actualidad puede ser del orden de 128 cores, hasta Terabytes de RAM y con almacenamiento de algunas decenas de Terabytes).
- Si mi problema cabe en la GRAM de una GPU, entonces el posible usar una tarjeta gráfica (usando paquetes como MatLab o lenguajes como

Python, Julia, Fortran, C y C++, etc.), pero sólo puedo escalar a la capacidad de dichas tarjetas gráficas.

- Puedo usar un cluster que usa memoria distribuida-compartida en conjunto con tarjetas gráficas, este tipo de programación requiere una nueva expertes y generalmente implica el uso de paso de mensajes como MPI y rediseño de los algoritmos usados en nuestro programa.

Notemos primero, que no todos los algoritmos son paralelizables. En cualquier caso se tienen que ver los pros y contras de la paralelización para cada caso particular. Pero es importante destacar que existen una gran cantidad de bibliotecas y paquetes que ya paralelizan ciertos algoritmos que son ampliamente usados como la solución de sistemas lineales⁵¹ y no lineales. Además, el tiempo de programación necesario para desarrollar una aplicación paralela eficiente y eficaz para la gran mayoría de los programadores puede ser de semanas o meses en el mejor de los casos. Por ello, es necesario hacer un balance entre las diferentes opciones de paralelización para no invertir un tiempo precioso que puede no justificar dicha inversión económica, de recursos computacionales y sobre todo de tiempo.

Antes de iniciar con los tópicos del cómputo en paralelo, es necesario conocer cómo está constituida nuestra herramienta de trabajo: la computadora. Los conocimientos básicos de arquitectura de las computadoras nos permite identificar los componentes de Hardware que limitarán nuestros esfuerzos por realizar una paralelización óptima de nuestro programa y en muchos casos descubriremos de la peor manera que los algoritmos usados en nuestro programa no son los más adecuados para paralelizar; y descubriremos con horror que en algunos casos los tiempos de ejecución no mejoran sin importar cuantos equipos adicionales usemos, en el peor de los casos, el tiempo de ejecución se incrementa al aumentar equipos en vez de disminuir.

⁵¹Algunas de las bibliotecas más usadas para resolver sistemas lineales usando matrices bandadas y dispersas son PETCs, HYPRE, ATLAS, LAPACK++, LAPACK, EISPACK, LINPACK, BLAS, entre muchas otras alternativas, tanto para implementaciones secuenciales como paralelas y más recientemente para hacer uso de los procesadores CUDA en las GPU de nVidia.

22.1 Computadoras Actuales

La computadora (también conocida como ordenador) actual es una máquina digital programable que ejecuta una serie de comandos para procesar los datos de entrada, obteniendo convenientemente información que posteriormente se envía a las unidades de salida. Una computadora está formada físicamente por numerosos circuitos integrados y varios componentes de apoyo, extensión y accesorios, que en conjunto pueden ejecutar tareas diversas con suma rapidez y bajo el control de un programa (Software).

La constituyen dos partes esenciales, el Hardware, que es su estructura física (circuitos electrónicos, cables, gabinete, teclado, etc.), y el Software, que es su parte intangible (programas, datos, información, documentación, etc.).

Con respecto al Hardware⁵², se encuentra compuesto por una serie de dispositivos, clasificados según la función que estos desempeñen. Dicha clasificación se compone de:

- Los dispositivos de entrada son todos aquellos que permiten la entrada de datos a una computadora. Estos dispositivos (periféricos), son los que permiten al usuario interactuar con la computadora. Ejemplos: teclado, Mouse (ratón), micrófono, Webcam, Scanner, etc.
- Los dispositivos de salida, son todos aquellos que permiten mostrar la información procesada por la computadora. Ejemplos: monitor, impresora, auriculares, altavoces, etc.
- Los dispositivos de comunicación son aquellos que permiten la comunicación entre dos o más computadoras. Ejemplos: Modem, Router, placa de red, Bluetooth, etc.

⁵²En Debian GNU/Linux podemos instalar la aplicación *lshw* para conocer los distintos componentes de la computadora, mediante:

```
# apt install lshw
```

Así, para ver de forma resumida los dispositivos que componen la computadora usamos:

```
# lshw -short
```

Si necesitamos más detalle usamos:

```
# lshw
```

- Los dispositivos de almacenamiento, son todos aquellos que permiten almacenar datos en el ordenador. Ejemplos: disco duro, Pendrive, Diskette, CD y DVD, etc.
- Los dispositivos de cómputo, son aquellos encargados de realizar las operaciones de control necesarias, sobre el resto de los dispositivos la computadora.

La unidad central de procesamiento (Central Processing Unit CPU) se comunica a través de un conjunto de circuitos o conexiones llamada Bus de datos o canal de datos. El bus conecta la CPU a los dispositivos de almacenamiento, los dispositivos de entrada y los de salida.

Desde el punto de vista funcional es una máquina que posee, al menos, una unidad central de procesamiento, unidad de memoria (Random Access Memory RAM , Read Only Memory ROM y Caché) y dispositivos de entrada/salida (periféricos). Los periféricos de entrada permiten el ingreso de datos, la CPU se encarga de su procesamiento (operaciones aritmético-lógicas) y los dispositivos de salida los comunican a los medios externos. Es así, que la computadora recibe datos, los procesa y emite la información resultante, la que luego puede ser interpretada, almacenada, transmitida a otra máquina o dispositivo o sencillamente impresa; todo ello a criterio de un operador o usuario y bajo el control de un programa de computación.

CPU la Unidad Central de Proceso (Central Processing Unit CPU) es aquella parte del procesador que se encarga de ejecutar las diversas acciones que ordenemos al dispositivo que debe llevar a cabo. La CPU es el componente básico dentro de todo dispositivo inteligente, ya que prácticamente cualquier proceso que se ordene al sistema pasa por él. Con el paso del tiempo, además, su eficiencia y calidad ha alcanzado grandes cotas, aunque tecnologías al alza como las NPU han supuesto un mayor salto cualitativo que el de aumentar la potencia bruta de la CPU.

GPU, para el procesamiento gráfico la Unidad de Procesamiento Gráfico (Graphics Processing Unit GPU) es el apartado que se dedica a las acciones de mayor peso: las de componente gráfico. De este modo, acciones como la ejecución de videojuegos, o la edición y renderizado de vídeos, se llevan a cabo a través de la GPU del sistema. La calidad del teléfono, ordenador u otro dispositivo inteligente, suele ir supeditada a menudo a la calidad de

su GPU, que dependerá de la banda de precio del aparato en cuestión. Eso sí, si los otros componentes no tienen la misma calidad, se puede producir el temido cuello de botella.

En los smartphones, la GPU ya va integrada en los procesadores, pero en los PC's, como AMD y NVIDIA se muestran como marcas especializadas en las GPUs. Cuentan con todo tipo de gamas y también poseen un amplio abanico de precios, un valor siempre directamente proporcional a la calidad y capacidad de sus gráficas, y a su vigencia en el mercado.

NPU, redes neuronales en tu dispositivo la Unidad de Procesamiento Neuronal (Neural Processing Unit NPU), a diferencia de la GPU, que cuenta con un funcionamiento paralelo al de la CPU, puede encargarse de funciones similares a las de la CPU, pero lo hace de un modo mucho más eficiente. Impulsada por Inteligencia Artificial, una NPU es capaz de priorizar procesos para ejecutarlos de un modo exponencialmente más veloz y con un consumo mucho menor. En móviles, se usa especialmente para mejorar el procesado de fotografías, aunque participa en muchos otros procesos.

Además, esta arquitectura todavía tiene años de progreso por delante, a diferencia de la CPU y la GPU, cuyas mejoras ya son de carácter más leve y basadas en aumentar la potencia bruta. La tecnología NPU⁵³, en cambio, lleva menos tiempo entre nosotros y todavía tiene mucho margen de mejora para ofrecer un rendimiento cada vez más poderoso.

Desde la llegada de los procesadores de más de un núcleo a PC, como consecuencia de la imposibilidad de hacerlos escalar en potencia solo por velocidad de reloj, la forma de entender los diferentes chips cambió. Han pasado ya dos décadas de dicho cambio y ante la inminente salida de los Chips disgregados o por Chiplets al mercado de masas, no está de más recordar la organización más común en el mundo del Hardware en todo este tiempo.

¿Qué es un SoC? las siglas SoC significan System on a Chip y hace referencia a todo chip que tiene la mayoría de componentes integrados en una misma pieza de silicio sin llegar a ser un microcontrolador. Se trata de la pieza de Hardware más usada por el hecho de que a día de hoy todo procesador para

⁵³Para muestra, la compañía Cerebras con su procesador WSE-3 aglutina 4 billones de transistores, tiene una superficie de 46,225 mm², integra nada menos que 900,000 núcleos optimizados para IA y es capaz de entrenar hasta 24,000 millones de parámetros, lo que también equivaldría a un rendimiento máximo de IA de 125 petaflops.

PC, teléfono móvil, consola de videojuegos, televisor o incluso servidores, es un SoC y pese a las diferencias entre ellos, todos tienen una organización común.

En realidad, todas las CPU actuales son SoC, ya que se trata de "varias CPU" diseñadas para funcionar alrededor de un elemento de intercomunicación central. Este se encarga de interconectar los diferentes elementos entre sí y de darles acceso a interfaces externas.

Por ejemplo, con la memoria RAM (u otros), a la que está asociado el controlador de memoria que comparten todos sus elementos o a los periféricos, y a los cuales se puede acceder directamente con una serie de interfaces específicas, o a través de un Chipset externo encargado de gestionar los diferentes periféricos y componentes.

En PC, debido a que la comunicación con los periféricos se hace a través del uso de direcciones de memoria de la RAM principal, los componentes relacionados con esta se encuentran subordinados al controlador de memoria. Por lo que son una pieza más conectada a la parte central.

¿Qué es una APU y en qué se diferencia de un SoC? las siglas APU significan Accelerated Processor Unit y fue usada por AMD cuando sus CPU e iGPU (o GPU integrada, la veremos a continuación) no traían consigo ningún sistema de gestión de E/S (entrada y salida). Sin embargo, la cosa empezó a cambiar ya con la arquitectura "Carrizo" que fue el nombre clave de los últimos SoC antes del lanzamiento de los AMD Ryzen, donde se incorporaron varias interfaces de periféricos directamente en la CPU.

A día de hoy todo es un SoC, lo que ocurre es que, en sobremesa, las torres tienen tanta conectividad y capacidad de expansión que se suele emplear un Chipset, y lo mismo ocurre en estaciones de trabajo y servidores, pero no en el resto. Y es que si hablamos de un Chip para un PC portátil, una consola o un móvil, entonces al no existir tantas interfaces para periféricos y otros componentes, entonces éstos se pueden integrar en un mismo chip.

En realidad, el término APU es más bien comercial de AMD y al día de hoy se utiliza como sinónimo de SoC, pero se puede resumir en que una APU carece de cualquier gestión de periféricos y requiere de un Chip externo para ello. Mientras que un SoC tiene las especificaciones mínimas en ese aspecto, pese a ser también ampliables. Para simplificar la idea, un SoC es una APU más completa.

¿iGPU qué es y cuáles son sus características concretas frente a una dGPU? las siglas iGPU corresponden a integrated GPU, y hace referencia a todo componente de este tipo que se encuentre integrado en una APU o un SoC. Por lo que se trata de procesadores gráficos de potencia limitada que se pueden ver lastrados en velocidad de reloj por el problema del ahogamiento termal (Thermal Throttling) que se produce cuando muchas partes comparten el mismo espacio físico.

Si bien, es posible llegar a ciertos niveles de rendimiento que son aceptables de cara a reproducir las escenas en 3D a tiempo real en los videojuegos, y otras tareas de carácter profesional donde se usa una tarjeta gráfica, estas se ven cuanto menos limitadas:

El hecho de tener que compartir espacio en el mismo Chip con los diferentes núcleos de la CPU produce que esta no pueda alcanzar la misma velocidad de reloj que se alcanzaría siendo un Chip aparte e independiente.

En PC, debido a que como la memoria RAM usa DDR o LPDDR, el ancho de banda es pequeño y hemos de partir del hecho de que el rendimiento de todo Chip gráfico, incluido una iGPU, depende del ancho de banda que se le puede otorgar con dicha memoria RAM.

Los SoC con iGPU actuales tienen un tamaño fijo, definido este por la cantidad de pines soportados por la interfaz con la placa base. Esto limita el tamaño, no solo del Chip, sino también de la gráfica integrada en el mismo.

iGPU en consolas de videojuegos al contrario de lo que ocurre con los PC, las consolas de videojuegos no tienen que seguir una serie de normas respecto a sus componentes. Para empezar, no ven el tamaño de sus chips limitados por un estándar de placa base, dado que son productos únicos y exclusivos. Esto les permite tener el tamaño que quieran, incluso más que una CPU convencional para PC, lo que les permite tener una iGPU en su SoC mucho más avanzada.

El otro punto es la memoria utilizada, ya que la de las tarjetas gráficas también se usan como memoria principal. Esta da el ancho de banda necesario para que la iGPU alcance cierto nivel de rendimiento, pero su latencia es mucho más alta que la RAM convencional de PC que está más optimizada en ese aspecto, por lo que el rendimiento de su CPU suele ser más bajo que su equivalente para ordenador.

¿Qué es una dGPU en un PC o portátil? las dGPU, o GPU dedicadas, no son otra cosa que las GPU de toda la vida, pero la particularidad es que también son SoC, ya que tiene varios núcleos, especializados en tareas gráficas, alrededor de una interfaz central y compartiendo todos ellos un mismo acceso a memoria.

Sin embargo, carecen de núcleos de CPU en su interior, de ahí a que no se les llame APU o SoC, por el hecho de que de existir estos elementos pasarían a ser una iGPU. Por lo tanto, su principal particularidad de las dGPU es que tienen su propia memoria RAM (SDRAM para ser concretos) la cual suele rodear estos Chips, ya sea en forma de tarjeta gráfica o soldados en la placa de los ordenadores portátiles. A esta la llamamos VRAM y es de uso exclusivo de la dGPU o GPU.

Los equipos de cómputo los podemos clasificar⁵⁴ por:

- Equipos móviles: estos equipos buscan un equilibrio entre su capacidad de cómputo versus el rendimiento energético de sus baterías -para operar el mayor tiempo posible sin recargarse- y su peso, entre estos equipos destacan las Laptops, Notebook, Netbook, Ultrabook, tabletas, teléfonos inteligentes, etc.
- Equipos de escritorio: estos equipos al estar permanentemente conectados a la corriente eléctrica pueden tener un mayor número de componentes y disponen de una mejor capacidad disipación de calor por lo que pueden contener una mayor cantidad componentes, como discos, RAM o tarjetas de video y el tamaño, peso o consumo energético no es un inconveniente.
- Servidores: son equipos que suelen atender a múltiples usuarios simultáneamente y disponen de gran cantidad de Cores, RAM, disco y son interconectados por red de alta velocidad con otros servidores para atender las crecientes necesidades de los centros de datos los cuales deben estar permanentemente en operación. Generalmente los equipos son montados en Racks con otras decenas de equipos, por lo que su arquitectura se ve limitada a una moderada generación de calor por

⁵⁴El ordenador del Apollo 11, el Block II, funcionaba a una velocidad de 2 MHz y tenía 2 KB de memoria RAM y 32 KB de memoria ROM. Para ponerlo en contexto, el chip de un cargador USB-C moderno es 563 veces más potente que la computadora que se usó en el Apollo 11, al menos en términos de potencia bruta.

parte de sus componentes ya que su sistema de ventilación es por aire para todo el Rack.

- Estaciones de trabajo: son equipos individuales diseñados para atender cargas computacionales intensas, por lo que requieren Hardware más complejo y potente como puede ser múltiples tarjetas de video (con decenas de miles de Cores gráficos) , discos (con cientos de Terabytes), gran cantidad de RAM (pueden llegar a superar el Terabyte) y sistema de refrigeración por aire o líquido, etc.
- Cómputo intensivo: son equipos interconectados por red de alta velocidad con procesadores y tarjetas gráficas dedicadas para el cálculo numérico que soportan cargas intensas por largos periodos de tiempo, los más comunes son los que forman parte de los Cluster que llegan a tener millones de cores.

FLOPS Una medida relativamente objetiva para analizar el rendimiento de un dispositivo suele ser medir sus operaciones de punto flotante por segundo o más conocidas como FLOPS. Hay que tener en cuenta que la medición de FLOPS es muy compleja porque las diferentes operaciones en punto flotante llevan diferentes cantidades de tiempo para ejecutarse. Y no todo el mundo utiliza las mismas operaciones para establecer los cálculos.

Por ejemplo, una división simple como $1/5$, toma significativamente menos tiempo que el cálculo del logaritmo de 5. Por eso, se estableció el algoritmo de Linpack como un estándar representativo con el que poder medir todos los sistemas bajo el mismo baremo de FLOPS.

Es importante señalar que el algoritmo de Linpack utiliza el formato en punto flotante de doble precisión (64-bit). Sin embargo, como veremos la mayoría de los valores que dan los fabricantes son con precisión simple (32-bit). Además, los valores que dan los fabricantes suelen ser teóricos y en la práctica suelen ser inferiores debido a otros factores limitantes como la frecuencia de reloj o la velocidad de las memorias ROM y RAM.

Por tanto, aunque todos hemos acabado midiendo el rendimiento en FLOPS⁵⁵, no es una medida absoluta de la potencia del CPU ni de una

⁵⁵El Cray-1 fue puesto marcha en 1975 y utilizaba una CPU a 80 MHz y llevaba integrada una unidad SIMD de 64 bits de precisión de punto flotante, lo cual fue un salto de gigante que permitió un salto de los 3 MFLOPS de potencia del CDC 6600 a los 160 MFLOPS en el Cray-1.

GPU. Por ejemplo para algunos dispositivos tenemos:

Móviles El SoC Snapdragon 821 que monta una GPU Adreno 530 tiene una potencia de 519.2 gigaFLOPS (0.52 TFLOPS), y los Chips Apple A9X del iPad Pro alcanzan los 345.6 gigaFLOPS (0.35 TFLOPS), todos ellos medidos con precisión simple de 32-bits.

CPU

- Intel Xeon W-3245: 1.4 TFLOPS
- Intel Core i9-9900X: 1.2 TFLOPS
- AMD Ryzen 9 3950X: 1.1 TFLOPS

Los procesadores de gama media-alta rondan el medio TFLOPS:

- AMD Ryzen 7 3700X: 546.0 GFLOPS - 0.55 TFLOPS
- Intel Core i9-9900: 499.0 GFLOPS - 0.50 TFLOPS
- AMD Ryzen 5 3600X: 461.0 GFLOPS - 0.46 TFLOPS

Tarjetas Gráficas Ojo: La tabla está ordenada por los valores en precisión simple (32-bit) primer columna

GPU	FP32 TFLOPS	FP64 TFLOPS
TITAN V	13.8	6.9
Radeon RX Vega 64	12.7	0.8
GeForce GTX 1080 Ti	11.3	0.4
GeForce GTX 1080	8.9	0.3
Radeon R9 Fury X	8.6	0.5
Radeon HD 7990	7.8	1.9
GeForce GTX 1070	6.5	0.2
Radeon RX 480	5.8	0.4
GeForce GTX 690	5.6	0.2
Radeon R9 290X	5.6	0.7
GeForce GTX 780 Ti	5.3	0.2
Radeon HD 6990	5.1	1.3
GeForce GTX 980	4.9	0.15

Radeon RX 470	4.9	0.3
Radeon R9 290	4.8	0.6
GeForce GTX Titan	4.7	1.5
GeForce GTX 1060	4.4	0.14
Radeon HD 7970 GHz	4.3	1.1
GeForce GTX 780	4.1	0.17
Radeon R9 280X	4.0	1.0
Radeon R9 280	3.3	0.83
GeForce GTX 680	3.1	0.13
Radeon HD 7950	2.9	0.71

Como podemos ver, las tarjetas gráficas de Nvidia, normalmente, tienen una potencia muy alta en precisión simple, pero muy mala en precisión doble. La precisión simple es la que se usa en los juegos, pero la precisión doble es la que se utiliza en los cálculos complejos científicos y en el minado de muchas criptomonedas.

Consolas Todas ellas son en valores de precisión simple (32-bit)

- PlayStation 4: 1.3 TFLOPS
- Xbox One: 1.8 TFLOPS
- PlayStation 4 Pro: 4.2 TFLOPS
- Nintendo Switch: entre 0.4 y 0.5 TFLOPS
- PlayStation 5 promete una GPU con 10.28 TFLOPS
- La Xbox Series X promete una GPU de 12 TFLOPS

SuperCómputo Se dio a conocer en noviembre del 2023 la publicación de la 62ª edición del ranking de las 500 computadoras de mayor rendimiento del mundo:

- El Frontier, ubicado en el Laboratorio Nacional Oak Ridge del Departamento de Energía de Estados Unidos, sigue posicionado en el primer lugar (manteniendo desde mediados del año pasado). El clúster tiene 8.7 millones de núcleos de procesador y proporciona un rendimiento de 1.194 exaflops, el doble que el clúster, que ocupa el segundo lugar (con un menor consumo de energía).

- El nuevo clúster Aurora, desplegado en el Laboratorio Nacional Argonne del Departamento de Energía de Estados Unidos. El clúster tiene casi 4.8 millones de núcleos de procesador y proporciona un rendimiento de 585 petaflops, que es 143 petaflops más que el clúster que anteriormente ocupaba el segundo lugar.
- El clúster Eagle, lanzado este año por Microsoft para la nube Azure. El clúster contiene 1.12 millones de núcleos de procesador (CPU Xeon Platinum 8480C 48C 2GHz) y demuestra un rendimiento máximo de 561 petaflops. El Software del clúster está basado en Ubuntu 22.04.

El umbral mínimo de rendimiento para ingresar al Top 500 durante 6 meses fue de 2.02 petaflops (hace seis meses, 1.87 petaflops). Hace cinco años, sólo 272 clusters mostraban un rendimiento de más de un petaflop, hace seis años 138 y hace siete años - 94). Para el Top 100, el umbral de entrada aumentó de 6.3 a 7.89 petaflops, y para el Top 10, de 61.44 a 94.64 petaflops.

El rendimiento total de todos los sistemas en la clasificación durante 6 meses aumentó de 5.2 a 7 exaflops (hace cuatro años era 1.650 exaflops y hace seis años, 749 petaflops). El sistema que cierra el ranking actual ocupaba en la última edición el puesto 454.

Para poner en contexto los avances en este campo, en el año 2004 IBM era dueña y señora del mundo de la supercomputación, su espectacular BlueGene/L dominaba la lista TOP.500. Aquel monstruo contaba con 32,768 procesadores PowerPC 440 a 700 MHz y 16 TB de memoria. 20 años después una sola NVIDIA GeForce RTX 4090 con 24 GB de memoria GDDR6X es más potente que esa supercomputadora -lo es al menos en rendimiento bruto-, BlueGene/L contaba en ese momento con un rendimiento de 70.72 TFLOPS, pero la propia NVIDIA dejaba claro en el lanzamiento de sus RTX 4090 que estas tarjetas gráficas contaban con una potencia de 83 TFLOPS.

Es más, cuatro RTX 4090 con soporte FP8 logran también rivalizar con la supercomputadora más potente de 2009. Y eso sin apretarle las tuercas a las RTX 4090: en noviembre de 2022 es precisamente lo que hicieron en Wccftech y lograron que la RTX 4090 se convirtiera en la primera tarjeta gráfica del mundo en alcanzar los 100 TFLOPS.

Esa comparación es como decimos real en esa potencia de cálculo en bruto, pero también es cierto que en esa y otras supercomputadoras se tenían mecanismos especiales de comunicación entre procesadores o de transferencia

de datos, algo para lo que las GPUs actuales, aún siendo sobresalientes, no están tan optimizadas.

¿Cómo Trabaja una Computadora? Todas las computadoras sean de uno o más procesadores ejecutan los programas realizando los siguientes pasos:

1. Se lee una instrucción
2. Se decodifica la instrucción
3. Se encuentra cualquier dato asociado que sea necesario para procesar la instrucción
4. Se procesa la instrucción
5. Se escriben los resultados

Esta serie de pasos, simple en apariencia, se complican debido a la jerarquía de memoria RAM, en la que se incluye la memoria Caché, la memoria principal y el almacenamiento no volátil como pueden ser los discos duros o de estado sólido (donde se almacenan las instrucciones y los datos del programa), que son más lentos que el procesador en sí mismo. Con mucha frecuencia, el paso (3) origina un retardo muy largo (en términos de ciclos del procesador) mientras los datos llegan en el bus de la computadora.

Durante muchos años, una de las metas principales del diseño microinformático ha sido la de ejecutar el mayor número posible de instrucciones en paralelo, aumentando así la velocidad efectiva de ejecución de un programa. No obstante, estas técnicas han podido implementarse en Chips semiconductores cada vez más pequeños a medida que la fabricación de estos fue progresando y avanzando, lo que ha abaratado notablemente su costo.

El procesador es el cerebro de un ordenador. No hay que olvidar otros componentes como la memoria, el almacenamiento o la tarjeta gráfica dedicada, desde luego, pero el procesador está un escalafón por encima en la jerarquía.

Piensa que, si cambiamos el procesador en dos equipos con la misma memoria, almacenamiento o tarjeta gráfica, el comportamiento puede variar notablemente. Sin embargo, para un mismo procesador, los cambios en el resto de componentes no impactan de forma tan directa en la experiencia de uso de un equipo.

¿Qué es una CPU? Antes de nada, vamos a definir exactamente lo que es una CPU o un procesador. Como bien indican sus siglas en inglés (Central Processing Unit) es la unidad de procesamiento -puede ser Intel, AMD, ARM, etc- encargada de interpretar las instrucciones de un Hardware haciendo uso de distintas operaciones aritméticas y matemáticas. Características principales de un procesador:

- Frecuencia de reloj. Este primer término hace referencia a la velocidad de reloj que hay dentro del propio procesador. Es un valor que se mide en Mhz o Ghz y es básicamente la cantidad de potencia que alberga la CPU. La mayoría de ellas cuentan con una frecuencia base -para tareas básicas- y otra turbo que se utiliza para procesos más exigentes -con un aumento en el consumo de energía y por ende un aumento en la temperatura del procesador, requiriendo sistemas de disipación de calor eficientes-.
- Consumo energético. Es normal que nos encontremos con CPU 's donde su consumo energético varía notablemente. Es un valor que se muestra en vatios (W) y como es obvio, aquellos procesadores de gama superior, serán más propensos a consumir más energía. Ante esto, es importante contar con un eficiente sistema de enfriamiento además de contar con una fuente de alimentación acorde a la potencia requerida por el procesador, la tarjeta gráfica y sus respectivos sistemas de enfriamiento.
- Número de núcleos. Con el avance de la tecnología, ya es posible encontrar tanto procesadores de Intel como de AMD que cuentan ya con decenas de núcleos. Estos cores son los encargados de llevar a cabo multitud de tareas de manera simultánea.
- Número de hilos. Si un procesador tiene Hyperthreading en el caso de Intel o SMT (Simultaneous Multi-Threading) en el caso de AMD, significa que cada uno de los núcleos es capaz de realizar dos tareas de manera simultánea, lo que se conoce como hilos de proceso. Por lo tanto, un procesador de cuatro núcleos físicos con Hyperthreading tendría ocho hilos de proceso, y sería capaz de ejecutar ocho órdenes al mismo tiempo -los hilos no tienen las mismas capacidades de un core real y en muchos casos su uso merma el rendimiento del CPU, pero los sistemas operativos los reconocen como si fueran cores reales-⁵⁶.

⁵⁶El AMD EPYC 9845 de 160 núcleos y 320 hilos a una frecuencia de 2,00 GHz basada

- Memoria Caché. A la hora de "recordar" cualquier tarea, el propio ordenador hace uso de la memoria RAM. Sin embargo no es eficiente este proceso y por tanto es necesario que utilice la memoria Caché de la CPU para paliar esta deficiencia. El Caché se caracteriza porque se llega a ella de forma más rápida y puede ser tipo L1, L2 y L3.
- Zócalo. Es el tipo de conector con pines o Socket al que se conecta la placa base. Por ejemplo, las últimas de Intel suelen tener el Socket LGA 1200, mientras que las de AMD con Ryzen son AM4.
- Red. Si bien la red es un recurso indispensable en un equipo de cómputo, en el caso de equipos paralelos la velocidad de la red es el mayor cuello de botella en cuanto a rendimiento, por ello es necesario usar redes de alto desempeño como las de InfiniBand con un alto costo económico pero de alto desempeño que pueden llegar al orden de cientos de Gigabytes por segundo.

Nuevos Procesadores la creciente demanda de dispositivos de cómputo ha generado una gran variedad de procesadores, los podemos clasificar como:

- Procesador compuesto por múltiples núcleos de alta eficiencia -con un consumo energético reducido- que sacrifican potencia de procesamiento en aras de extender la carga útil de las baterías de los dispositivos móviles.
- Procesador compuesto por múltiples núcleos de alto rendimiento que pueden estar al tope de su capacidad sin generar excesivo calor y son especialmente usados en servidores y en cómputo intensivo.
- Procesadores compuestos por múltiples núcleos de alto rendimiento que pueden ajustar su velocidad de reloj de manera dinámica para tratar cargas de trabajo pesadas por un cierto tiempo -pues generan gran cantidad de calor-, por lo que requieren un sistema eficiente de enfriamiento, son ideales para estaciones de trabajo.
- Procesadores compuestos por múltiples núcleos híbridos que en lugar de tener un único tipo de núcleo multipropósito, estos Chips cuentan con dos grupos de núcleos. El primero de ellos, compuesto por

en Zen 5c, este se acompaña de 640 MB de caché L3.

múltiples núcleos de alta eficiencia, se encarga de procesar las tareas más livianas o en segundo plano que deba realizar un procesador, todo ello, con un consumo energético menor. El otro grupo, compuesto por múltiples núcleos de alto rendimiento, sigue una dinámica opuesta, su consumo energético es superior, pero únicamente entran en funcionamiento cuando la tarea en cuestión requiere un extra de procesamiento.

Para gestionar esta división de núcleos híbridos, se ha integrado un "Thread Director", un elemento que se encarga de determinar qué núcleo procesa cada tarea. Las compañías, además, ha modificado cómo funciona la caché de sus procesadores:

- Cada núcleo de rendimiento tiene su propia caché L2.
- Cada cluster de núcleos de eficiencia tiene una "piscina" de memoria L2 común, de la que beben todos los núcleos que sean partícipes.
- Tanto los núcleos de rendimiento como los de eficiencia tienen acceso a una "piscina" de memoria L3 común para todos ellos.

Otros de los cambios que impactarán en el desempeño de las CPUs es el aumento de velocidad y una mayor cantidad de memoria Caché, compatibilidad con memorias DDR6 y con la interfaz PCIe 6.0.

PCIe PCI Express (Peripheral Component Interconnect Express), abreviado como PCIe, es una tecnología de conexión de Hardware utilizada para la comunicación de alta velocidad entre diferentes componentes de un equipo informático. Este estándar se ha convertido en la interfaz más habitual para la conexión de tarjetas de expansión, como tarjetas gráficas que sirven para correr juegos, tarjetas de sonido, tarjetas de red y dispositivos de almacenamiento de alta velocidad.

Una de las ventajas destacadas de PCI Express es su arquitectura de canales independientes, que permiten la transferencia simultánea de datos en ambos sentidos. Cada carril tiene una tasa de transferencia específica, medida en gigabits por segundo (Gbps), y la capacidad de un slot PCIe se expresa como el número de carriles que tiene. Esto se traduce en un ancho de banda total mayor, lo que facilita la conexión de dispositivos que

requieren altas tasas de transferencia, como las tarjetas gráficas modernas o los dispositivos de almacenamiento de última generación.

Los diferentes tipos de ranuras de PCI Express según su tamaño PCIe X1 carriles 1, pines 18, PCIe x4 carriles 4, pines 32, PCIe x8 carriles 8, pines 49, PCIe x16 Carriles 16, pines 82.

Adicionalmente, también es interesante fijarse en las diferentes versiones que se han ido lanzando desde que PCIe se lanzó al mercado:

- PCIe 1.0 ancho banda 8 GB/s, velocidad de transferencia 2.5 GT/s
- PCIe 2.0 ancho banda 16 GB/s, velocidad de transferencia 5 GT/s
- PCIe 3.0 ancho banda 32 GB/s, velocidad de transferencia 8 GT/s
- PCIe 4.0 ancho banda 64 GB/s, velocidad de transferencia 16 GT/s
- PCIe 5.0 ancho banda 128 GB/s, velocidad de transferencia 32 GT/s
- PCIe 6.0 ancho banda 256 GB/s, velocidad de transferencia 64 GT/s

Características Arquitectónicas los procesadores Intel x86 admiten un formato de precisión extendido de 80 bits con un significado de 64 bits, que es compatible con el especificado en el estándar IEEE. Cuando un compilador usa este formato con registros de 80 bits para acumular sumas y productos internos, está trabajando efectivamente con un redondeo unitario de 2^{-64} en vez de 2^{-53} para precisión doble, dando límites de error más pequeños en un factor de hasta $2^{11} = 2048$.

Algunos procesadores Intel y AMD tienen una operación fusionada de multiplicación y suma (FMA), que calcula una multiplicación y una suma combinadas $x + yz$ con un error de redondeo en lugar de dos. Esto da como resultado una reducción en los límites de error por un factor 2.

Las operaciones FMA de bloques de precisión mixta $D = C + AB$, con matrices A, B, C y D de tamaño fijo, están disponibles en las unidades de procesamiento tensorial de Google, las GPU NVIDIA y en la arquitectura ARMv8-A. Para entradas de precisión media, estos dispositivos pueden producir resultados de calidad de precisión simple, lo que puede proporcionar un aumento significativo en la precisión cuando los bloques FMA se encadenan para formar un producto matricial de dimensión arbitraria.

Meltdown y Spectre El tres de enero del 2018 se dio a conocer al público, que 6 meses antes se habían detectado dos distintos fallos en los procesadores de los equipos de cómputo, comunicaciones y redes de internet que usamos. Esto para dar tiempo a los desarrolladores de procesadores y de sistemas operativos de implementar estrategias para mitigar el problema. Estos son problemas de diseño de los procesadores de Intel, AMD, IBM POWER y ARM, esto significa que procesos con privilegios bajos -aquellos que lanzan las aplicaciones de usuarios convencionales- podían acceder a la memoria del Kernel del sistema operativo⁵⁷.

Un ataque que explota dicho problema permitiría a un Software malicioso espiar lo que están haciendo otros procesos y también espiar los datos que están en esa memoria en el equipo de cómputo (o dispositivo móvil) atacado. En máquinas y servidores multiusuario, un proceso en una máquina virtual podría indagar en los datos de los procesos de otros procesos en ese servidor compartido.

Ese primer problema, es en realidad solo parte del desastre. Los datos actuales provienen especialmente de un grupo de investigadores de seguridad formados por expertos del llamado Project Zero⁵⁸ de Google. Ellos han publicado los detalles de dos ataques (no son los únicos⁵⁹) basados en estos fallos de diseño. Los nombres de esos ataques son Meltdown y Spectre. Y en un sitio Web dedicado a describir estas vulnerabilidades destacan que "aunque los programas normalmente no tienen permiso para leer datos de otros programas, un programa malicioso podría explotar Meltdown, Spectre

⁵⁷En GNU/Linux, el Kernel (si usamos una versión actualizada) nos indica las fallas del procesador a las que es vulnerable, usando:

```
$ cat /proc/cpuinfo
$ lscpu
```

⁵⁸<https://googleprojectzero.blogspot.com/>

⁵⁹Entre las distintas vulnerabilidades detectadas y sus variantes resaltan: Meltdown (AC, DE, P, SM, SS, UD, GP, NM, RW, XD, BR, PK, BND), Spectre (PHT, BTB, RSB, STL, SSB, RSRE), PortSmash, Foreshadow, Spoiler, ZombieLoad (1 y 2), Kaiser, RIDL, Plundervolt, LVI, Take a Way, Collide+Probe, Load+Reload, LVI-LFB, MSD, CSME, RYZENFALL (1, 2, 3, 4), FALLOUT (1, 2, 3), CHIMERA (FW, HW), MASTERKEY (1, 2, 3), SWAPGS, ITLB_Multihit, SRBDS, L1TF, etc. Más información en:

<https://cve.mitre.org>
<https://meltdownattack.com/>

y apropiarse de secretos almacenados en la memoria de otros programas". Como revelan en su estudio, la diferencia fundamental entre ambos es que Meltdown permite acceder a la memoria del sistema, mientras que Spectre permite acceder a la memoria de otras aplicaciones para robar esos datos.

Ya que Meltdown y Spectre son problemas de diseño en los procesadores, no es posible encontrar solución por Hardware para los procesadores existentes y dado que constantemente aparecen nuevas formas de explotar dichos fallos, la única manera de mantener el equipo de cómputo, comunicaciones y redes de internet a salvo es mediante Software que debe implementar las soluciones en los sistemas operativos. En particular en el Kernel de Linux se trabaja en parchar en cada versión del Kernel todos los fallos reportados, por esto y por otra gama de fallos e inseguridades es necesario mantener siempre el sistema operativo y sus aplicaciones actualizadas.

Como se había comentado anteriormente, estos problemas de diseño afectan a todos los procesadores Intel, AMD, IBM POWER y ARM. Eso incluye básicamente a todos los procesadores que están funcionando al día de hoy⁶⁰ en nuestros equipos, ya que estos procesadores llevan produciéndose desde 1995. Afecta a una amplia gama de sistemas.

En el momento de hacerse pública su existencia se incluían todos los dispositivos que no utilizasen una versión convenientemente parcheada de IOS, GNU/Linux, MacOS, Android, Windows y Android. Por lo tanto, muchos servidores y servicios en la nube se han visto impactados, así como potencialmente la mayoría de dispositivos inteligentes y sistemas embebidos que utilizan procesadores con arquitectura ARM (dispositivos móviles, televisores inteligentes y otros), incluyendo una amplia gama de equipo usado en redes. Se ha considerado que una solución basada únicamente en Software para estas fallas alenta los equipos de cómputo entre un 20 y un 40 por ciento dependiendo de la tarea que realizan y el procesador del equipo.

Memoria RAM La memoria RAM (Random Access Memory) o memoria de acceso aleatorio es un componente físico de nuestro ordenador, generalmente instalado sobre la misma placa base. La memoria RAM es extraíble y se puede ampliar mediante módulos de distintas capacidades.

La función de la memoria RAM es la de cargar los datos e instrucciones que se ejecutan en el procesador. Estas instrucciones y datos provienen del

⁶⁰Solo en el año 2021 se detectaron 16 vulnerabilidades en procesadores INTEL y 31 en los procesadores AMD.

sistema operativo, dispositivos de entrada y salida, de discos duros y todo lo que está instalado en el equipo.

En la memoria RAM se almacenan todos los datos e instrucciones de los programas que se están ejecutando, estas son enviadas desde las unidades de almacenamiento antes de su ejecución. De esta forma podremos tener disponibles todos los programas que ejecutamos. Se llama memoria de acceso aleatorio porque se puede leer y escribir en cualquiera de sus posiciones de memoria sin necesidad de respetar un orden secuencial para su acceso.

De forma general existen o han existido dos tipos de memorias RAM. Las de tipo asíncrono, que no cuentan con un reloj para poder sincronizarse con el procesador. Y las de tipo síncrono que son capaces de mantener la sincronización con el procesador para ganar en eficacia y eficiencia en el acceso y almacenamiento de información en ellas. Veamos cuales existen de cada tipo.

Memorias de Tipo Asíncrono o DRAM las primeras memorias DRAM (Dinamic RAM) o RAM dinámica eran de tipo asíncrono. Se denomina DRAM por su característica de almacenamiento de información de forma aleatoria y dinámica. Su estructura de transistor y condensador hace que para que un dato quede almacenado dentro una celda de memoria, será necesario alimentar el condensador de forma periódica.

Estas memorias dinámicas eran de tipo asíncrono, por lo que no existía un elemento capaz de sincronizar la frecuencia del procesador con la frecuencia de la propia memoria. Esto provocaba que existiera menor eficiencia en la comunicación entre estos dos elementos.

Memorias de Tipo Síncrono o SDRAM a diferencia de las anteriores esta memoria RAM dinámica cuenta con un reloj interno capaz de sincronizar esta con el procesador. De esta forma se mejoran notablemente los tiempos de acceso y la eficiencia de comunicación entre ambos elementos. Actualmente todas nuestras computadoras cuentan con este tipo de memorias operando en ellos. Las principales tipos de memoria son: DDR, DDR2, DDR3, DDR4 y la nueva DDR5. Donde las tasas de transferencia (GB/s) son: DDR (2.1 - 3.2), DDR2 (4.2 - 6.4), DDR3 (8.5 - 14.9), DDR4 (17 - 25.6) y DDR5 (38.4 - 51.2).

Aparte las características propias de cada una de las diferentes memorias DDR, la característica más importante es que, por ejemplo, en la memoria

DDR4 cuatro cores pueden acceder simultáneamente a ella y en la DDR5 serán cinco cores.

Caché L1, L2 y L3 La memoria Caché es otra de las especificaciones importantes de los procesadores, y sirve de manera esencial de la misma manera que la memoria RAM: como almacenamiento temporal de datos. No obstante, dado que la memoria Caché está en el procesador en sí, es mucho más rápida y el procesador puede acceder a ella de manera más eficiente, así que el tamaño de esta memoria puede tener un impacto bastante notable en el rendimiento, especialmente cuando se realizan tareas que demandan un uso intensivo del CPU como en el cómputo de alto desempeño o cómputo científico.

La Caché se divide en diferentes jerarquías de acceso:

- La Caché L1 es el primer sitio donde la CPU buscará información, pero también es la más pequeña y la más rápida, a veces para mayor eficiencia, la Caché L1 se subdivide en L1d (datos) y L1i (instrucciones), actualmente los procesadores modernos en cada core tiene su propio cache de datos e instrucciones.
- La Caché L2 suele ser más grande que la L1 pero es algo más lenta. Sin embargo, por norma general es la que mayor impacto tiene en el rendimiento, este también está incluido en cada core.
- La Caché L3 es mucho más grande que las anteriores, y generalmente se comparte entre todos los núcleos del procesador (a diferencia de las anteriores, que normalmente van ligadas a cada core). Este tercer nivel es en el que buscará el procesador la información tras no encontrarla en la L1 y L2, por lo que su tiempo de acceso es todavía mayor.

Para poner en contexto la relevancia de la memoria Caché, supongamos que el acceso a los datos de la memoria Caché L1 por el procesador es de dos ciclos de reloj, el acceso a la memoria Caché L2 es de 6 ciclos de reloj, el acceso a la memoria Caché L3 es de 12 ciclos y el acceso a la RAM es de 32 ciclos de reloj.

Además supongamos que la operación suma y resta necesitan de 2 ciclos de reloj para completar la operación una vez que cuente con los datos involucrados en dicha operación, que la multiplicación requiere 4 ciclos de reloj

para completar la operación, la división necesita 6 ciclos de reloj para completar la operación y estamos despreciando el tiempo necesario para poner los datos del Caché L1 a los registros del procesador para poder iniciar el cálculo, así también despreciamos el tiempo requerido para sacar el resultado de los registros del procesador al Caché L1.

Esto nos da una idea del número máximo teórico de operaciones básicas que un procesador puede realizar por segundo dependiendo de la velocidad de reloj de la CPU⁶¹.

Si nosotros necesitamos hacer la multiplicación de una matriz $\underline{\underline{A}}$ es de tamaño $n \times n$ por un vector \underline{u} de tamaño n y guardar el resultado en el vector \underline{f} de tamaño n . Entonces algunos escenarios son posibles:

1. Si el código del programa cabe en el Caché L1 de instrucciones y la matriz \underline{A} , los vectores \underline{u} y \underline{f} caben íntegramente en el Caché L1 de datos, entonces el procesador estará siendo utilizado de forma óptima al hacer los cálculos pues no tendrá tiempos muertos por espera de datos.
2. Si el código del programa cabe en el Caché L1 de instrucciones y los vectores \underline{u} y \underline{f} caben íntegramente en el Caché L1 de datos pero la matriz $\underline{\underline{A}}$ está dispersa entre los Cachés L1 y L2, entonces el procesador estará teniendo algunos tiempos muertos mientras carga la parte que necesita de la matriz del Caché L2 a L1 para hacer los cálculos y utilizado de forma óptima el procesador mientras no salga del Caché L1.
3. Si el código del programa cabe en el Caché L1 de instrucciones y los vectores \underline{u} y \underline{f} caben íntegramente en el Caché L1 de datos pero la matriz $\underline{\underline{A}}$ está dispersa entre los Cachés L1, L2 y L3, entonces el procesador estará teniendo muchos tiempos muertos mientras carga la parte que necesita de la matriz del Caché L3 y L2 a L1 para hacer los cálculos resultando en mediana eficiencia en el uso del procesador.
4. Si el código del programa cabe en el Caché L1 de instrucciones y los vectores \underline{u} y \underline{f} caben íntegramente en los Cachés L3, L2 y L1 pero

⁶¹Por ejemplo en un procesador AMD Ryzen 9 3900X con 12 Cores (2 Threads por Core) por procesador emulando un total de 24 Cores, corre a una frecuencia base de 3,340 MHz, con una frecuencia mínima de 2,200 MHz y máxima de 4,917 Mhz, con Caché L1d de 384 KiB, L1i de 384 KiB, Caché L2 de 6 MiB y Caché L3 de 64 MiB.

los datos de la matriz \underline{A} está dispersa entre la RAM y los Cachés L3, L2 y L1, entonces el procesador estará teniendo un exceso de tiempos muertos mientras carga la parte que necesita de la matriz de la RAM a los Cachés L3, L2 y L1 para hacer los cálculos resultando en una gran pérdida de eficiencia en el uso del procesador.

Además, debemos recordar que la computadora moderna nunca dedica el cien por ciento del CPU a un solo programa, ya que los equipos son multitarea⁶² y multiusuario⁶³ por lo que la conmutación de procesos (que se realiza cada cierta cantidad de milisegundos) degrada aún más la eficiencia computacional de los procesos que demandan un uso intensivo de CPU⁶⁴.

Last Level Cache se le llama Last Level Cache siempre al último nivel de Caché de una CPU, existen dos tipos:

- Last Level Cache Estándar.
- Victim Cache.

⁶²Cuentan con la capacidad para ejecutar varios procesos simultáneamente en uno o más procesadores, para ello necesitan hacer uso de la conmutación de tareas, es decir, cada cierto tiempo detiene el programa que está corriendo y guardan sus datos, para poder cargar en memoria otro programa y sus respectivos datos y así reiniciar su ejecución por un período determinado de tiempo, una vez concluido su tiempo de ejecución se reinicia la conmutación de tareas con otro proceso.

⁶³Se refiere a todos aquellos sistemas operativos que permiten el empleo de sus procesamientos y servicios al mismo tiempo. Así, el sistema operativo cuenta con la capacidad de satisfacer las necesidades de varios usuarios al mismo tiempo, siendo capaz de gestionar y compartir sus recursos en función del número de usuarios que estén conectados a la vez.

⁶⁴Actualmente existen una gran cantidad de distribuciones de GNU/Linux que vienen muy optimizadas intentando conseguir la mejor desenvolvura de su arquitectura y configuraciones de serie. En el caso de la configuración por omisión de Debian GNU/Linux y Ubuntu, están pensadas para que sean lo más robusta posible y que se use en todas las circunstancias imaginables, por ello están optimizadas de forma muy conservadora para tener un equilibrio entre eficiencia y consumo de energía. Pero es posible agregar uno o más Kernels GNU/Linux generados por terceros que contenga las optimizaciones necesarias para hacer más eficiente y competitivo en cuestiones de gestión y ahorro de recursos del sistema.

Hay varias opciones del Kernel GNU/Linux optimizado (**Liquorix** viene optimizado para multimedia y Juegos, por otro lado **XanMod** tiene uno para propósito general, otro aplicaciones críticas en tiempo real y otro más para cálculos intensivos) de las últimas versiones estable del Kernel.

Una Victim Cache no actúa como la Caché de último nivel de una CPU, sino que en ese caso lo hace el penúltimo nivel y en la Victim Cache acaban los últimos datos descartados de la Caché y que han sido volcados en la RAM, los cuales son copiados en la Victim Cache para poder acceder a ellos más rápido.

Smart Cache la Smart Cache (o Caché) es esencialmente L3 pero optimizada por Intel para ser más eficiente a la hora de compartir la información en los núcleos de la CPU. A efectos prácticos, se comporta de igual manera que la Caché L3.

Disco Son dispositivos no volátiles (los hay del orden de 24 TB y continuamente incrementan su capacidad⁶⁵), lo que significa que retienen datos incluso cuando no tienen energía. La información almacenada permanece segura e intacta a menos que el disco duro sea destruido o interferido. La información se almacena o se recupera de manera aleatoria en lugar de acceso secuencial. Esto implica que se puede acceder a los bloques de datos en cualquier momento sin necesidad de pasar por otros bloques de datos.

⁶⁵Durante años la tecnología más popular entre los fabricantes ha sido la PMR (Perpendicular Magnetic Recording), también conocida como CMR (Conventional Magnetic Recording). A esta tecnología luego se le sumó la variante SMR (Shingled Magnetic Recording), que lograba aumentar la densidad de grabación, pero lo hacía sacrificando velocidad de transferencia y fiabilidad de las operaciones.

Western Digital ha creado la tecnología ePMR (energy-assisted Perpendicular Magnetic Recording) que permite ofrecer mayores densidades de grabación y, según este fabricante, mejorar la fiabilidad de las escrituras y evitar así los sacrificios que había que hacer con SMR (en los últimos tiempos han aparecido unidades de 20 y 24 TB basadas en dicha tecnología).

Más interesante aún es el sistema de grabación MAMR (Microwave-Assisted Magnetic Recording) que hace uso de microondas para calentar el medio de almacenamiento y así lograr mejorar densidad de grabación y fiabilidad de lecturas y escrituras. Hace años ya prometían que gracias a esta tecnología contaríamos con unidades de 40 TB en 2025, pero parece que dicho logro aún tardará en llegar.

Esta otra opción es una alternativa a las microondas, pero en HAMR (Heat-Assisted Magnetic Recording) el proceso de calentar el medio de almacenamiento lo realiza un láser. Toshiba prometió lanzar unidades de más de 30 TB en 2024, mientras que Seagate también quería ofrecer esa capacidad de forma inminente para luego dar el salto a unidades de 40 TB e incluso a los 100 TB que plantean para 2030. Ahí es donde probablemente entre en acción la evolución de HAMR+, que tratará de exprimir aún más la densidad de grabación.

Actualmente, podemos agrupar los discos duros disponibles en cuatro tipos:

- Parallel Advanced Technology Attachment (PATA)
- Serial ATA (SATA)
- Interfaz de sistema de computadora pequeña (SCSI)
- Adjunto de tecnología avanzada paralela
- Unidades de estado sólido (SSD)

En promedio, las velocidades máximas de los discos actuales son:

- Disco SATA3 de 5,400 RPM, Lectura: 102 MB/s, Escritura: 96 MB/s
- Disco SATA3 de 7,200 RPM, Lectura: 272 MB/S, Escritura: 200 MB/s
- Disco SSD SATA, Lectura 550 MB/s, Escritura 520 MB/s
- Disco SSD NVMe, Lectura 6,600 MB/s, Escritura 5,500 MB/s
- Disco SSD PCI 5.0, Lectura 13,000 MB/s, Escritura 12,000 MB/s
- Unidad Flash USB⁶⁶ 2.0, 35 MB/s
- Unidad Flash USB 3.0 o 3.1 gen 1, 5 Gbit/s
- Unidad Flash USB 3.0 o 3.1 gen 2, 10 Gbit/s
- Unidad Flash USB 3.2 gen 2x2, 20 Gbit/s

⁶⁶Los colores en los puertos USB son: USB 1.X blanco (12 Mbps), USB 2.X Negro (480 Mbps), USB 3.0 Azul oscuro (5 Gbps), USB 3.1 Azul claro (10 Gbps), USB 3.2 Rojo (20 Gbps). En el caso del USB de color Amarillo, éste es un puerto de carga aún con el dispositivo apagado.

Disco de Estado Sólido SSD Estos son los últimos avances en tecnología de almacenamiento que tenemos en la industria de las computadoras. Son totalmente diferentes de las otras unidades en que no consisten en partes móviles. Tampoco almacenan datos utilizando magnetismo. En su lugar, hacen uso de la tecnología de memoria flash, circuitos integrados o dispositivos semiconductores para almacenar datos de forma permanente, al menos hasta que se borren. Estas son algunas de sus ventajas.

- Acceso a datos más rápido
- Menos susceptible a los golpes
- Menores tiempos de acceso y latencia
- Menos consumo de energía

Los SSD actuales están disponibles tanto en versiones SATA como en versiones M.2, U.2 y en formato de tarjeta PCI Express 4.0. Los tres últimos hacen uso del protocolo NVMe y la interfaz PCI Express 4.0 x4, lo que les permite superar los 6,600 MB/s de velocidades de lectura y escritura, frente a los 550 MB/s que suelen alcanzar como máximo las unidades SATA. La nueva versión PCI 5.0 ofrece un ancho de banda de 32 GT/s el doble de PCI 4.0, permitiendo discos SSD con 13,000 MS/s de velocidad de lectura secuencial y realizar hasta 2,500K operaciones por segundo de lectura aleatoria y tamaño máximo 15.36 TB a un precio exorbitante.

Tarjetas microSD Cuando compramos una tarjeta microSD para ampliar el almacenamiento de nuestro Smartphone, cámara, tableta o cualquier otro dispositivo electrónico la mayoría de la gente normalmente solo se fija en la capacidad de almacenamiento. Ahora bien, ¿qué significan todas esas etiquetas y nombres que llevan adscritas las microSD? ¿Cuál es la diferencia entre una microSDXC y una microSDHC? ¿Es mejor una UHS-I o una UHS-II? ¿Qué quiere decir que una tarjeta es A1 y V30? A continuación, intentamos aclarar toda esta nomenclatura.

Lo primero que tenemos que tener claro es que cuando analizamos una tarjeta micro SD existen multitud de factores que limitan su velocidad y capacidad de almacenamiento. Su rendimiento depende de factores como el tipo de tarjeta que estamos usando, su clase, y otros detalles como el tipo de bus o el número de operaciones que puede realizar por segundo.

Tipos de tarjetas microSD actualmente existen 4 generaciones distintas de tarjetas de memoria microSD. Cuanto más modernas sean, mayores velocidades y almacenamiento podrán ofrecer:

- Tarjetas micro SD (Secure Digital): Estas son las memorias de primera generación. Las desarrolló el fabricante SanDisk y fueron las primeras en utilizar el formato de 15 x 11 x 1 milímetros. Su capacidad máxima es de 32 GB.
- Tarjetas micro SDHC (Secure Digital High Capacity): Tarjetas de segunda generación. Cuentan con un bus de datos mejorado que permite alcanzar velocidades superiores, aunque su capacidad máxima sigue siendo de 32 GB.
- Tarjetas micro SDXC (Secure Digital Extended Capacity): Estas micro SD utilizan un sistema de archivos exFAT y su velocidad de transferencia puede llegar hasta los 312 MB/s. Su capacidad de almacenamiento puede llegar hasta los 2 TB y es el tipo de tarjeta más común utilizado a día de hoy.
- Tarjetas micro SDUC: Estas son las tarjetas de memoria más modernas y punteras. Utilizan el sistema de archivos exFAT y permiten almacenar entre 2 TB y 128 TB de datos.

Evidentemente con esto no es suficiente. Si queremos tener una idea aproximada de la velocidad de la micro SD tendremos que fijarnos en aspectos como la clase y tipo de bus que emplea.

La clase es una característica que nos indica la velocidad de transferencia de datos mínima de la tarjeta de memoria. Actualmente hay 4 tipos de clase diferentes:

- Clase 2: Velocidad mínima de 2 MB/s
- Clase 4: Velocidad mínima de 4 MB/s
- Clase 6: Velocidad mínima de 6 MB/s
- Clase 10: Velocidad mínima de 10MB/s

Hoy en día la mayoría de tarjetas micro SD son de clase 10, ya que son capaces de transferir más de 10 MB/s y superan esa cifra fácilmente.

El bus determina la velocidad de la interfaz de la tarjeta de memoria, y nos puede servir como indicativo para conocer la rapidez con la que se pueden leer y escribir los datos:

- Bus estándar: Su velocidad de transferencia alcanza hasta los 12.5 MB/s. Es el tipo de bus utilizado en tarjetas de clase 2, 4 y 6.
- Bus de alta velocidad (High Speed): Se utiliza en tarjetas de clase 10 y alcanza una velocidad de hasta 25 MB/s.
- Bus Ultra High Speed (UHS): Estos son los buses con la interfaz más rápida, y existen varios tipos:
 - UHS-I: Hay dos tipos de buses UHS-I. Por un lado, tenemos el UHS-I clase 1 (U1) que alcanza velocidades de 50 MB/s. Y luego tenemos el UHS-I clase 3 (U3) que llega hasta los 104 MB/s.
 - UHS-II: Alcanza velocidades de transferencia hasta 312 MB/s.
 - UHS-III: Velocidades de transferencia de datos que alcanzan hasta los 624 MB/s.
- SD-Express: Este es el tipo de bus más potente de todos, llegando hasta los 985 MB/s.

Como referencia, te interesará saber que actualmente la mayoría de tarjetas micro SD de gama media utilizan un bus UHS-I de clase 3 (U3) con velocidades de lectura de hasta 104 MB/s.

Otro factor importante es la velocidad de lectura y escritura aleatoria (IOPS) u operaciones por segundo que puede realizar una tarjeta. Este dato determina el rendimiento mínimo en la lectura y escritura aleatoria de la SD:

- Clase de rendimiento de aplicación A1: Las tarjetas A1 tienen una velocidad mínima de lectura aleatoria de 1,500 IOPS, y una velocidad mínima de escritura aleatoria de 500 IOPS.
- Clase de rendimiento de aplicación A2: Las tarjetas A2 ofrecen velocidades superiores, con 4,000 IOPS de lectura y 2,000 IOPS de escritura.

Normalmente con una tarjeta A1 es más que suficiente para tareas del día a día, aunque si necesitamos un rendimiento superior, por ejemplo, para ejecutar aplicaciones desde la SD o jugar a videojuegos, las tarjetas A2 ofrecen un mejor rendimiento.

La velocidad de escritura aleatoria (A1 y A2) es un dato relevante para los Smartphones y tabletas, pero si tenemos una cámara de grabación, una Action camera o un Dron, la característica en la que nos tenemos que fijar es en el Velocidad de escritura secuencial (sistema V) que utiliza (en inglés, Video Speed Class). O dicho de otra forma, en su velocidad de escritura secuencial.

Esta característica nos indica la cantidad de datos que se pueden grabar en la micro SD de forma constante sin bajar de una velocidad mínima. Esto resulta esencial cuando queremos grabar vídeos en alta y ultra-alta definición:

- V30 (Video Speed Class 30): Velocidad de escritura mínima de 30 MB/s
- V60 (Video Speed Class 60): Velocidad de escritura mínima de 60 MB/s
- V90 (Video Speed Class 90): Velocidad de escritura mínima de 90 MB/s

Por ejemplo, si vamos a grabar vídeo en resolución 4K directamente en la tarjeta micro SD, es necesario que la velocidad V sea lo máximo posible, especialmente si vamos a utilizar un amplio BitRate con bajos niveles de compresión, o calidades superiores como el 8K.

Cintas Magnéticas En el año 2010, se comunicó que todos los datos utilizados para el proyecto del satélite Nimbus se recuperaron de cintas que en ese momento tenían 46 años. A partir de dicho comunicado se extendió el uso de la cinta magnética para almacenamiento de datos en todo el mundo.

En un mundo altamente digital, la cinta magnética es una de las pocas tecnologías que utiliza señales analógicas para mover parte de los datos, en su esencia, la cinta se parece mucho a un HDD, utiliza materiales de base magnética, pero en este caso, la cinta es literalmente una base de material generalmente nailon que tiene un revestimiento magnético. Y en lugar de un disco giratorio, la cinta entra, está enhebrada. Puede parecer una cinta VHS, pero es mucho más robusta.

La cinta se mueve linealmente hacia la unidad de cinta y hacia el cartucho de cinta. Para escribir, el cabezal de la cinta toma señales electrónicas y crea un mini campo magnético que puede cambiar la polaridad del material de la película para formar un patrón de ceros y unos. Una vez que los datos se escriben en la cinta, no se pueden cambiar (pero se pueden borrar y reescribir).

La inmutabilidad y las capacidades de encriptación de la cinta, así como la simplicidad de crear un espacio para almacenarla en una bóveda hacen de la cinta un arma clave para asegurar que los datos sobrevivan frente al Ransomware. Uno de los grandes productores de cintas en la actualidad es IBM, los cuales argumentan que esta función hace que la cinta sea el medio ideal para almacenar datos de archivo a los que no es necesario acceder con frecuencia.

La cinta también puede servir como una copia de seguridad y de versiones fuera de línea de archivos importantes o confidenciales que son resistentes a los ataques cibernéticos. Los tipos de datos que permanecen en la cinta abarcan registros financieros, registros médicos, información de identificación personal y documentos que forman parte de una retención legal de múltiples gobiernos.

Una sola cinta mide aproximadamente 3 pulgadas por 3 pulgadas y 3/4 de pulgada de grosor. Es más pequeño que una unidad de disco duro (HDD), pero pesa alrededor de 0.6 kilogramos. Un cartucho puede almacenar 18 Terabytes de datos sin comprimir y 45 Terabytes comprimidos. IBM está trabajando para duplicar esta capacidad en la próxima generación de la tecnología. En cuanto a la velocidad de recuperación, se obtiene un flujo de datos de una unidad de cinta de 1,000 Megabits por segundo, comprimidos.

Una biblioteca de cintas puede variar en tamaño desde algo que puede poner en su escritorio hasta algo que es del tamaño de un refrigerador pequeño (alrededor de 8 pies cuadrados). La pequeña biblioteca del tamaño de un refrigerador tiene capacidad para 1584 cartuchos. IBM promociona que su biblioteca Diamondback será la biblioteca de cintas más densa del mercado. Podrá contener 69 Petabytes de información mientras ocupa menos de 8 pies cuadrados de espacio.

La cinta magnética supera al disco duro y al flash en cuanto a longevidad, costo financiero y costo de huella de carbono, pero pierde en velocidad de acceso. Las cintas no son recomendables para poner datos de producción en vivo o incluso copias de seguridad, pero son perfectas para cualquier información a la que se acceda con poca frecuencia y que deba conservarse durante

mucho tiempo, como registros médicos o datos de archivo.

Es conocido que muchas empresas han usado y seguirán usando cinta magnética en sus operaciones, entre las que destacan las hiperescalas (empresas que han crecido tanto que ofrecen sus propias infraestructuras o tienen datos masivos como resultado de su infraestructura) siempre necesitan muchas formas diferentes de tecnología para manejar la variedad de datos que ingresan a sus sistemas para alimentar una gama de servicios. Entre otras destacan: Bancos, Gobiernos, Milicia y organizaciones como CERN, así como corporaciones como Amazon, Google, Meta, Baidu, Alibaba y Tencent.

Tarjeta Gráfica La tarjeta gráfica o tarjeta de vídeo es un componente que viene integrado en la placa base de la computadora o se instala aparte para ampliar sus capacidades. Concretamente, esta tarjeta está dedicada al procesamiento de datos relacionados con el vídeo y las imágenes que se están reproduciendo en la computadora.

Procesador Gráfico GPU el corazón de la tarjeta gráfica es la GPU o Unidad de procesamiento gráfico, un circuito muy complejo que integra varios miles de millones de transistores diminutos y puede tener desde uno a miles de núcleos (ya es común encontrar computadoras personales con tarjeta de 10496 cores y 24 GB de GRAM) que tienen capacidad de procesamiento independiente. De la cantidad y capacidad de estos núcleos dependerá la potencia.

Así como los procesadores centrales de las CPU, están diseñados con pocos núcleos pero altas frecuencias de reloj, las GPU tienden al concepto opuesto, contando con grandes cantidades de núcleos con frecuencias de reloj relativamente bajas. Luego tienes la memoria gráfica de acceso aleatorio o GRAM, que son Chips de memoria que almacenan y transportan información entre sí. Esta memoria no es algo que vaya a determinar de forma importante el rendimiento máximo de una tarjeta gráfica, aunque si no es suficiente puede acabar lastrando y limitando la potencia de la CPU.

La idea de usar esa potencia para otros menesteres se denomina GPGPU (General Purpose Computation on Graphics Processing Units) o GPU Computing. En el momento en el que las tarjetas gráficas permiten que se programen funciones sobre su Hardware se empieza a hacer uso de GPGPU. Al principio era necesario utilizar los lenguajes enfocados a la visualización en pantalla (como OpenGL) para realizar otros cálculos no relacionados con los

gráficos. Esto implicaba el uso de funciones muy poco flexibles, originalmente diseñadas para otros fines, lo que hacía que la labor de programar para tal fin fuese realmente tediosa y complicada.

Para facilitar el empleo de las tarjetas gráficas para cualquier uso no vinculado con los gráficos, NVidia desarrolló toda una tecnología alrededor de la tarjeta, que permitía usar la misma para cualquier tarea: CUDA. ATI, la principal (y actualmente casi única) competidora, un poco más tarde haría lo propio lanzando su propia tecnología: Stream. En un principio las tarjetas gráficas solo trabajaban con aritmética de 32 bits, pero en la actualidad ya se cuenta con aritmética de 64 bit (para lograr esto, muchas tarjetas usan dos de sus cores de 32 bits para emular uno de 64 bits, reduciendo su número de cores útiles a la mitad).

Procesador Gráfico Integrado muchos procesadores CPU incorporan una o más GPU en su interior, llamada gráfica integrada (iGPU Integrated Graphics Processing Units o APU Accelerated Processing Unit). Generalmente es muy poco potente, pero lo suficiente para realizar tareas básicas como navegar por Internet, ver vídeos, e incluso para algunos juegos básicos, especialmente en las últimas generaciones puesto que cada vez son más potentes. No obstante, en las últimas generaciones de procesadores cada vez se están introduciendo gráficos integrados más potentes, y ya son capaces de manejar varios monitores, resoluciones 4K e incluso son capaces de mover algunos juegos a una tasa digna de FPS.

Tarjeta Gráfica por ejemplo, la tarjeta gráfica de AMD Instinct MI200 cuenta con más de 200,000 cores y 128 GB de HBM2, NVidia GEFORCE RTX 3090 proporciona 10,496 cores y 24 GB de GDDR6x, NVidia A100 cuenta con 80 GB de memoria HBM2 6192 cores y 432 núcleos tensor⁶⁷, mientras que la tarjeta NVidia Titan RTX proporciona 130 Tensor TFLOP de rendimiento, 576 núcleos tensores y 24 GB de memoria GDDR6.

Por otra parte, Intel ha desarrollado una aceleradora gráfica Artic Sound-M pensada para centro de datos (especialmente diseñada para juegos en la nube) que utiliza una GPU DG2 Xe-HPG que viene con una configuración de 512 unidades de ejecución lo que equivale a 4,096 Shaders, por ejemplo, esta

⁶⁷Un Tensor core (o núcleos Tensor) calculan la operación de una matriz 4x4 completa, la cual se calcula por reloj. Estos núcleos pueden multiplicar dos matrices FP16 4x4 y sumar la matriz FP32 al acumulador.

aceleradora puede manejar hasta 8 Streamings simultáneos de video 4K o más de 30 si el vídeo es en 1080p y cuenta con más de 60 funciones virtualizadas.

En agosto del 2021, se anunció la construcción de la supercomputadora Polaris, acelerado por 2240 GPU NVIDIA A100 Tensor Core, el sistema puede alcanzar casi 1.4 exaflops de rendimiento teórico de IA y aproximadamente 44 petaflops de rendimiento máximo de doble precisión. Polaris, que será construido por Hewlett Packard Enterprise, combinará simulación y aprendizaje automático al abordar cargas de trabajo informáticas de alto rendimiento de inteligencia artificial y con uso intensivo de datos, impulsadas por 560 nodos en total, cada uno con cuatro GPU NVIDIA A100.

Tipos de Redes Según el Medio Físico Si bien, nuestros dispositivos de cómputo pueden funcionar sin conexión de red, estos se ven inmediatamente limitados. La red nos permite conectarnos a Internet que es el camino por el cual nos conectamos con el mundo. Las formas de conectar nuestros equipos a Internet en un principio fue exclusivamente por red alámbrica, desde ya hace unos años a la fecha se dispone de conexión a red alámbrica e inalámbrica, pero actualmente nuestros dispositivos cuentan casi exclusivamente con conexión inalámbrica.

¿Qué es el ancho de banda? Se trata de la capacidad máxima y la cantidad de datos que se pueden transmitir a través de una conexión (de internet, por ejemplo), en un momento determinado. Algo que debemos tener claro es que el ancho de banda de red es fundamental para la calidad y velocidad de la conexión.

El ancho de banda se mide en *bit/s* o en sus múltiplos *k/bits* o *m/bits* por segundo. Y para la mayoría de los casos, debemos asegurarnos siempre de tener el mayor ancho de banda que nos sea posible, porque de esta manera podremos tener una mejor y más rápida transferencia de datos.

¿Qué es entonces la velocidad de transmisión? Este término se puede definir como la velocidad a la que se transmite la información. Cuando un usuario adquiere un paquete con una empresa prestadora de servicios de internet, recibe, por ejemplo, 10 mbps, 30 mbps, 100 mbps, etc. Y esto se refiere a la cantidad de datos que podemos descargar o subir a la red. Como recomendación, lo indicado es que para que la velocidad pueda existir será

necesario tener un ancho de banda igual o superior a la velocidad contratada en el paquete de servicio.

Normalmente vemos en los anuncios de todos los operadores, que estos ofrecen una cantidad cualquiera de megas de navegación; este valor numérico corresponde a la velocidad de descarga únicamente. Para encontrar la velocidad de subida, es necesario acceder a un test que nos revele cuál es el resultado y si lo que nos prometen, es verdad o no.

¿Qué es la latencia? Es el tiempo total que transcurre desde que enviamos una información, hasta que la misma llega a un receptor. Su valor de medición se hace en milisegundos, también se conoce como *Ping* y está presente en actividades que realiza cotidianamente como jugar en línea o hacer videollamadas.

¿Altera la latencia la velocidad de la conexión? La velocidad de conexión influye en la latencia una vez que pasamos de un rango predeterminado. Poniéndolo en un ejemplo, si tenemos una conexión a Internet de 1 Mbps y la comparamos con una conexión de 100 Mbps, dependiendo del tamaño del paquete se notará una mejora grande en la velocidad. Otros factores que importan a la hora de hablar de latencia son el estar conectado a internet por Wi-fi o un cable, si tiene servicio de fibra óptica o qué tanta distancia hay entre su ordenador y un Router, etc.

La latencia en la conexión también es la suma de otros retardos:

- De procesamiento: se define en el tiempo que tardan los Routers en examinar la cabecera y a su vez, la respuesta en determinar a dónde hay que enviar cualquier paquete haciendo una previa comprobación de sus tablas de enrutamiento.
- De cola: tiempo de espera del paquete para poder ser transmitido a través de un enlace físico. Cabe anotar que no podemos saber previamente si va a haber un retardo de cola o no, ya que este cambia en tiempo real.
- De transmisión: es el tiempo que tarda el paquete en arribar hasta el siguiente nodo o destino final.

- De propagación: es el tiempo que tarda un bit en propagarse desde un punto cualquiera de origen hasta llegar a uno de destino. Su velocidad depende del medio físico por el que se transporte.

El retardo total es la suma de todos los retardos anteriormente enunciados.

Comúnmente se asocia la latencia con la banda de ancha, pero existe una diferencia sustancial entre ambas. Si bien ambas afectan la velocidad de la conexión, la banda ancha permite que se pueda transmitir una gran cantidad de datos, mientras la latencia determina a qué velocidad se transmite esa cantidad de datos.

Si bien, tener red nos permite estar conectados, tenemos una gran limitación por las velocidades de conexión a las que tendremos acceso según el tipo de medio físico que usemos para conectarnos, así como el número de dispositivos con los que compartamos la conexión. Las redes inalámbricas parecen ser omnipresentes, pero las velocidades de interconexión dejan mucho que desear como veremos a continuación.

Redes alámbricas se comunica a través de cables de datos (generalmente basada en Ethernet). Los cables de datos, conocidos como cables de red de Ethernet o cables con hilos conductores (CAT5), conectan computadoras y otros dispositivos que forman las redes. Las redes alámbricas son mejores cuando se necesita mover grandes cantidades de datos a altas velocidades, como medios multimedia de calidad profesional.

Ventajas

- Costos relativamente bajos
- Ofrece el máximo rendimiento posible
- Mayor velocidad - cable de Ethernet estándar hasta 1 Gbps⁶⁸
- Mayor rendimiento de Voz sobre IP.
- Mejores estándares Ethernet en la industria.
- Mayor capacidad de ancho de banda por cables.

⁶⁸El término bps se refiere a transmitir Bits por segundo (se requieren 8 Bits para formar un Byte). Por eso el término de 1,000 Mbps es equivalente a 125 MB/s.

- Aplicaciones que utilizan un ancho de banda continuo.

Desventajas

- El costo de instalación siempre ha sido un problema común en este tipo de tecnología, ya que el estudio de instalación, canaletas, conectores, cables y otros suman costos muy elevados en algunas ocasiones.
- El acceso físico es uno de los problemas más comunes dentro de las redes alámbricas. Ya que para llegar a ciertos lugares, es muy complicado el paso de los cables a través de las paredes de concreto u otros obstáculos.
- Dificultad y expectativas de expansión es otro de los problemas más comunes, ya que cuando pensamos tener un número definido de nodos en una oficina, la mayoría del tiempo hay necesidades de construir uno nuevo y ya no tenemos espacio en los Switches instalados.

Hoy en día se puede hacer la siguiente clasificación de las redes de protocolo Ethernet para cable y fibra óptica⁶⁹:

- Ethernet, que alcanza no más de 10 Mbps de velocidad
- Fast Ethernet, que puede trabajar con hasta 100 Mbps
- Gigabit Ethernet, alcanza hasta 1 Gbps (1000 Mbps aprox.)
- 2.5 y 5 Gigabit Ethernet hasta 2.5 Gbps si usamos cableado Cat 5a y nada menos que 5 Gbps con cableado Cat 6

⁶⁹El récord mundial en mayo del 2022 de transmisión en fibra óptica es de 1.02 petabits por segundo enviados a través de 51.7 kilómetros (que podría transmitir hasta 10 millones de canales por segundo de vídeo a resolución 8K), que rompe al récord anterior de 319 terabits por segundo sobre una distancia de 1,800 millas (con el estimado de descarga de 80,000 películas simultáneamente en un segundo).

En 2023 los investigadores de Electronics and Computer Engineering de Aston University en U.K. alcanzaron una velocidad de 301 terabits por segundo (Tbps), equivalente a transferir 1,800 películas 4K a través de Internet en un segundo, utilizando cables de fibra óptica existentes. En comparación, la velocidad media de banda ancha fija en los EE. UU. es de 242,38 megabits por segundo (Mbps), según Speed Test.

Los resultados de la prueba, que se realizaron utilizando el tipo de cables de fibra ya tendidos en el suelo, lograron esta velocidad vertiginosa enviando luz infrarroja a través de hilos tubulares de vidrio, que es como funciona generalmente la banda ancha de fibra óptica. Pero aprovecharon una banda espectral que nunca se ha utilizado en sistemas comerciales, llamada "banda E", utilizando dispositivos nuevos hechos a medida.

- 10 Gigabit Ethernet, que puede alcanzar hasta los 10 GMbps
- 1000GbE para alcanzar la Ethernet Terabit (125 Gbytes)

La categoría del cable o el tipo de fibra óptica determina la velocidad máxima soportada por cada tipo de cable, pero aunque haya cables con la misma velocidad hay otros factores que determinan su usabilidad, como el ancho de banda o la frecuencia. La frecuencia o ancho de banda determina la potencia de la red, a mayor frecuencia mayor ancho de banda y menor pérdida de datos. Este factor es importante si vamos a conectar varios equipos al mismo cable de red o vamos a hacer una gran tirada de cable, ya que cuanto más largo sea el cable de red más potencia perderá. Siempre tendrá más velocidad un cable corto que un cable largo, pero si el ancho de banda es amplio tardará más metros en perder potencia y velocidad.

Ethernet es el estándar que domina la gran mayoría de mercado, presente de manera casi exclusiva tanto en mercado doméstico, como en pequeña y mediana empresa representa también un porcentaje muy significativo en grandes centros de datos. Pero existen otros protocolos que se pueden situar en el mismo nivel de calidad que Gigabit Ethernet como InfiniBand.

InfiniBand es un bus serie bidireccional de comunicaciones de alta velocidad en las que las rápidas comunicaciones entre servidores son críticas para el rendimiento, llegando a ofrecer velocidades de hasta 2.0 Gbps netos en cada dirección del enlace en un nodo simple, 4 Gbps netos en un nodo doble y hasta 8 Gbps netos en un nodo cuadruple. Estos nodos a su vez se pueden agrupar en grupos de 4 ó 12 enlaces llegando a velocidades de hasta 96 Gbps netos en un grupo de 12 nodos cuádruples. El factor de velocidad neta viene relacionado con que Infiniband de cada 10 bits que transmite 8 de ellos son datos, basándose en la codificación 8B/10B.

Recientemente se han implementado sistemas en los que ya no se utiliza esta codificación 8B/10B sino la 64B/66B que permite mejorar el porcentaje de datos útiles por trama enviada y que ha permitido los nodos FDR-10 (Fourteen Data Rate-10 a 10 Gbps), FDR (Fourteen Data Rate a 13.64 Gbps) y EDR (Enhanced Data Rate a 25 Gbps). Este último en un grupo de 12 nodos proporciona hasta 300 Gbps. Los últimos desarrollos de Gigabit Ethernet, proporcionan hasta 100 Gbps por puerto.

Estas enormes velocidades de conexión hacen que Infiniband sea una conexión con una muy importante presencia en superordenadores y clústers, por ejemplo del top 500 de superordenadores en 2020, 226 están conectados

internamente con Infiniband, 188 lo están con Gigabit Ethernet y el resto con Myrinet, Cray, Fat Tree u otras interconexiones a medida.

Una de las principales ventajas de Infiniband sobre Ethernet es su bajísima latencia, por ejemplo y basándonos en los datos del estudio de Qlogic "Introduction to Ethernet Latency, an explanation to Latency and Latency measurement", la latencia en 10 Gbps Ethernet se sitúa en 5 microsegundos mientras que la de Infiniband se sitúa por debajo de los 3 microsegundos.

Los sistemas de conmutadores inteligentes InfiniBand de NVIDIA Mellanox ofrecen el mayor rendimiento y densidad de puertos para computación de alto rendimiento (HPC), IA, Web 2.0, Big Data, nubes y centros de datos empresariales. La compatibilidad con configuraciones de 36 a 800 puertos a hasta 200 Gbps por puerto permite que los clústeres de cómputo y los centros de datos convergentes funcionen a cualquier escala, lo que reduce los costos operativos y la complejidad de la infraestructura.

Redes inalámbricas: es una red en la que dos o más terminales (ordenadores, tabletas, teléfonos inteligentes, etc.) se pueden comunicar sin la necesidad de una conexión por cable. Se basan en un enlace que utiliza ondas electromagnéticas (radio e infrarrojo) en lugar de cableado estándar. Permiten que los dispositivos remotos se conecten sin dificultad, ya se encuentren a unos metros de distancia como a varios kilómetros.

Asimismo, la instalación de estas redes no requiere de ningún cambio significativo en la infraestructura existente como pasa con las redes cableadas. Tampoco hay necesidad de agujerear las paredes para pasar cables ni de instalar porta cables o conectores. Esto ha hecho que el uso de esta tecnología se extienda con rapidez.

Tipos de redes inalámbricas

- LAN Inalámbrica: Red de área local inalámbrica. También puede ser una red de área metropolitana inalámbrica.
- GSM (Global System for Mobile Communications): la red GSM es utilizada mayormente por teléfonos celulares.
- D-AMPS (Digital Advanced Mobile Phone Service): está siendo reemplazada por el sistema GSM.
- Fixed Wireless Data: Es un tipo de red inalámbrica de datos que puede ser usada para conectar dos o más edificios juntos para extender o

compartir el ancho de banda de una red sin que exista cableado físico entre los edificios.

- Wi-Fi⁷⁰: es uno de los sistemas más utilizados para la creación de redes inalámbricas en computadoras, permitiendo acceso a recursos remotos como internet e impresoras. Utiliza ondas de radio.

Ventajas

- La instalación de redes inalámbricas suele ser más económica.
- Su instalación también es más sencilla.
- Permiten gran alcance; las redes hogareñas inalámbricas suelen tener hasta 100 metros desde la base transmisora.
- Permite la conexión de gran cantidad de dispositivos móviles. En las redes cableadas mientras más dispositivos haya, más complicado será el entramado de cables.

⁷⁰Notemos que una conexión de WiFi tradicional sin obstáculos con una intensidad de banda de 2.4 GHz puede tener un alcance máximo de 46 metros y para la banda de 5.0 GHz un alcance de 15 metros. Pero hay muchos objetos que interfieren con la señal del Router del WiFi y el sitio en el que colocamos nuestro dispositivo inalámbrico como computadora o teléfono inteligente:

- Superficies y/o objetos de metal o vidrio blindado
- Refrigeradores, lavadoras y radiadores
- Hornos de microondas, cámaras Web, monitores de bebés y teléfonos inalámbricos
- Paredes y muros
- Dispositivos arquitectónicos que funcionan como una jaula de Faraday (es un contenedor recubierto por materiales conductores de electricidad como mallas metálicas, papel aluminio, cajas o cestos de basura de acero que funcionan como un blindaje contra los efectos de un campo eléctrico proveniente del exterior).

En caso de que varios dispositivos se conecten a la red, se puede optar por colocar el Router en un punto medio, para que ninguna zona quede sin cobertura. Por otra parte, para una mejor conexión, también procura que el Router se encuentre en una zona elevada, pues esto mejorará el alcance de la señal inalámbrica. Una excelente opción para mejorar la calidad del internet inalámbrico cuando hay obstáculos es utilizar un repetidor de WiFi, este dispositivo es muy útil para amplificar la señal y llevarla a más sitios de nuestra red.

- Posibilidad de conectar nodos a grandes distancias sin cableado, en el caso de las redes inalámbricas corporativas.
- Permiten más libertad en el movimiento de los nodos conectados, algo que puede convertirse en un verdadero problema en las redes cableadas.
- Permite crear una red en áreas complicadas donde, por ejemplo, resulta dificultoso o muy caro conectar cables.

Desventajas

- Calidad de Servicio: La velocidad que posee la red inalámbrica no supera la cableada, ya que esta puede llegar a los 10 Mbps, frente a 100 Mbps que puede alcanzar la cableada. Hay que tomar en cuenta la tasa de error debida a las interferencias.
- Costo: En algunos casos, puede ser más barato cablear una casa/oficina que colocar un servicio de red inalámbrica.
- La señal inalámbrica puede verse afectada e incluso interrumpida por objetos, árboles, paredes, espejos, entre otros.

La velocidad máxima de transmisión inalámbrica de la tecnología 802.11b es de 11 Mbps. Pero la velocidad típica es solo la mitad: entre 1.5 y 5 Mbps dependiendo de si se transmiten muchos archivos pequeños o unos pocos archivos grandes. La velocidad máxima de la tecnología 802.11g es de 54 Mbps. Pero la velocidad típica de esta última tecnología es solo unas 3 veces más rápida que la de 802.11b: entre 5 y 15 Mbps. Resumiendo, las velocidades típicas de los diferentes tipos de red son:

Estándar	V. Máxima	V. Practica	Frecuencia	Ancho Banda	Alcance
802.11	2Mbit/s	1Mbit/s	2.4Ghz	22MHz	330 metros
802.11a(WiFi5)	54Mbit/s	22Mbit/s	5.4Ghz	20MHz	390 metros
802.11b	11Mbit/s	6Mbit/s	2.4Ghz	22MHz	460 metros
802.11g	54Mbit/s	22Mbit/s	2.4Ghz	20MHz	460 metros
802.11n	600Mbit/s	100Mbit/s	2.4Ghz y 5.4Ghz	20 y 40MHz	820 metros
802.11ac	6.93Gbps	100Mbit/s	5.4Ghz	80 o hasta 160MHz	Poco alcance, pero sin interferencias
802.11ad	7.13Gbit/s	Hasta 6Gbit/s	60Ghz	2MHz	300 metros
802.11ah	35.6Mbps	26.7Mbps	0.9Ghz	2MHz	1000 metros
802.11ax(WiFi6)	9.6Gbps	6.9Gbps	2.4Ghz y 5.4Ghz	20MHz	1000 metros

Como puedes ver, los principales factores que influyen en la calidad de una conexión WiFi, son la frecuencia, el ancho de banda y el alcance total. Considerando que, todo esto junto con la velocidad máxima y la velocidad práctica, se congregan en lo que es cada versión de este tipo de conexión a la red. Además, la conexión se degradará inexorablemente con la cantidad de dispositivos conectados y su consumo de datos.

Velocidad de los Proveedores cuando se contrata el servicio de internet, la velocidad de interconexión del mismo depende de cuánto sea el cobro, pero en la mayoría de los casos se tendrá una velocidad de descarga mayor a la velocidad de carga y nuestra red será una intranet (dirección de IP dinámica) compartida con otros miles de usuarios abonados al servicio de internet del proveedor. También es posible contratar una interconexión con velocidades homogéneas para carga y descarga dedicada, el costo del mismo se puede hasta triplicar con respecto a uno no homogéneo.

En caso de requerir una dirección de internet homologada o pública, el costo de contratar el servicio aumenta considerablemente, pero de esta forma nuestros equipos son visibles en el Internet (esto conlleva un aumento de riesgos al estar nuestros equipos más vulnerables a ataques informáticos).

Información de mi Computadora Si lo que deseamos es un listado detallado del Hardware de nuestro equipo de cómputo, entonces podemos usar cualquiera de estos comandos (que previamente deberemos instalar):

```
$ lscpu
# lshw
# dmidecode
# hwinfo
```

22.2 ¿Que es Computación Paralela?

En el sentido más simple, la computación paralela es el uso simultáneo de múltiples recursos computacionales para resolver un problema computacional:

- Un problema se divide en partes discretas que se pueden resolver simultáneamente

- Cada parte se descompone en una serie de instrucciones
- Las instrucciones de cada parte se ejecutan simultáneamente en diferentes procesadores
- Se emplea un mecanismo global de control/coordinación

¿Por qué se hace programación paralela? El hecho de que la programación paralela sea un paradigma da cuenta de que existe una razón por la cual no ha dejado de ser necesaria o no ha sido totalmente automatizable, igualmente hay otras razones interesantes detrás para entender la existencia, actualidad y contemporaneidad de la programación paralela:

- Ley de Moore: Esta ley propuesta por Gordon E. Moore en 1965 dice resumidamente que el número de transistores en un Chip determinado se doblaría cada dos años. Esto quiere decir un aumento del rendimiento en los procesadores del alrededor del 50%, esto se traduce en escalar la velocidad de reloj de los procesadores, pero esta ley no es fidedigna desde el 2002 dónde solo ha habido un 20%, lo cuál sigue siendo un aumento considerable, sin embargo, no sería suficiente para que todos los avances en computación que se han logrado hasta el día y las necesidades de procesamiento en crecimiento exponencial puedan satisfacerse totalmente.
- Overclocking infinito: El Overclocking tiene un límite a pesar de que existiera una refrigeración perpetúa y adecuada del procesador. Esto es debido a las corrientes parásitas que impiden una velocidad teóricamente infinita a la cual los circuitos pueden cambiar entre estados, o de hecho sus transistores.
- Automatización del paralelismo: Se dice en este paradigma que el éxito es inversamente proporcional al número de cores precisamente porque existen complejidades en el corazón del paralelismo que implican cosas que todavía no se pueden predecir ni con inteligencia artificial, se menciona cuales son las posibles estrategias para atacar un problema de forma paralela, esto da cuenta de que existe una forma prácticamente determinada de abordarlos pero no de automatizarlos, a pesar de que sí existan algunas partes que son automatizables en el proceso.

- Solución en el Hardware: Un diseño adecuado del Hardware permitiría que la paralelización siempre estuviera presente con respecto a los procesadores que se están usando de tal modo que alguno los problemas que son inherentes al paradigma pudieran evitarse. Esto ha resultado imposible hasta la fecha, de hecho, solo diseñar solamente algo tan efectivo y tradicional como se ha hecho en programación secuencial es algo que no existe hasta ahora. Existen algunas aproximaciones como OpenMP y programación por hilos de las que hablaremos más adelante.

Ventajas

- Resuelve problemas que no se podrían realizar en una sola CPU
- Resuelve problemas que no se pueden resolver en un tiempo razonable
- Permite ejecutar problemas de un orden y complejidad mayor
- Permite ejecutar código de manera más rápida (aceleración)
- Permite ejecutar en general más problemas
- Obtención de resultados en menos tiempo
- Permite la ejecución de varias instrucciones en simultáneo
- Permite dividir una tarea en partes independientes

Desventajas

- Mayor consumo de energía
- Mayor dificultad a la hora de escribir programas
- Dificultad para lograr una buena sincronización y comunicación entre las tareas
- Retardos ocasionados por comunicación entre tareas
- Número de componentes usados es directamente proporcional a los fallos potenciales
- Condiciones de carrera

- Múltiples procesos se encuentran en condición de carrera si el resultado de los mismos depende del orden de su llegada
- Si los procesos que están en condición de carrera no son correctamente sincronizados, puede producirse una corrupción de datos

Paralelismo de Grano Fino, Grano Grueso y Paralelismo Vergonzoso las aplicaciones a menudo se clasifican según la frecuencia con que sus subtarear se sincronizan o comunican entre sí. Una aplicación muestra un paralelismo de grano fino si sus subtarear deben comunicarse muchas veces por segundo, se considera paralelismo de grano grueso si no se comunican muchas veces por segundo, y es vergonzosamente paralelo si nunca o casi nunca se tienen que comunicar. Aplicaciones vergonzosamente paralelas son consideradas las más fáciles de paralelizar.

El tipo de problemas complejos que aborda el cómputo de alto rendimiento (HPC por High-Performance Computing) no se pueden resolver con una computadora de escritorio y tienen una escala determinada: toma mucho tiempo resolverlos, se necesita una gran cantidad de memoria (RAM), se tienen que realizar muchísimos experimentos parecidos y hay restricciones concretas de tiempo para encontrar resultados. La mayoría de estas temáticas están relacionadas con la innovación en la industria y su aplicación en áreas como energía, medio ambiente, biotecnología, medicina, ingeniería, etc.

A su vez, la evolución del procesador la cantidad de transistores sigue creciendo pero la velocidad de los núcleos individuales (core) casi no aumenta. Podríamos comprar un procesador de 128 núcleos pero el desafío de aprovechar la potencia de un procesador es cómo distribuimos el tiempo de ejecución de una misma tarea computacional.

De este modo, cada tarea computacional se divide en dos partes: serial y paralelizable. Como todas las aplicaciones tienen una parte secuencial, el tiempo de cómputo de la aplicación paralela está acotado por esa sección serial (Ley de Ahmdahl). Esto implica dos cuestiones fundamentales:

1. Es necesario reducir lo más posible el tiempo de ejecución serial
2. En una tarea computacional existe un máximo de procesadores que conviene poner a trabajar en forma paralela, más allá de ese punto no se justifica seguir paralelizando la tarea.

¿Qué debe tener en cuenta un centro de HPC para aprovechar al máximo los recursos de cómputo? Cada centro de HPC tiene que configurar, optimizar aplicaciones y sistemas operativos. Allí el monitoreo es clave, con la escala de las infraestructuras también surgen aplicaciones con problemas de memoria (cuello de botella) y la necesidad de utilizar mejores redes (PARAMNet-3, Infiniband, GigE). Tener poder de cómputo no es barato y solamente con el Hardware no alcanza. El equipo de operaciones de HPC necesita un alto grado de conocimiento y de disciplina para encarar la tarea.

Con el uso del cómputo de alto desempeño es posible por ejemplo reducir el tiempo que nos lleva descubrir nuevos medicamentos de años a meses. Una computadora de alto rendimiento es capaz de resolver este y otros tipos de problemas científicos avanzados mediante simulaciones, modelos y análisis. Estos sistemas abren las puertas de la “Cuarta Revolución Industrial”, ya que ayudan a resolver muchas de las problemáticas más importantes del mundo. Los sistemas HPC ya se utilizan para las siguientes tareas:

- Descubrir nuevos componentes de drogas y probar los conocidos para combatir diferentes tipos de cáncer y otras enfermedades
- Simular dinámicas moleculares para crear nuevos materiales, como tejidos balísticos
- Pronosticar cambios climáticos considerables para mejorar la preparación de las comunidades afectadas

Las supercomputadoras representan lo último en sistemas HPC. La definición de supercomputadora depende de diferentes estándares que van cambiando a medida que las capacidades evolucionan. Un solo clúster de supercomputadoras puede incluir decenas de miles de procesadores, y los sistemas más caros y potentes del mundo pueden costar más de US\$ 100 millones de dólares.

Casos de Uso Emergentes a medida que las tecnologías mejoraron, la computación de alto rendimiento comenzó a implementarse en una gama de capacidades más amplia. Ahora contamos con más potencia de procesamiento y memoria que nunca para solucionar problemas más complejos.

- Aprendizaje automático: como subconjunto de la inteligencia artificial (IA), el aprendizaje automático (AA) se refiere a un sistema que tiene la capacidad de aprender de forma activa por sí mismo, a diferencia de recibir, de forma pasiva, instrucciones para ejecutar. Los sistemas HPC pueden utilizarse en aplicaciones de AA altamente avanzadas donde se analizan grandes cantidades de datos, por ejemplo, investigación del cáncer para detectar el melanoma en imágenes.
- Análisis de grandes conjuntos de datos: se recurre a la comparación rápida y a la correlación de grandes conjuntos de datos para complementar investigaciones y resolver problemas académicos, científicos, financieros, comerciales, gubernamentales, de salud y de seguridad cibernética. Este trabajo requiere un rendimiento masivo y capacidades de cómputo de una potencia enorme. Con un estimado de 50 petabytes de datos al año generados por las misiones, la NASA se apoya en la supercomputación para analizar observaciones y realizar simulaciones con grandes conjuntos de información.
- Modelado avanzado y simulación: al no tener que realizar un montaje físico en las primeras etapas del proceso, el modelado avanzado y la simulación permiten que las empresas ahorren tiempo, materiales y costos de contratación de personal para lanzar sus productos al mercado con mayor rapidez. El modelado y la simulación en HPC se aplican en el descubrimiento y la prueba de fármacos, diseños automotrices y aeroespaciales, pronóstico de sistemas climáticos o meteorológicos, y aplicaciones energéticas.

Cómo Funciona la Computación de Alto Rendimiento Existen dos métodos principales para procesar la información en HPC:

- Procesamiento en serie: es el que realizan las unidades de procesamiento central (CPU). Cada núcleo de CPU, por lo general, realiza solo una tarea a la vez. Las CPU son fundamentales para ejecutar diferentes funciones, como sistemas operativos y aplicaciones básicas (por ej., procesamiento de textos, productividad en la oficina).
- Procesamiento en paralelo: es el que se puede realizar mediante varias CPU o unidades de procesamiento de gráficos (GPU). Las GPU, diseñadas originalmente para gráficos independientes, son capaces de rea-

lizar diferentes operaciones aritméticas por medio de una matriz de datos (como píxeles de pantalla) de forma simultánea. La capacidad para trabajar en varios planos de datos al mismo tiempo hace que las GPU sean la elección natural para el procesamiento en paralelo en tareas de aplicaciones de aprendizaje automático (AA), como el reconocimiento de objetos en videos.

Para superar los límites de la supercomputación, se necesitan diferentes arquitecturas de sistemas. Para que el procesamiento en paralelo pueda llevarse a cabo, la mayoría de los sistemas HPC integran varios procesadores y módulos de memoria a través de interconexiones con un ancho de banda enorme. Ciertos sistemas HPC combinan varias CPU y GPU, lo que se conoce como computación heterogénea.

La potencia de procesamiento de las computadoras se mide en unidades llamadas “FLOPS”⁷¹ (operaciones de punto flotante por segundo). A principios de 2019, la supercomputadora más potente que existe alcanzó los 143,5 peta-Flops (143×10^{15}). Este tipo de supercomputadora se llama equipo de petaescala y puede realizar más de mil billones de FLOPS. Por su parte, una computadora de escritorio para juegos de alta gama es más de un millón de veces más lenta y llega apenas a los 200 giga-FLOPS (1×10^9).

Gracias a los avances tanto en procesamiento como rendimiento, en el año 2020 fuimos testigos de un nuevo salto en la era de la supercomputación: la exaescala, que es casi 1000 veces más rápida que la petaescala. Esto significa que un sistema de exaescala realiza 10^{18} (o mil millones por mil millones) operaciones por segundo. Necesitaríamos 5 millones de computadoras de escritorio para alcanzar el nivel de rendimiento de procesamiento en supercomputación de 1 exa-Flops, suponiendo que cada computadora de escritorio es capaz de realizar 200 giga-FLOPS.

Arquitecturas de Software y Hardware En las dos siguientes secciones se explican en detalle las dos clasificaciones de computadoras más conocidas en la actualidad. La primera clasificación, es la clasificación clásica de Flynn

⁷¹ “FLOPS” describe una velocidad de procesamiento teórica: para hacer posible esa velocidad es necesario enviar datos a los procesadores de forma continua. Por lo tanto, el procesamiento de los datos se debe tener en cuenta en el diseño del sistema. La memoria del sistema, junto con las interconexiones que unen los nodos de procesamiento entre sí, impactan en la rapidez con la que los datos llegan a los procesadores.

en dónde se tienen en cuenta sistemas con uno o varios procesadores, la segunda clasificación es moderna en la que sólo se tienen en cuenta los sistemas con más de un procesador.

El objetivo es presentar de una forma clara los tipos de clasificación que existen en la actualidad desde el punto de vista de distintos autores, así como cuáles son las ventajas e inconvenientes que cada uno ostenta, ya que es común que al resolver un problema particular se usen una o más arquitecturas de Hardware interconectadas generalmente por red.

22.3 Clasificación Clásica de Flynn

La clasificación clásica de arquitecturas de computadoras que hace alusión a sistemas con uno o varios procesadores fue realizada por Michael J. Flynn y la publicó por primera vez en 1966 y por segunda vez en 1970.

Esta taxonomía se basa en el flujo que siguen los datos dentro de la máquina y de las instrucciones sobre esos datos. Se define como flujo de instrucciones al conjunto de instrucciones secuenciales que son ejecutadas por un único procesador y como flujo de datos al flujo secuencial de datos requeridos por el flujo de instrucciones. Con estas consideraciones, Flynn clasifica los sistemas en cuatro categorías:

Single Instruction stream, Single Data stream (SISD) Los sistemas Monoprocesador de este tipo se caracterizan por tener un único flujo de instrucciones sobre un único flujo de datos, es decir, se ejecuta una instrucción detrás de otra. Este es el concepto de arquitectura serie de Von Neumann donde, en cualquier momento, sólo se ejecuta una única instrucción, un ejemplo de estos sistemas son las máquinas secuenciales convencionales.

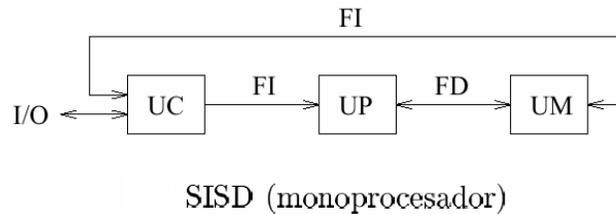
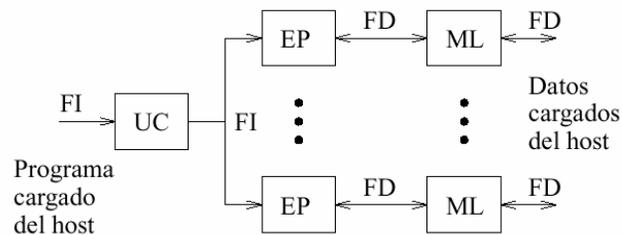


Figura 30: Ejemplo de máquina SISD

Single Instruction stream, Multiple Data stream (SIMD) Estos sistemas de procesador Matricial tienen un único flujo de instrucciones que operan sobre múltiples flujos de datos. Ejemplos de estos sistemas los tenemos en las máquinas vectoriales con Hardware escalar y vectorial.

El procesamiento es síncrono, la ejecución de las instrucciones sigue siendo secuencial como en el caso anterior, todos los elementos realizan una misma instrucción pero sobre una gran cantidad de datos. Por este motivo existirá concurrencia de operación, es decir, esta clasificación es el origen de la máquina paralela.

El funcionamiento de este tipo de sistemas es el siguiente. La unidad de control manda una misma instrucción a todas las unidades de proceso (ALUs). Las unidades de proceso operan sobre datos diferentes pero con la misma instrucción recibida.



SIMD (procesador matricial)

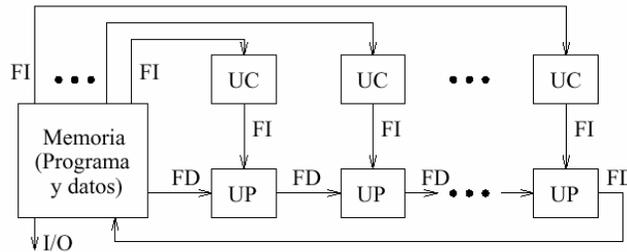
Figura 31: Ejemplo de máquina SIMD

Existen dos alternativas distintas que aparecen después de realizarse esta clasificación:

- Arquitectura Vectorial con segmentación, una CPU única particionada en unidades funcionales independientes trabajando sobre flujos de datos concretos.
- Arquitectura Matricial (matriz de procesadores), varias ALUs idénticas a las que el procesador da instrucciones, asigna una única instrucción pero trabajando sobre diferentes partes del programa.

Multiple Instruction stream, Single Data stream (MISD) Sistemas con múltiples instrucciones Array Sistólico que operan sobre un único flujo de datos. Este tipo de sistemas no ha tenido implementación hasta hace poco tiempo. Los sistemas MISD se contemplan de dos maneras distintas:

- Varias instrucciones operando simultáneamente sobre un único dato.
- Varias instrucciones operando sobre un dato que se va convirtiendo en un resultado que será la entrada para la siguiente etapa. Se trabaja de forma segmentada, todas las unidades de proceso pueden trabajar de forma concurrente.

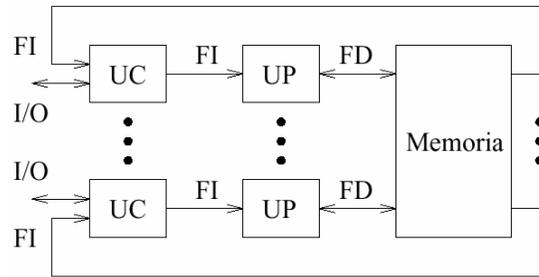


MISD (array sistólico)

Figura 32: Ejemplo de máquina MISD

Multiple Instruction stream, Multiple Data stream (MIMD) Sistemas con un flujo de múltiples instrucciones Multiprocesador que operan sobre múltiples datos. Estos sistemas empezaron a utilizarse antes de la década de los 80s. Son sistemas con memoria compartida que permiten ejecutar varios procesos simultáneamente (sistema multiprocesador).

Cuando las unidades de proceso reciben datos de una memoria no compartida estos sistemas reciben el nombre de MULTIPLE SISD (MSISD). En arquitecturas con varias unidades de control (MISD Y MIMD), existe otro nivel superior con una unidad de control que se encarga de controlar todas las unidades de control del sistema -ejemplo de estos sistemas son las máquinas paralelas actuales-.



MIMD (multiprocesador)

Figura 33: Ejemplo de máquina MIMD

22.4 Categorías de Computadoras Paralelas

Clasificación moderna que hace alusión única y exclusivamente a los sistemas que tienen más de un procesador (i.e máquinas paralelas). Existen dos tipos de sistemas teniendo en cuenta su acoplamiento:

- Los sistemas fuertemente acoplados son aquellos en los que los procesadores dependen unos de otros.
- Los sistemas débilmente acoplados son aquellos en los que existe poca interacción entre los diferentes procesadores que forman el sistema.

Atendiendo a esta y a otras características, la clasificación moderna divide a los sistemas en dos tipos: Sistemas multiprocesador (fuertemente acoplados) y sistemas multicomputadoras (débilmente acoplados).

22.4.1 Equipo Paralelo de Memoria Compartida

Un multiprocesador puede verse como una computadora paralela compuesta por varios procesadores interconectados que comparten un mismo sistema de memoria.

Los sistemas multiprocesadores son arquitecturas MIMD con memoria compartida. Tienen un único espacio de direcciones para todos los procesadores y los mecanismos de comunicación se basan en el paso de mensajes desde el punto de vista del programador.

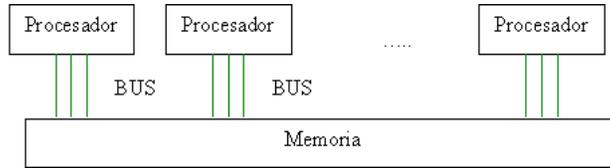


Figura 34: Arquitectura de una computadora paralela con memoria compartida

Dado que los multiprocesadores comparten diferentes módulos de memoria, pueden acceder a un mismo módulo varios procesadores, a los multiprocesadores también se les llama sistemas de memoria compartida.

Para hacer uso de la memoria compartida por más de un procesador, se requiere hacer uso de técnicas de semáforos que mantienen la integridad de la memoria; esta arquitectura no puede crecer mucho en el número de procesadores interconectados por la saturación rápida del bus o del medio de interconexión.

Dependiendo de la forma en que los procesadores comparten la memoria, se clasifican en sistemas multiprocesador UMA, NUMA, COMA y Pipeline, que explicamos a continuación:

Uniform Memory Access (UMA) Sistema multiprocesador con acceso uniforme a memoria. La memoria física es uniformemente compartida por todos los procesadores, esto quiere decir que todos los procesadores tienen el mismo tiempo de acceso a todas las palabras de la memoria. Cada procesador tiene su propia Caché privada y también se comparten los periféricos.

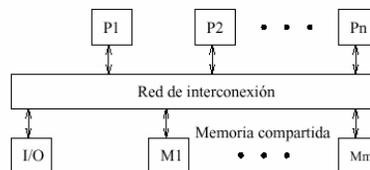


Figura 35: Acceso Uniforme a la memoria UMA

Los multiprocesadores son sistemas fuertemente acoplados (tightly-coupled), dado el alto grado de compartición de los recursos (Hardware o Software) y

el alto nivel de interacción entre procesadores, lo que hace que un procesador depende de lo que hace otro. El sistema de interconexión debe ser rápido y puede ser de uno de los siguientes tipos: bus común, red Crossbar⁷² y red Multietapa. Este modelo es conveniente para aplicaciones de propósito general y de tiempo compartido por varios usuarios, existen dos categorías de sistemas UMA.

- Sistema Simétrico: cuando todos los procesadores tienen el mismo tiempo de acceso a todos los componentes del sistema (incluidos los periféricos), reciben el nombre de sistemas multiprocesador simétrico. Los procesadores tienen el mismo dominio (prioridad) sobre los periféricos y cada procesador tiene la misma capacidad para procesar.
- Sistema Asimétrico: son sistemas con procesador principal y procesadores subordinados, en donde sólo el primero puede ejecutar aplicaciones y donde el tiempo de acceso para diferentes procesadores no es el mismo. Los procesadores subordinados (Attached) ejecutan código usuario bajo la supervisión del principal, por lo tanto cuando una aplicación es ejecutada en un procesador principal dispondrá de una cierta prioridad.

Non Uniform Memory Access (NUMA) Un sistema multiprocesador NUMA es un sistema de memoria compartida donde el tiempo de acceso varía según donde se encuentre localizado el acceso.

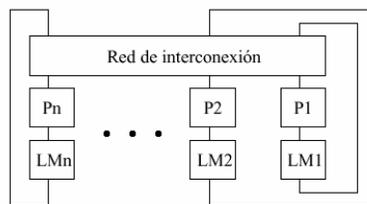


Figura 36: Acceso no uniforme a la memoria NUMA

El acceso a memoria, por tanto, no es uniforme para diferentes procesadores, existen memorias locales asociadas a cada procesador y estos pueden

⁷²Red reconfigurable que permite la conexión de cada entrada con cualquiera de las salidas, es decir, permite cualquier permutación.

acceder a datos de su memoria local de una manera más rápida que a las memorias de otros procesadores, debido a que primero debe aceptarse dicho acceso por el procesador del que depende el módulo de memoria local.

Todas las memorias locales conforman la memoria global compartida y físicamente distribuida y accesible por todos los procesadores.

Cache Only Memory Access (COMA) Los sistemas COMA son un caso especial de los sistemas NUMA. Este tipo de sistemas no ha tenido mucha trascendencia, al igual que los sistemas SIMD.

Las memorias distribuidas son memorias Cachés, por este motivo es un sistema muy restringido en cuanto a la capacidad de memoria global. No hay jerarquía de memoria en cada módulo procesador. Todas las Cachés forman un mismo espacio global de direcciones. El acceso a las Cachés remotas se realiza a través de los directorios distribuidos de las Cachés.

Dependiendo de la red de interconexión utilizada, se pueden utilizar jerarquías en los directorios para ayudar a la localización de copias de bloques de Caché.

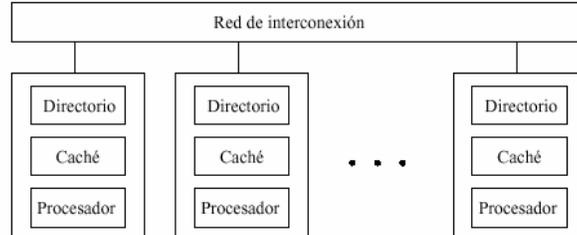


Figura 37: Ejemplo de COMA

Procesador Vectorial Pipeline En la actualidad es común encontrar en un solo procesador los denominados Pipeline o Procesador Vectorial Pipeline del tipo MISD. En estos procesadores los vectores fluyen a través de las unidades aritméticas Pipeline.

Las unidades constan de una cascada de etapas de procesamiento compuestas de circuitos que efectúan operaciones aritméticas o lógicas sobre el flujo de datos que pasan a través de ellas, las etapas están separadas por

registros de alta velocidad usados para guardar resultados intermedios. Así la información que fluye entre las etapas adyacentes está bajo el control de un reloj que se aplica a todos los registros simultáneamente.

22.4.2 Equipo Paralelo de Memoria Distribuida

Los sistemas multicomputadoras (los más comunes son los Clusters) se pueden ver como una computadora paralela en el cual cada procesador tiene su propia memoria local. En estos sistemas la memoria se encuentra distribuida y no compartida como en los sistemas multiprocesador. Los procesadores se comunican a través de paso de mensajes, ya que éstos sólo tienen acceso directo a su memoria local y no a las memorias del resto de los procesadores.

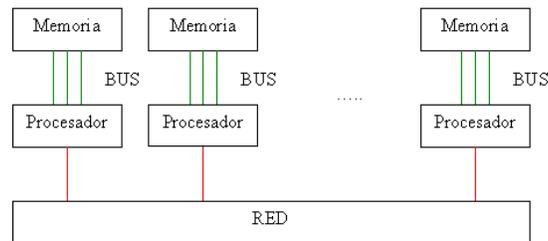


Figura 38: Arquitectura de una computadora paralela con memoria distribuida

La transferencia de los datos se realiza a través de la red de interconexión que conecta un subconjunto de procesadores con otro subconjunto. La transferencia de unos procesadores a otros se realiza por múltiples transferencias entre procesadores conectados dependiendo del establecimiento de dicha red.

Dado que la memoria está distribuida entre los diferentes elementos de proceso, estos sistemas reciben el nombre de distribuidos. Por otra parte, estos sistemas son débilmente acoplados, ya que los módulos funcionan de forma casi independiente unos de otros. Este tipo de memoria distribuida es de acceso lento por ser peticiones a través de la red, pero es una forma muy efectiva de tener acceso a un gran volumen de memoria.

22.4.3 Equipo Paralelo de Memoria Compartida-Distribuida

La tendencia actual en las máquinas paralelas es de aprovechar las facilidades de programación que ofrecen los ambientes de memoria compartida y la escalabilidad de los ambientes de memoria distribuida. En este modelo se conectan entre sí módulos de multiprocesadores, pero se mantiene la visión global de la memoria a pesar de que es distribuida.

Clusters El desarrollo de sistemas operativos y compiladores del dominio público (Linux y Software GNU), estándares para interfaz de paso de mensajes (Message Passing Interface MPI), conexión universal a periféricos (Peripheral Component Interconnect PCI), entre otros, han hecho posible tomar ventaja de los recursos económicos computacionales de producción masiva (procesadores, discos, redes).

La principal desventaja que presenta a los proveedores de multicomputadoras es que deben satisfacer una amplia gama de usuarios, es decir, deben ser generales. Esto aumenta los costos de diseños y producción de equipos, así como los costos de desarrollo de Software que va con ellos: sistema operativo, compiladores y aplicaciones. Todos estos costos deben ser añadidos cuando se hace una venta. Por supuesto alguien que sólo necesita procesadores y un mecanismo de pase de mensajes no debería pagar por todos estos añadidos que nunca usará. Estos usuarios son los que están impulsando el uso de Clusters principalmente de computadoras personales (PC), cuya arquitectura se muestra a continuación:

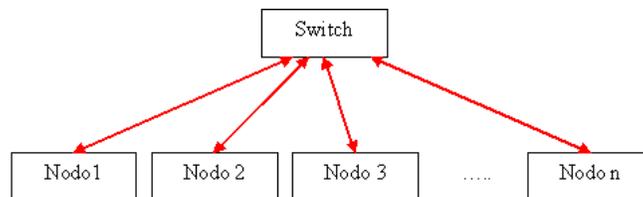


Figura 39: Arquitectura de un cluster

Los Cluster⁷³ se pueden clasificar en dos tipos según sus características físicas:

⁷³Existe el Ranking de las 500 supercomputadoras más poderosas del mundo (esta se actualiza cada seis meses en junio y noviembre) y puede ser consultada en:

<https://top500.org>

- Cluster homogéneo: si todos los procesadores y/o nodos participantes en el equipo paralelo son iguales en capacidad de cómputo -es permitido variar la cantidad de memoria o disco duro en cada procesador-
- Cluster heterogéneo: es aquel en que al menos uno de los procesadores y/o nodos participantes en el equipo paralelo son de distinta capacidad de cómputo

El Clustering no presenta dependencias a nivel de Hardware (no todos los equipos necesitan el mismo Hardware) ni a nivel de Software (no necesitan el mismo sistema operativo). Este tipo de sistemas disponen de una interfaz que permite dirigir el comportamiento de los Clusters. Dicha interfaz es la encargada de la interacción con usuarios y procesos, realizando la división de la carga entre los diversos servidores que compongan el Cluster.

Los Clusters pueden formarse de diversos equipos; los más comunes son los de computadoras personales, pero es creciente el uso de computadoras multiprocesador de más de un procesador de memoria compartida interconectados por red con los demás nodos del mismo tipo, incluso el uso de computadoras multiprocesador de procesadores vectoriales Pipeline. Los Clusters armados con la configuración anterior tienen grandes ventajas para procesamiento paralelo:

- La reciente explosión en redes implica que la mayoría de los componentes necesarios para construir un Cluster son vendidos en altos volúmenes y por lo tanto son económicos. Ahorros adicionales se pueden obtener debido a que sólo se necesitará una tarjeta de vídeo, un monitor y un teclado por Cluster. El mercado de los multiprocesadores es más reducido y más costoso
- El reemplazar un componente defectuoso en un Cluster es relativamente trivial comparado con hacerlo en un multiprocesador, permitiendo una mayor disponibilidad de Clusters cuidadosamente diseñados

Desventajas del uso de Clusters de computadoras personales para procesamiento paralelo:

- Con raras excepciones, los equipos de redes generales producidos masivamente no están diseñados para procesamiento paralelo

y típicamente su latencia es alta y los anchos de banda pequeños comparados con multiprocesadores. Dado que los Clusters explotan tecnología que sea económica, los enlaces en el sistema no son veloces implicando que la comunicación entre componentes debe pasar por un proceso de protocolos de negociación lentos, incrementando seriamente la latencia. En muchos y en el mejor de los casos (debido a costos) se recurre a una red tipo Fast Ethernet restringimiento la escalabilidad del Cluster

- Hay poco soporte de Software para manejar un Cluster como un sistema integrado
- Los procesadores no son tan eficientes como los procesadores usados en los multiprocesadores para manejar múltiples usuarios y/o procesos. Esto hace que el rendimiento de los Clusters se degrade con relativamente pocos usuarios y/o procesos
- Muchas aplicaciones importantes disponibles en multiprocesadores y optimizadas para ciertas arquitecturas, no lo están en Clusters

Sin lugar a duda los Clusters presentan una alternativa importante para varios problemas particulares, no sólo por su economía, si no también porque pueden ser diseñados y ajustados para ciertas aplicaciones. Las aplicaciones que pueden sacar provecho de Clusters son en donde el grado de comunicación entre procesos es de bajo a medio.

Tipos de Cluster Básicamente existen tres tipo de Clusters, cada uno de ellos ofrece ventajas y desventajas, el tipo más adecuado para el cómputo científico es del de alto-rendimiento, pero existen aplicaciones científicas que pueden usar más de un tipo al mismo tiempo.

- Alta disponibilidad (Fail-over o High-Availability): este tipo de Cluster está diseñado para mantener uno o varios servicios disponibles incluso a costa de rendimiento, ya que su función principal es que el servicio jamás tenga interrupciones como por ejemplo un servicio de bases de datos de transacciones bancarias
- Alto rendimiento (HPC o High Performance Computing): este tipo de Cluster esta diseñado para obtener el máximo rendimiento de la aplicación utilizada incluso a costa de la disponibilidad del

sistema, es decir el Cluster puede sufrir caídas, este tipo de configuración está orientada a procesos que requieran mucha capacidad de cálculo.

- **Balanceo de Carga (Load-Balancing):** este tipo de Cluster está diseñado para balancear la carga de trabajo entre varios servidores, lo que permite tener, por ejemplo, un servicio de cálculo intensivo multiusuarios que detecte tiempos muertos del proceso de un usuario para ejecutar en dichos tiempos procesos de otros usuarios.

Grids Son cúmulos (grupo de Clusters) de arquitecturas en paralelo interconectados por red, los cuales distribuyen tareas entre los Clusters que lo forman, estos pueden ser homogéneos o heterogéneos en cuanto a los nodos componentes del cúmulo. Este tipo de arquitecturas trata de distribuir cargas de trabajo acorde a las características internas de cada Cluster y las necesidades propias de cada problema, esto se hace a dos niveles, una en la parte de programación en conjunto con el balance de cargas y otra en la parte de Hardware que tiene que ver con las características de cada arquitectura que conforman el cúmulo.

Balance de Carga A la hora de diseñar un sistema paralelo con compartición de recursos, es necesario considerar cómo balancear la carga de trabajo. Se entiende este concepto, como la técnica usada para dividir el trabajo a compartir entre varios procesos, equipos de cómputo, u otros recursos. Está muy relacionada con los sistemas multiproceso, que trabajan o pueden trabajar con más de una unidad para llevar a cabo su funcionalidad.

Para evitar los cuellos de botella, el balance de la carga de trabajo se reparte de forma equitativa a través de un algoritmo que estudia las peticiones del sistema y las redirecciona a la mejor opción.

Balance de Carga por Hardware presenta las siguientes características:

- A partir de un algoritmo de planificación de procesos -Round Robin, LRU-, examina las peticiones entrantes y selecciona el más apropiado entre los distintos elementos del sistema

- La selección del siguiente libre del sistema está basada en el algoritmo de sustitución y es aleatoria
- La sesión debe de ser mantenida por el desarrollador
- Al ser un proceso de Hardware, es muy rápido

Balance de Carga por Software presenta las siguientes características:

- Examinan la solicitud para garantizar que se puede satisfacer la demanda de los usuarios
- Distintas peticiones del mismo usuario son servidas simultáneamente o secuencialmente según se definan
- Más lentos que los balanceadores de Hardware
- Normalmente son soluciones baratas

Escalabilidad Se entiende por escalabilidad a la capacidad de adaptación y respuesta de un sistema con respecto al rendimiento del mismo a medida que aumentan de forma significativa la carga computacional del mismo. Aunque parezca un concepto claro, la escalabilidad de un sistema es un aspecto complejo e importante del diseño de sistemas paralelos.

La escalabilidad está íntimamente ligada al diseño de sistemas paralelos, influye en el rendimiento de forma significativa. Si una aplicación está bien diseñada, la escalabilidad no constituye un problema. Analizando la escalabilidad, se deduce de la implementación y del diseño general del sistema. No es atributo del sistema configurable.

La escalabilidad supone un factor crítico en el crecimiento de un sistema paralelo. Si un sistema tiene como objetivo crecer la carga computacional -en el número de usuarios o procesos- manteniendo su rendimiento actual, tiene que evaluar dos posibles opciones:

- Con un Hardware de mayor potencia o
- Con una mejor combinación de Hardware y Software

Se pueden distinguir dos tipos de escalabilidad, vertical y horizontal:

- El escalar verticalmente o escalar hacia arriba, significa el añadir más recursos a un solo nodo en particular dentro de un sistema, tal como el añadir memoria o un disco duro más rápido a una computadora.
- La escalabilidad horizontal, significa agregar más nodos a un sistema, tal como añadir una computadora nueva a un programa de aplicación para espejo.

Escalabilidad Vertical El escalar hacia arriba de un sistema viene a significar una migración de todo el sistema a un nuevo Hardware que es más potente y eficaz que el actual. Una vez se ha configurado el sistema futuro, se realizan una serie de validaciones y copias de seguridad y se pone en funcionamiento. Las aplicaciones que estén funcionando bajo la arquitectura Hardware antigua no sufren con la migración, el impacto en el código es mínimo.

Este modelo de escalabilidad vertical tiene un aspecto negativo. Al aumentar la potencia en base a ampliaciones de Hardware, llegará un momento que existirá algún tipo de limitación de Hardware. Además a medida que se invierte en Hardware de muy altas prestaciones, los costos se disparan tanto de forma temporal -ya que si se ha llegado al umbral máximo, hay componentes de Hardware que tardan mucho tiempo en ampliar su potencia de forma significativa- como económicos. Sin embargo a nivel estructural no supone ninguna modificación reseñable, lo que la convierte en una buena opción si los costos anteriores son asumibles.

Escalabilidad Horizontal La escalabilidad horizontal consiste en potenciar el rendimiento del sistema paralelo desde un aspecto de mejora global, a diferencia de aumentar la potencia de una única parte del mismo. Este tipo de escalabilidad se basa en la modularidad de su funcionalidad, por ello suele estar conformado por una agrupación de equipos que dan soporte a la funcionalidad completa. Normalmente, en una escalabilidad horizontal se añaden equipos para dar más potencia a la red de trabajo.

Con un entorno de este tipo, es lógico pensar que la potencia de procesamiento es directamente proporcional al número de equipos en la red. El total de la potencia de procesamiento es la suma de la velocidad física de cada equipo transferida por la partición de aplicaciones y datos extendida a través de los nodos.

Si se aplica un modelo de escalabilidad basado en la horizontalidad, no existen limitaciones de crecimiento a priori. Como principal defecto, este modelo de escalabilidad supone una gran modificación en el diseño, lo que conlleva a una gran trabajo de diseño y reimplantación. Si la lógica se ha concebido para un único servidor, es probable que se tenga que estructurar el modelo arquitectónico para soportar este modelo de escalabilidad.

El encargado de cómo realizar el modelo de partición de datos en los diferentes equipos es el desarrollador. Existen dependencias en el acceso a la aplicación. Es conveniente, realizar una análisis de actividad de los usuarios para ir ajustando el funcionamiento del sistema. Con este modelo de escalabilidad, se dispone de un sistema al que se pueden agregar recursos de manera casi infinita y adaptable al crecimiento de cargas de trabajo y nuevos usuarios.

La escalabilidad cuenta como factor crítico en el crecimiento de usuarios. Es mucho más sencillo diseñar un sistema con un número constante de usuarios -por muy alto que sea este- que diseñar un sistema con un número creciente y variable de usuarios. El crecimiento relativo de los números es mucho más importante que los números absolutos.

22.4.4 Cómputo Paralelo en Multihilos

En una computadora, sea secuencial o paralela, para aprovechar las capacidades crecientes del procesador, el sistema operativo divide su tiempo de procesamiento entre los distintos procesos, de forma tal que para poder ejecutar a un proceso, el Kernel les asigna a cada uno una prioridad y con ello una fracción del tiempo total de procesamiento, de forma tal que se pueda atender a todos y cada uno de los procesos de manera eficiente.

En particular, en la programación en paralelo usando MPI, cada proceso -que eventualmente puede estar en distinto procesador- se lanza como una copia del programa con datos privados y un identificador del proceso único, de tal forma que cada proceso sólo puede compartir datos con otro proceso mediante paso de mensajes.

Esta forma de lanzar procesos por cada tarea que se desee hacer en paralelo es costosa, por llevar cada una de ellas toda una gama de subprocesos para poderle asignar recursos por parte del sistema operativo. Una forma más eficiente de hacerlo es que un proceso pueda generar bloques de subprocesos que puedan ser ejecutados como parte del proceso (como subtareas), así en el tiempo asignado se pueden atender a más de un subproceso

de manera más eficiente, esto es conocido como programación multihilos.

Los hilos realizarán las distintas tareas necesarias en un proceso. Para hacer que los procesos funcionen de esta manera, se utilizan distintas técnicas que le indican al Kernel cuales son las partes del proceso que pueden ejecutarse simultáneamente y el procesador asignará una fracción de tiempo exclusivo al hilo del tiempo total asignado al proceso.

Los datos pertenecientes al proceso pasan a ser compartidos por los subprocesos lanzados en cada hilo y mediante una técnica de semáforos el Kernel mantiene la integridad de estos. Esta técnica de programación puede ser muy eficiente si no se abusa de este recurso, permitiendo un nivel más de paralelización en cada procesador. Esta forma de paralelización no es exclusiva de equipos multiprocesadores o multicomputadoras, ya que pueden ser implementados a nivel de sistema operativo.

22.4.5 Cómputo Paralelo en CUDA

Son las siglas de arquitectura unificada de dispositivos de cómputo (Compute Unified Device Architecture CUDA) que hace referencia a una plataforma de computación en paralelo (que incluye su propia RAM conocida como GRAM para hacer óptima la solicitud de dicho recurso por sus múltiples cores) incluyendo un compilador y un conjunto de herramientas de desarrollo que permiten a los programadores usar una variación del lenguaje de programación C -Por medio de Wrappers se puede usar MatLab, Python, Julia, Fortran y Java en vez de C/C++- para codificar algoritmos en las unidades de procesamiento de gráficos (Graphics Processing Unit GPU).

CUDA intenta explotar las ventajas de las GPU (de decenas a centenas de cores⁷⁴) frente a las CPU (decenas de cores⁷⁵) de propósito general utilizando el paralelismo que ofrecen sus múltiples cores o núcleos, que permiten el lan-

⁷⁴La GRAM de las tarjetas CUDA es relativamente pequeña (a lo más de algunas decenas de Gigabytes), pero está diseñada para que pueda ser compartida por sus cores de forma óptima. Pero si es necesario que múltiples cores de la GPU soliciten datos a la RAM del equipo de cómputo, pueden ocasionar degradación en la eficiencia de la GPU porque la RAM tiene una arquitectura DDR4 o DDR5 (lo que significa que solo puede atender peticiones de datos de sólo 4 o 5 cores simultáneamente).

⁷⁵Los actuales CPUs pueden tener decenas de cores y comparten hasta Terabytes de RAM. Pero la arquitectura actual de la RAM es de tipo DDR4 o DDR5, esto significa, que sólo hay 4 o 5 canales para satisfacer las solicitudes de petición de datos de/hacia los cores, si más cores necesita acceso a la RAM tendrán que esperar que los otros cores concluyan su uso, mermando la eficiencia de la CPU.

zamiento de un altísimo número de hilos simultáneos. Si una aplicación está bien diseñada utilizando numerosos hilos que realizan tareas independientes usando la memoria compartida de la tarjeta (que es lo que hacen las GPU al procesar gráficos, su tarea natural) hará un uso eficiente de este dispositivo.

Por ejemplo, la tarjeta gráfica de NVidia GEFORCE RTX 3090 proporciona 10,496 cores y 24 GB de GDDR6x, mientras que la tarjeta NVidia Titan RTX proporciona 130 Tensor⁷⁶ TFLOP de rendimiento, 576 núcleos tensores y 24 GB de memoria GDDR6.

Ahora, miles de desarrolladores, científicos e investigadores están encontrando innumerables aplicaciones prácticas para esta tecnología en campos como el procesamiento de vídeo e imágenes, la biología y la química computacional, la simulación de la dinámica de fluidos, la reconstrucción de imágenes de Tomografía Axial Computarizada TAC, el análisis sísmico o el trazado de rayos, entre otros.

Procesamiento paralelo con CUDA Los sistemas informáticos están pasando de realizar el «procesamiento central» en la CPU a realizar «coprocesamiento» repartido entre la CPU y la GPU. Para posibilitar este nuevo paradigma computacional, NVIDIA ha inventado la arquitectura de cálculo paralelo CUDA, que ahora se incluye en las GPUs GeForce, ION Quadro y Tesla GPUs, lo cual representa una base instalada considerable para los desarrolladores de aplicaciones.

CUDA ha sido recibida con entusiasmo por la comunidad científica. Por ejemplo, se está utilizando para acelerar AMBER, un simulador de dinámica molecular empleado por más de 60,000 investigadores del ámbito académico y farmacéutico de todo el mundo para acelerar el descubrimiento de nuevos medicamentos. En el mercado financiero, Numerix y CompatibL introdujeron soporte de CUDA para una nueva aplicación de cálculo de riesgo de contraparte y, como resultado, se ha multiplicado por 18 la velocidad de la aplicación. Cerca de 400 instituciones financieras utilizan Numerix en la actualidad.

Un buen indicador de la excelente acogida de CUDA es la rápida adopción de la GPU Tesla para aplicaciones de GPU Computing. En la actualidad hay más de 700 Clusters de GPUs instalados en compañías publicadas en Fortune

⁷⁶Un Tensor core (o núcleos Tensor) calculan la operación de una matriz 4x4 completa, la cual se calcula por reloj. Estos núcleos pueden multiplicar dos matrices FP16 4x4 y sumar la matriz FP32 al acumulador.

500 de todo el mundo, lo que incluye empresas como Schlumberger y Chevron en el sector energético o BNP Pariba en el sector bancario. Y en el 2021 se puso en funcionamiento el cluster Perlmutter que esta formado por 6,159 NVIDIA A100 Tensor Core GPUs para ciencias astrofísicas y atmosféricas.

Por otra parte, la reciente llegada de los últimos sistemas operativos de Microsoft y Apple está convirtiendo el GPU Computing en una tecnología de uso masivo. En estos nuevos sistemas, la GPU no actúa únicamente como procesador gráfico, sino como procesador paralelo de propósito general accesible para cualquier aplicación.

Plataforma de Cálculo Paralelo CUDA proporciona unas cuantas extensiones de MatLab, Fortran, Python, Julia, C, C++, etc. Que permiten implementar el paralelismo en el procesamiento de tareas y datos con diferentes niveles de granularidad. El programador puede expresar ese paralelismo mediante diferentes lenguajes de alto nivel como Fortran, Python, Julia, C y C++ o mediante estándares abiertos como las directivas de OpenACC -que es un estándar de programación para el cómputo en paralelo desarrollado por Cray, CAPS, Nvidia y PGI diseñado para simplificar la programación paralela de sistemas heterogéneos de CPU/GPU-. En la actualidad, la plataforma CUDA se utiliza en miles de aplicaciones aceleradas en la GPU y en miles de artículos de investigación publicados, en las áreas de:

- Bioinformática
- Cálculo financiero
- Dinámica de fluidos computacional (CFD)
- Ciencia de los datos, analítica y bases de datos
- Defensa e Inteligencia
- Procesamiento de imágenes y visión computarizadas
- EDA (diseño automatizado)
- Aprendizaje automático, Inteligencia artificial
- Ciencia de los materiales
- Medios audiovisuales y entretenimiento

- Imágenes médicas
- Dinámica molecular
- Análisis numérico
- Química cuántica
- Exploración sísmica
- Mecánica estructural computacional
- Visualización e interacción de proteínas
- Modelos meteorológicos y climáticos

Algunas bibliotecas aceleradas en la GPU:

- Thrust C++ Template
- cuBLAS
- cuSPARSE
- NPP
- cuFFT

Algunos lenguajes de programación:

- CUDA C/C++, Fortran, Python, Julia
- .NET, Java

Algunos compiladores disponibles:

- OpenACC, paraleliza automáticamente los bucles de código Fortran o C utilizando directivas
- Compilador de autoparalelización de C y Fortran (de PGI) para CUDA C
- Compilador de autoparalelización HMPP de CAPS para CUDA C basado en C y Fortran

- Fortran, Python, Julia, Java, C++ y C
- Compilador de Fortran para CUDA de PGI
- Traductor de Fortran a C para CUDA
- FLAGON: librería de Fortran 95 para cálculo en la GPU
- Interfaz (Wrapper) de Python para CUDA: PyCUDA
- Wrapper de Java
- jCUDA: Java para CUDA
- Vínculos para las librerías BLAS y FFT de CUDA
- JaCUDA
- Integración de .NET para CUDA
- Thrust: librería de plantillas de C++ para CUDA
- CuPP : infraestructura de C++ para CUDA
- Libra: capa de abstracción de C/C++ para CUDA
- F# para CUDA
- Librería ArrayFire para acelerar el desarrollo de aplicaciones de GPU Computing en C, C++, Fortran y Python

Soporte de MATLAB, Mathematica, R, LabView:

- MATLAB, Mathematica, R, LabView
- MathWorks: Librerías MATLAB procesadas con GPUs NVIDIA
- Plugin Jacket basado en CUDA para MATLAB
- GPULib: librería de funciones matemáticas con vínculos para IDL y MATLAB
- Programación con CUDA en Mathematica de Wolfram

- Plugin de Mathematica para CUDA
- Habilitación del GPU Computing en el entorno estadístico de R
- Librería CUDA para LabVIEW de National Instruments
- Formas de usar CUDA desde LabVIEW CVI

Además existen herramientas de productividad y Cluster:

- Soporte de Eclipse para CUDA
- CUDA Occupancy Calculator
- Administrador de Clusters de cálculo para GPUs Tesla
- PBS Works para GPUs Tesla
- Scyld de Penguin Computing

22.5 Métricas de Desempeño

Las métricas de desempeño del procesamiento de alguna tarea en paralelo es un factor importante para medir la eficiencia y consumo de recursos al resolver una tarea con un número determinado de procesadores y recursos relacionados de la interconexión de éstos.

Entre las métricas para medir desempeño en las cuales como premisa se mantiene fijo el tamaño del problema, destacan las siguientes:

- Factor de aceleración
- Eficiencia
- Fracción serial

Cada una de ellas mide algo en particular y sólo la combinación de estas dan un panorama general del desempeño del procesamiento en paralelo de un problema en particular en una arquitectura determinada al ser comparada con otras.

Factor de Aceleración (o Speed-Up) Se define como el cociente del tiempo que se tarda en completar el cómputo de la tarea usando un sólo procesador entre el tiempo que necesita para realizarlo en p procesadores trabajando en paralelo:

$$s = \frac{T(1)}{T(p)} \quad (22.1)$$

se asume que se usará el mejor algoritmo tanto para un solo procesador como para p procesadores.

Esta métrica en el caso ideal debería de aumentar de forma lineal al aumento del número de procesadores.

Eficiencia Se define como el cociente del tiempo que se tarda en completar el cómputo de la tarea usando un solo procesador entre el número de procesadores multiplicado por el tiempo que necesita para realizarlo en p procesadores trabajando en paralelo:

$$e = \frac{T(1)}{pT(p)} = \frac{s}{p}. \quad (22.2)$$

Este valor será cercano a la unidad cuando el Hardware se esté usando de manera eficiente, en caso contrario el Hardware será desaprovechado.

Fracción serial Se define como el cociente del tiempo que se tarda en completar el cómputo de la parte secuencial de una tarea entre el tiempo que se tarda en completar el cómputo de la tarea usando un solo procesador:

$$f = \frac{T_s}{T(1)} \quad (22.3)$$

pero usando la ley de Amdahl:

$$T(p) = T_s + \frac{T_p}{p}$$

y reescribiéndola en términos de factor de aceleración, obtenemos la forma operativa del cálculo de la fracción serial que adquiere la forma siguiente:

$$f = \frac{\frac{1}{s} - \frac{1}{p}}{1 - \frac{1}{p}}. \quad (22.4)$$

Esta métrica permite ver las inconsistencias en el balance de cargas, ya que su valor debería de tender a cero en el caso ideal, por ello un incremento en el valor de f es un aviso de granularidad fina con la correspondiente sobrecarga en los procesos de comunicación.

Costo o Trabajo Se define el costo o trabajo de resolver un problema en paralelo como el producto del tiempo de cálculo en paralelo T_p por el número de procesadores usando p y se representa por:

$$C_p = p * T_p.$$

Aceleración y Eficiencia Relativa Cuando se trabaja en más de un equipo paralelo -supongamos con p y p' procesadores con $p \geq p'$ - es común comparar el desempeño entre dichos equipos. Para ello se define la aceleración relativa como:

$$S_p^{p'} = \frac{T_p}{T_{p'}}$$

para $p \geq p'$, en la cual se espera que:

$$S_p^{p'} \simeq \frac{p}{p'}$$

y eficiencia relativa como:

$$E_p^{p'} = \frac{p'}{p} S_p^{p'} = \frac{p'}{p} \frac{T_p}{T_{p'}}.$$

Análisis de Rendimiento Usando Métricas Suponiendo un Cluster con 17 Cores o núcleos⁷⁷, se muestra una ejemplificación de las métricas para un problema de ejemplo:

⁷⁷A cada procesador encapsulado se le llama Core o núcleo, logrando que la comunicación entre ellos se realiza de una forma más rápida a través de un bus interno integrado en la propia pastilla de silicio sin tener que recurrir por tanto al bus del sistema mucho más lento. Al contrario del caso de la tecnología HyperThreading que el sistema operativo los reconoce como un core en el equipo, pero estos cores virtuales distan mucho del desempeño de uno real.

Cores	Tiempo	Aceleración	Eficiencia	Frac. Ser.
1	5295			
3	2538	2.08	0.69	0.218
5	1391	3.80	0.76	0.078
9	804	6.58	0.73	0.045
17	441	12.00	0.70	0.025

Nótese que en todos los casos la fracción serial disminuye sustancialmente con el aumento del número de procesadores, pero la aceleración está por debajo del valor esperado.

Suponiendo un Cluster A con 100 Cores y un Cluster B con 128 Cores, se muestra una ejemplificación de las métricas para un problema de ejemplo en ambos Clusters con los siguientes resultados:

Cores	Cluster A	Cluster B
16	9158 seg	ND
32	5178 seg	5937 seg
64	3647 seg	4326 seg
100	2661 seg	
128		2818 seg

Como se muestra en la tabla, en todos los casos el Cluster A usando como máximo 100 Cores obtiene un tiempo de cálculo inferior al que requiere Cluster B usando a lo más los 128 Cores.

Haciendo uso de las métricas de aceleración y eficiencia relativa⁷⁸ se tiene que para el Cluster B:

$$S_{128}^{32} = 5937/2818 = 2.10$$

donde lo esperado sería:

$$S_{128}^{32} = 32/128 = 4.00,$$

para el caso de la eficiencia:

$$E_{128}^{32} = (32/128) * (5937/2818) = 0.52.$$

⁷⁸ Aceleración relativa es $S_p^{p'} = \frac{T_p}{T_{p'}}$ para $p \geq p'$, en la cual se espera que $S_p^{p'} \simeq \frac{p}{p'}$ y eficiencia relativa es $E_p^{p'} = \frac{p'}{p} S_p^{p'} = \frac{p'}{p} \frac{T_p}{T_{p'}}$.

En el caso del Cluster A se tiene que:

$$S_{100}^{16} = 9158/2661 = 3.44$$

dónde lo esperado sería:

$$S_{100}^{16} = 16/100 = 6.35,$$

para el caso de la eficiencia:

$$E_{100}^{16} = (16/100) * (9158/2661) = 0.55.$$

Haciendo uso del mismo número de Cores base para el Cluster A que para Cluster B, se tiene que:

$$S_{100}^{32} = 5178/2661 = 1.94$$

dónde lo esperado sería:

$$S_{100}^{16} = 32/100 = 3.12,$$

para el caso de la eficiencia:

$$E_{100}^{16} = (32/100) * (5178/2661) = 0.62.$$

De todo lo anterior, se desprende que el Cluster A obtiene valores de una aceleración y eficiencias relativas ligeramente mejores que el Cluster B, pero esto no se refleja en la disminución de casi 6% del tiempo de ejecución y del uso de 28 Cores menos.

Además, el costo computacional:

$$C_p = P * T_p,$$

que para el caso del Cluster B es:

$$C_{128} = 360,704$$

y en Cluster A es:

$$C_{100} = 266,100$$

que representa una disminución de 27%; además de un factor muy importante, el Cluster A tuvo un costo monetario mucho menor con respecto del Cluster B.

22.6 Programación de Cómputo de Alto Rendimiento

Hay muchas aplicaciones a las herramientas computacionales, pero nos interesan aquellas que permitan resolver problemas concomitantes en Ciencia e Ingeniería. Muchas de estas aplicaciones caen en lo que comúnmente se llama cómputo científico. La computación científica es el campo de estudio relacionado con la construcción de modelos matemáticos, técnicas numéricas para resolver problemas científicos y de ingeniería; y su respectiva implementación computacional. La solución de estos problemas genera un alto consumo de memoria, espacio de almacenamiento o tiempo de cómputo; por ello nos interesa trabajar en computadoras que nos puedan satisfacer estas demandas.

La computación de alto rendimiento -High performance Computing o HPC en inglés- es la agregación de potencia de cálculo para resolver problemas complejos en Ciencia e Ingeniería o Gestión. Para lograr este objetivo, la computación de alto rendimiento se apoya en tecnologías computacionales como los Clusters, las supercomputadoras o la computación paralela. La mayoría de las ideas actuales de la computación distribuida se han basado en la computación de alto rendimiento.

La computación paralela o de alto rendimiento es una forma de cómputo en la que muchas instrucciones se ejecutan simultáneamente, operando sobre el principio de que problemas grandes, a menudo se pueden dividir en unos más pequeños, que luego son resueltos simultáneamente (en paralelo). Hay varias formas diferentes de computación paralela: paralelismo a nivel de bits, paralelismo a nivel de instrucción, paralelismo de datos y paralelismo de tareas.

El paralelismo se ha empleado durante muchos años, sobre todo en la computación de altas prestaciones, pero el interés en ella ha crecido últimamente debido a las limitaciones físicas que impiden el aumento de la frecuencia de los actuales procesadores comerciales. Como el consumo de energía –y por consiguiente la generación de calor– de las computadoras constituye una preocupación en los últimos años, la computación en paralelo se ha convertido en el paradigma dominante en la arquitectura de computadoras, principalmente en forma de procesadores multinúcleo.

Las computadoras paralelas pueden clasificarse según el nivel de paralelismo que admite su Hardware:

- Equipos con procesadores multinúcleo

- Equipos con multiprocesador (múltiples procesadores multinúcleo en una sola máquina)
- Procesadores masivamente paralelos
- Equipos con una o más tarjetas CUDA (Compute Unified Device Architecture)
- Cluster de equipos multinúcleo, multiprocesador y/o CUDAS
- Cúmulos de Clusters (Grids)

Muchas veces, para acelerar tareas específicas, se utilizan arquitecturas especializadas de computación en paralelo junto a procesadores tradicionales.

Existen múltiples vertientes en el cómputo en paralelo, algunas de ellas son:

- Cómputo en memoria compartida usando OpenMP
- Cómputo en memoria distribuida usando MPI

Como es de esperarse, los programas informáticos paralelos son más complejos y difíciles de escribir que los secuenciales, porque la concurrencia introduce nuevos tipos de errores de Software -por ello existe una creciente gama de **herramientas** que coadyuvan a mejorar la escritura, depuración y desempeño de los programas en paralelo-, pero la comunicación y sincronización entre diferentes subtareas son algunos de los mayores obstáculos para obtener un buen rendimiento del programa paralelo.

Además, el tiempo de programación necesario para desarrollar una aplicación paralela eficiente y eficaz para la gran mayoría de los programadores puede ser de semanas o meses en el mejor de los casos. Por ello, es necesario hacer un balance entre las diferentes opciones de paralelización para no invertir un tiempo precioso que puede no justificar dicha inversión económica, de recursos computacionales y sobre todo de tiempo.

Actualmente, en muchos centros de cómputo es una práctica común usar directivas de compilación en equipos paralelos sobre programas escritos de forma secuencial, con la esperanza que sean puestos por el compilador como programas paralelos. Esto en la gran mayoría de los casos genera códigos poco eficientes, pese a que corren en equipos paralelos y pueden usar toda

la memoria compartida de dichos equipos, el algoritmo ejecutado continúa siendo secuencial en una gran parte del código.

Si la arquitectura paralela donde se implemente el programa es UMA de acceso simétrico, los datos serán accedados a una velocidad de memoria constante. En caso contrario, al acceder a un conjunto de datos es común que una parte de estos sean locales a un procesador (con un acceso del orden de nanosegundos), pero el resto de los datos deberán ser accedados mediante red (con acceso del orden de milisegundos), siendo esto muy costoso en tiempo de procesamiento.

Por lo anterior, si se cuenta con computadoras con memoria compartida o que tengan interconexión por bus, salvo en casos particulares no será posible explotar éstas características eficientemente. Pero en la medida en que se adecuen los programas para usar bibliotecas y compiladores acordes a las características del equipo disponible -algunos de ellos sólo existen de manera comercial- la eficiencia aumentará de manera importante.

22.6.1 Programando con OpenMP para Memoria Compartida

OpenMP es una interfaz de programación de aplicaciones (API) para la programación multiproceso de memoria compartida en múltiples plataformas. Permite añadir concurrencia a los programas escritos en C, C++, Julia, Python y Fortran sobre la base del modelo de ejecución Fork-join. Está disponible en muchas arquitecturas, incluidas las plataformas de Unix, Linux y de Microsoft Windows. Se compone de un conjunto de directivas de compilador, rutinas de biblioteca, y variables de entorno que influyen en el comportamiento en tiempo de ejecución.

Definido conjuntamente por proveedores de Hardware y de Software, OpenMP es un modelo de programación portable y escalable que proporciona a los programadores una interfaz simple y flexible para el desarrollo de aplicaciones paralelas, para plataformas que van desde las computadoras de escritorio hasta supercomputadoras. Una aplicación construida con un modelo de programación paralela híbrido se puede ejecutar en un Cluster de computadoras utilizando OpenMP y MPI, o a través de las extensiones de OpenMP para los sistemas de memoria distribuida.

OpenMP se basa en el modelo *Fork-join*, paradigma que proviene de los sistemas Unix, donde una tarea muy pesada se divide en K hilos (Fork) con menor peso, para luego «recolectar» sus resultados al final y unirlos en un solo resultado (Join).

Cuando se incluye una directiva de compilador OpenMP esto implica que se incluye una sincronización obligatoria en todo el bloque. Es decir, el bloque de código se marcará como paralelo y se lanzarán hilos según las características que nos dé la directiva, y al final de ella habrá una barrera para la sincronización de los diferentes hilos (salvo que implícitamente se indique lo contrario con la directiva `Nowait`). Este tipo de ejecución se denomina *Fork-join*.

OpenMP también soporta el modelo de paralelismo de tareas. El equipo de hilos del bloque de código paralelo ejecuta las tareas especificadas dentro de dicho bloque. Las tareas permiten un paralelismo asíncrono. Desde la versión 4.0 lanzada en 2013 admite la especificación de dependencias entre tareas, relegando a la biblioteca de tiempo de ejecución de OpenMP el trabajo de planificar las tareas y ponerlas en ejecución. Los hilos de ejecución irán ejecutando las tareas a medida que estas estén disponibles (sus dependencias ya estén satisfechas). El uso de tareas da lugar a sincronización con una granularidad más fina. El uso de barreras no es estrictamente necesario, de manera que los hilos pueden continuar ejecutando tareas disponibles sin necesidad de esperar a que todo el equipo de hilos acabe un bloque paralelo. El uso de tareas con dependencias crea un grafo, pudiéndose aplicar propiedades de grafos a la hora de escoger tareas para su ejecución.

Salvo el uso de implementaciones de Hardware de la biblioteca de tiempo de ejecución OpenMP (p.ej. en una matriz de puertas programables FPGAs), los sobrecostes de las tareas es mayor, este sobrecoste ha de ser amortizado mediante el potencial paralelismo adicional que las tareas exponen.

Estructura del Programa en C++ Ejemplo de cálculo de Pi usando OpenMP:

```
#include <stdio.h>
// Indica si se carga lo referente a OpenMP
#ifdef _OPENMP
#include <omp.h>
int threads=omp_get_num_threads();
#else
int threads=0;
#endif
#define STEPCOUNTER 1000000000
int main (void)
```

```
{
  long i;
  double pi=0;
  printf("threads %d", threads);
#pragma omp parallel for reduction(+:pi)
  for (i=0; i < STEPCOUNTER; i++)
  {
    pi += 1.0/(i*4.0 +1.0);
    pi -= 1.0/(i*4.0 +3.0);
  }
  pi = pi*4.0;
  printf("PI = %2.16lf ",pi);
  return 0;
}
```

El compilador de OpenMP es el mismo que para los lenguajes C, C++ o Fortran respectivamente. Por ello, para usarlo en C++ en línea de comandos, instalamos el compilador g++, mediante:

```
# apt install g++
```

Así, para compilar con g++⁷⁹, sin usar OpenMP, usamos:

```
$ g++ pi.cpp -o pi
```

Ejecutar midiendo el tiempo⁸⁰:

```
$ time ./pi
```

⁷⁹Compilar fuentes en C++ solicitando que el ejecutable tenga el nombre *ejemp*:

```
$ g++ *.cpp -o ejemp
```

en este caso no se usa ninguna directiva para optimizar el ejecutable generado. Para compilar usando diversas optimizaciones (O1, -O2 o -O3) usar por ejemplo:

```
$ g++ -O1 *.cpp
```

⁸⁰Los sistemas operativos tipo Linux/Unix cuentan con un comando básico que mide el tiempo de ejecución de cualquier comando, usando:

```
$ time ls
```

Ahora, usando el compilador para OpenMP usamos:

```
$ g++ -o pi -fopenmp pi.cpp
```

Indicar el número de hilos, por ejemplo 2:

```
$ export OMP_NUM_THREADS=2
```

Ejecutar:

```
$ time ./pi
```

22.6.2 Programando con MPI para Memoria Distribuida

Para poder intercomunicar dos o más Cores en una o en múltiples computadoras se usa la «interfaz de paso de mensajes (Message Passing Interface MPI)» (véase [15], [16], [78] y [17]), una biblioteca de comunicación para procesamiento en paralelo que puede ser usada desde lenguajes de programación como: C, C++, Python, Julia, Fortran. MPI ha sido desarrollado como un estándar para el paso de mensajes y operaciones relacionadas.

Este enfoque es adoptado por usuarios e implementadores de bibliotecas, en el cual se proveen a los programas de procesamiento en paralelo de portabilidad y herramientas necesarias para desarrollar aplicaciones que puedan usar el cómputo paralelo de alto desempeño.

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

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

pero podemos instalar el paquete que además de medir el tiempo de ejecución nos diga que recursos se usaron en la ejecución del comando indicado, para instalarlo usamos:

```
# apt install time
```

el uso es el mismo del comando interno time.

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

- Para el que envía, la dirección de los datos a transmitir y el proceso destino al cual los datos se enviarán.
- Para el que recibe, debe tener la dirección de memoria donde se pondrán los datos recibidos, junto con la dirección del proceso que los envió.

Es decir:

$Send(dir, lg, td, dest, etiq, com)$

$\{dir, lg, td\}$ describe cuántas ocurrencias lg de elementos del tipo de dato td se transmitirán empezando en la dirección de memoria dir .

$\{des, etiq, com\}$ describe el identificador $etiq$ de destino des asociado con la comunicación com .

$Recv(dir, mlq, td, fuent, etiq, com, st)$

$\{dir, lg, td\}$ describe cuántas ocurrencias lg de elementos del tipo de dato td se transmitirán empezando en la dirección de memoria dir .

$\{fuent, etiq, com, est\}$ describe el identificador $etiq$ de la fuente $fuent$ asociado con la comunicación com y el estado est .

El conjunto básico de directivas (en nuestro caso sólo se usan estas) en C++ de MPI son (véase [15] y [16]):

MPI::Init	Inicializa al MPI
MPI::COMM_WORLD.Get_size	Busca el número de procesos existentes
MPI::COMM_WORLD.Get_rank	Busca el identificador del proceso
MPI::COMM_WORLD.Send	Envía un mensaje
MPI::COMM_WORLD.Recv	Recibe un mensaje
MPI::Finalize	Termina al MPI

Estructura del Programa en C++ Ejemplo de Hola_Mundo en MPI:

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char *argv[])
{
    int rank, size;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    printf("Hola! Soy el %d de %d\n", rank, size);
    MPI_Finalize();
    return 0;
}
```

Otro ejemplo, para realizar una suma en MPI:

```
#include <iostream>
#include <iomanip>
#include <mpi.h>
using namespace std;
int main(int argc, char ** argv){
    int mynode, totalnodes;
    int sum = 0,startval,endval,accum;
    MPI_Status status;
    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD, &totalnodes);
    MPI_Comm_rank(MPI_COMM_WORLD, &mynode);
    startval = 1000*mynode/totalnodes+1;
    endval =1000*(mynode+1)/totalnodes;
    for(int i=startval;i<=endval;i=i+1) sum = sum + i;
    if(mynode!=0)
    MPI_Send(&sum, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);
    else
    for(int j=1;j<totalnodes;j=j+1){
    MPI_Recv(&accum, 1, MPI_INT, j, 1,
    MPI_COMM_WORLD, &status);
    sum = sum + accum;
    }
```

```
}
if(mynode == 0)
cout << "The sum from 1 to 1000 is: " << sum << endl;
MPI_Finalize();
}
```

Existe una gran variedad de compiladores de MPI en línea de comandos, algunos disponibles en Debian GNU/Linux son instalados mediante:

```
# apt install lam-runtime libmpich-dev mpi-default-dev mpich\
mpi-default-bin openmpi-bin valgrind-mpi xmpi
```

Para compilar y ejecutar es posible usar alguna de estas opciones:

```
mpic++, mpic++.openmpi, mpiexec.mpich, mpif90.openmpi,
mpirun.lam, mpicc, mpicxx, mpiexec.openmpi, mpifort, mpirun.mpich,
mpiCC, mpicxx.mpich, mpif77, mpifort.mpich, mpirun.openmpi,
mpicc.mpich, mpicxx.openmpi, mpif77.mpich, mpifort.openmpi,
mpitask, mpicc.openmpi, mpiexec, mpif77.openmpi, mpimsg, mpi-
vars, mpiCC.openmpi, mpiexec.hydra, mpif90, mpipython, mpichver-
sion, mpipython, mpiexec.lam, mpif90.mpich, mpirun
```

Por ejemplo, para compilar *ejemp.cpp* en *mpic++* solicitando que el ejecutable tenga el nombre *ejemp*, usamos:

```
$ mpic++ ejemp.cpp -o ejemp
```

en este caso no se uso ninguna opción de optimización en tiempo de compilación, se puede hacer uso de ellas (-O1, -O2 o -O3), mediante:

```
$ mpic++ -O3 ejemp.cpp -o ejemp
```

para ejecutar el programa ya compilado en 4 procesos y medir el tiempo de ejecución, usamos:

```
$ time mpirun -np 4 ejemp
```

También podemos compilar *ejemp.c* en *mpicc* solicitando que el ejecutable tenga el nombre *ejemp*:

```
$ mpicc ejemp.cpp -o ejemp
```

en este caso no se usó ninguna opción de optimización en tiempo de compilación, se puede hacer uso de ellas (-O1, -O2 o -O3), mediante:

```
$ mpicc -O3 ejemp.c -o ejemp
```

para ejecutar el programa ya compilado en 4 procesos, usamos:

```
$ mpirun -np 4 ejemp
```

Un último ejemplo, en el caso de usar *mpiCC.mpic* y *lamboot*, entonces es necesario compilar usando:

```
$ mpiCC.mpich -O2 ejemp.cpp -o ejemp -lm
```

iniciar ambiente de ejecución paralelo:

```
$ lamboot -v
```

correr usando 8 procesos por ejemplo:

```
$ mpirun.mpich -np 8 ejemp
```

correr usando 5 procesos según lista *machines.lolb* por ejemplo:

```
$ mpirun.mpich -machinefile machines.lolb -np 5 ejemp
```

terminar ambiente de ejecución paralelo:

```
$ lamhalt -v
```

Observación 12 *Para que en la ejecución de MPI no pida la clave de usuario:*

```
$ ssh-keygen -t rsa
```

En cada pregunta responder con ENTER, para después copiar usando:

```
$ ssh-copy-id usuario@servidor
```

Ojo: Si continúa pidiendo clave, es que está instalado rsh o lsh.

22.6.3 Esquema de Paralelización Principal-Subordinados

El esquema de paralelización Principal-Subordinados -también conocido como Maestro-Esclavo-, permite sincronizar por parte del nodo principal las tareas que se realizan en paralelo usando varios nodos subordinados, éste modelo puede ser explotado de manera eficiente si existe poca comunicación entre el principal y el subordinado -entre los subordinados no debe de existir comunicación- y los tiempos consumidos en realizar las tareas asignadas son mayores que los períodos involucrados en las comunicaciones para la asignación de dichas tareas. De esta manera se garantiza que la mayoría de los procesadores estarán trabajando de manera continua y existirán pocos tiempos muertos.

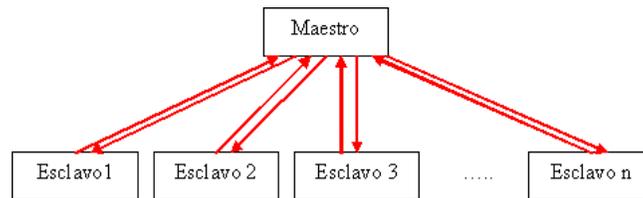


Figura 40: Esquema del Maestro-Esclavo

Donde, tomando en cuenta la implementación en estrella del Cluster, el modelo de paralelismo de MPI (véase 22.6.2) y las necesidades propias de comunicación del programa, el nodo principal tendrá comunicación sólo con cada nodo subordinado y no existirá comunicación entre los nodos subordinados, esto reducirá las comunicaciones y optimizará el paso de mensajes, algunos ejemplos de este esquema se pueden consultar en:

Herramientas
<http://132.248.181.216/Herramientas/>

Un factor limitante en este esquema es que el nodo principal deberá de atender todas las peticiones hechas por cada uno de los nodos subordinados, esto toma especial relevancia cuando todos o casi todos los nodos subordinados compiten por ser atendidos por el nodo principal.

Se recomienda implementar este esquema en un Cluster heterogéneo en donde el nodo principal sea más poderoso computacionalmente que los nodos subordinados. Si a este esquema se le agrega una red de alta velocidad y de

baja latencia, se le permitirá operar al Cluster en las mejores condiciones posibles, pero este esquema se verá degradado al aumentar el número de nodos subordinados inexorablemente.

Pero hay que ser cuidadosos en cuanto al número de nodos subordinados que se usan en la implementación en tiempo de ejecución versus el rendimiento general del sistema al aumentar estos, algunas observaciones posibles son:

- El esquema Principal-Subordinados programado usando MPI lanza P procesos (uno para el nodo principal y $P - 1$ para los nodos subordinados), estos en principio corren en un solo procesador pero pueden ser lanzados en múltiples procesadores usando una directiva de ejecución, de esta manera es posible que en una sola máquina se programe, depure y sea puesto a punto el código usando mallas pequeñas (del orden de cientos de nodos) y cuando este listo puede mandarse a producción en un Cluster.
- El esquema Principal-Subordinados no es eficiente si sólo se usan dos procesadores (uno para el nodo principal y otro para el nodo subordinado), ya que el nodo principal en general no realiza los cálculos pesados y su principal función será la de distribuir tareas; los cálculos serán delegados al nodo subordinado.

Estructura del Programa Principal-Subordinados La estructura del programa se realizó para que el nodo principal mande trabajos de manera síncrona a los nodos subordinados. Cuando los nodos subordinados terminan la tarea asignada, avisan al nodo principal para que se les asigne otra tarea (estas tareas son acordes a la etapa correspondiente del método de descomposición de dominio ejecutándose en un instante dado). En la medida de lo posible se trata de mandar paquetes de datos a cada nodo subordinado y que estos regresen también paquetes al nodo principal, a manera de reducir las comunicaciones al mínimo y tratar de mantener siempre ocupados a los nodos subordinados para evitar los tiempos muertos, logrando con ello una granularidad gruesa, ideal para trabajar con Clusters.

La estructura básica del programa bajo el esquema Principal-Subordinados codificada en C++ y usando MPI (véase [22.6.2](#)) es:

```
main(int argc, char *argv[])
```

```
{
    MPI::Init(argc,argv);
    ME_id = MPI::COMM_WORLD.Get_rank();
    MP_np = MPI::COMM_WORLD.Get_size();
    if (ME_id == 0) {
        // Operaciones del Principal
    } else {
        // Operaciones del Subordinado con identificador ME_id
    }
    MPI::Finalize();
}
```

En este único programa se deberá de codificar todas las tareas necesarias para el nodo principal y cada uno de los nodos subordinados, así como las formas de intercomunicación entre ellos usando como distintivo de los distintos procesos a la variable *ME_id*. Para más detalles de esta forma de programación y otras funciones de MPI (véase [22.6.2](#), [\[15\]](#) y [\[16\]](#)).

El principal factor limitante para el esquema Principal-Subordinados es que se presupone contar con un nodo principal lo suficientemente poderoso para atender simultáneamente las tareas síncronas del método, ya que este distribuye tareas acorde al número de nodos subordinados, estas si son balanceadas, ocasionaran que muchos de los procesadores subordinados terminen aproximadamente al mismo tiempo y el nodo principal tendrá que atender múltiples comunicaciones simultáneamente, degradando su rendimiento al aumentar el número de nodos subordinados que atender.

Para los factores limitantes inherente al propio esquema Principal-Subordinados, es posible implementar algunas operaciones del nodo principal en paralelo, ya sea usando equipos multiprocesador o en más de un nodo distinto a los nodos subordinados.

22.6.4 Opciones de Paralelización Híbridas

En la actualidad, casi todos los equipos de cómputo usados en estaciones de trabajo y Clusters cuentan con dos o más Cores, en ellos siempre es posible usar MPI para intercambiar mensajes entre procesos corriendo en

el mismo equipo de cómputo, pero no es un proceso tan eficiente como se puede querer. En estas arquitecturas llamadas de memoria compartida es mejor usar OpenMP o cualquiera de sus variantes para trabajar en paralelo. Por otro lado es común contar con las cada vez más omnipresentes tarjetas NVIDIA, y con los cada vez más numerosos Cores CUDA -que una sola tarjeta NVIDIA TESLA puede tener del orden de miles de ellos- y que en un futuro serán cada vez más numerosos.

Para lograr obtener la mayor eficiencia posible de estos tres niveles de paralelización, se están implementando procesos híbridos (véase [75] y [76]), en donde la intercomunicación de equipos con memoria compartida se realiza mediante MPI y la intercomunicación entre Cores que comparten la misma memoria se realiza con OpenMP, además las operaciones matriciales, vectoriales, etc. se le encargan a los numerosos Cores CUDA de las tarjetas NVIDIA.

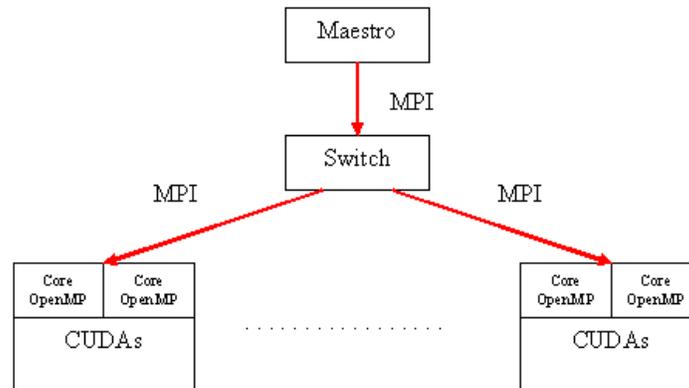


Figura 41: Paralelización Híbrida

Se han desarrollado múltiples programas que sus algoritmos resuelven ecuaciones diferenciales parciales concomitantes en Ciencias e Ingenierías que hacen uso de esta forma integradora de paralelismo. Para ello, la interconexión de equipos de memoria compartida se realizaría mediante MPI y en cada equipo de memoria compartida se manipularían uno o más subdominios mediante OpenMP -ya que cada subdominio es independiente de los demás- y la manipulación de matrices y operaciones entre matrices y vectores que requiere cada subdominio se realizarían en las tarjetas NVIDIA mediante los numerosos Cores CUDA sin salir a la RAM de la computadora.

Permitiendo así, tener una creciente eficiencia de paralelización que optimizan en gran medida los recursos computacionales, ya que todas las matrices y vectores se generarían en la GRAM de la tarjeta NVIDIA. De forma tal que sea reutilizable y que pueda usarse en problemas en los que el número de grados de libertad sea grande, permitiendo hacer uso de equipos de cómputo cada vez más asequibles y de menor costo, pero con una creciente eficiencia computacional que compiten con los grandes equipos de cómputo de alto desempeño.

22.7 Programando Desde la Nube

Existen diferentes servicios Web⁸¹ que permiten editar, compilar y ejecutar código de diversos lenguajes y paquetes desde el **navegador**, esto en aras de que los estudiantes y profesores que cuenten con algún sistema de acceso a red y un navegador puedan programar en los más diversos lenguajes, IDEs y terminales sin hacer instalación alguna en su equipo de cómputo, tableta o teléfono celular.

Google Colaboratory Integrante de la G Suite for Education de Google permite a los usuarios que pertenezcan a esta Suite (como gran parte de los estudiantes de la UNAM) tener acceso desde el navegador para escribir y ejecutar código de Python (Jupyter), es posible elegir correr nuestro Notebook en una CPU, GPU o en una TPU de forma gratuita. Tiene algunas restricciones, como por ejemplo que una sesión dura 12 hrs, pasado ese tiempo se limpia nuestro ambiente y perdemos las variables y archivos que tengamos almacenados allí.

Es conveniente para principiantes que requieran experimentar con Machine Learning y Deep Learning pero sin recurrir en costos de procesamiento Cloud. Además el ambiente de trabajo ya viene con muchas librerías instaladas y listas para utilizar (como por ejemplo *Tensorflow*, *Scikit-learn*, *Pytorch*, *Keras* y *OpenCV*), ahorrándonos el trabajo de configurar nuestro ambiente

⁸¹Cuando se trabaja desde la Web es recomendable usar el modo Privado o Incógnito para no guardar el historial de navegación, información introducida en los formularios y borrar al cerrar el navegador los datos de los sitios visitados. Pero recuerda que los sitios Web que visitamos sí guardan información de nuestra visita, nuestro proveedor de Internet también guarda constancia de nuestra visita y si descargamos algo, esto no se borra al igual que el historial de descargas, además de las marcas de páginas o favoritos se conservarán al cerrar el navegador.

de trabajo. Podemos importar nuestros archivos y datos desde *Google Drive*, *GitHub*, etc.

Más información sobre Google Colaboratory en:

<https://colab.research.google.com/notebooks/intro.ipynb>

23 Apéndice E: Aritmética de Punto Flotante

La aritmética de punto flotante es considerada un tema esotérico para muchas personas. Esto es sorprendente porque el punto flotante es omnipresente en los sistemas informáticos. Casi todos los lenguajes de programación tienen un tipo de datos de punto flotante, i.e. los números no enteros como $1.2 \times 10^4 + 45$.

Un número de punto flotante de 64 bits (*double* en lenguaje C) tiene aproximadamente 16 dígitos decimales de precisión con un rango del orden de 1.7×10^{-308} a 1.7×10^{308} de acuerdo con el estándar 754 de IEEE, la implementación típica de punto flotante⁸².

Una de las primeras cosas que uno encuentra con sorpresa cuando hace cálculos en una computadora es que por ejemplo, si usamos números muy grandes y seguimos incrementando su valor eventualmente el resultado será negativo ... ¿qué pasó? Esto se llama desbordamiento aritmético al intentar crear un valor numérico que está fuera del rango que puede representarse con un número dado de dígitos, ya sea mayor que el máximo o menor que el mínimo valor representable. Algo similar pasa al restar al menor número representable en la máquina, el resultado será positivo y se denomina sub-desbordamiento.

Las computadoras, desde las PC hasta las supercomputadoras, tienen aceleradores de punto flotante; la mayoría de los compiladores deberán compilar algoritmos de punto flotante de vez en cuando y prácticamente todos los sistemas operativos deben responder a excepciones de punto flotante como el desbordamiento.

Desde ya hace mucho tiempo, los procesadores Intel x86 y todos los procesadores de las siguientes generaciones de todas las marcas admiten un formato de precisión extendido de 80 bits con un significado de 64 bits, que es compatible con el especificado en el estándar IEEE. Cuando un compilador usa este formato con registros de 80 bits para acumular sumas y productos internos, está trabajando efectivamente con un redondeo unitario de 2^{-64} en vez de 2^{-53} para precisión doble, dando límites de error más pequeños en un factor de hasta $2^{11} = 2048$.

¿Dieciséis lugares decimales es mucho? Casi ninguna cantidad medida

⁸² Además, existe el número de 80 bits (*long double* en lenguaje C) que tiene 18 dígitos decimales de precisión con un rango del orden de 3.4×10^{-4096} a 1.1×10^{4096} . Y no podemos olvidar, el número de 32 bits (*float* en lenguaje C) que tiene 14 dígitos decimales de precisión con un rango del orden de 3.4×10^{-38} a 3.4×10^{38} .

se conoce con tanta precisión. Por ejemplo: en la constante en la ley de gravedad de Newton sólo se conoce con seis cifras significativas. En la carga de un electrón se conoce con 11 cifras significativas, mucha más precisión que la constante gravitacional de Newton, pero aún menos que un número de punto flotante⁸³.

Entonces, ¿cuándo no son suficientes 16 dígitos de precisión? Un área problemática es la resta. Las otras operaciones elementales (suma, multiplicación, división) son muy precisas. Siempre que no se presenten desbordamientos y subdesbordamientos, estas operaciones suelen producir resultados que son correctos hasta el último bit. Pero la resta puede ser desde exacta hasta completamente inexacta. Si dos números concuerdan con n cifras, puede perder hasta n cifras de precisión en su resta. Este problema puede aparecer inesperadamente en medio de otros cálculos.

¿Qué pasa con el desbordamiento o con el subdesbordamiento? ¿Cuándo se necesitan números mayores que 10^{308} ? No tan a menudo, pero en los cálculos de probabilidad, por ejemplo, se usan todo el tiempo a menos que se haga un uso inteligente.

Es común en probabilidad calcular un número de tamaño mediano que es el producto de un número astronómicamente grande y un número infinitesimalmente pequeño. El resultado final encaja perfectamente en una computadora, pero es posible que los números intermedios no se deban a un desbordamiento o un subdesbordamiento. Por ejemplo, el número máximo de punto flotante en la mayoría de las computadoras está entre 170 factorial y 171 factorial. Estos grandes factoriales aparecen frecuentemente en aplicaciones, a menudo en proporciones con otros grandes factoriales.

⁸³¿Cuántos dígitos de π necesitamos?

- 3.1415 para diseñar los mejores motores.
- 3.1415926535 para obtener la circunferencia de la Tierra dentro de una fracción de pulgada.
- 3.141592653589793 para los cálculos de navegación interplanetaria de la NASA-JPL.
- 3.1415926535897932384626433832795028842 para medir el radio del universo con una precisión igual al tamaño de un átomo de hidrógeno.

En el 2022 se calcularon los primero 100 billones de decimales del número π , para lograr este hito necesitó 157 días, 23 horas, 31 minutos y 7,651 segundos de cálculos, 515 Terabytes de almacenamiento y desplegar un abanico de tecnologías de computación en Compute Engine, un servicio de computación de Google Cloud.

Anatomía de un Número de Punto Flotante Un número de punto flotante de 64 bits codifica un número de la forma $\pm p \times 2^e$. El primer bit codifica el signo, 0 para números positivos y 1 para números negativos. Los siguientes 11 bits codifican el exponente e , y los últimos 52 bits codifican la precisión p .

El exponente se almacena con un sesgo de 1023. Es decir, los exponentes positivos y negativos se almacenan todos en un solo número positivo almacenando $e + 1023$ en lugar de almacenarlos directamente. Once bits pueden representar números enteros desde 0 hasta 2047. Restando el sesgo, esto corresponde a valores de e de -1023 a $+1024$. Definimos $e_{min} = -1022$ y $e_{max} = +1023$. Los valores $e_{min} - 1$ y $e_{max} + 1$ están reservados para usos especiales.

Los números de punto flotante se almacenan normalmente en forma normalizada. En base 10, un número está en notación científica normalizada si el significando es ≥ 1 y < 10 . Por ejemplo, 3.14×10^2 está en forma normalizada, pero 0.314×10^3 y 31.4×10^2 no lo están.

En general, un número en base β está en forma normalizada si tiene la forma $p \times \beta^e$ donde $1 \leq p < \beta$. Esto dice que para expresar un número binario, es decir, $\beta = 2$, el primer bit del significado de un número normalizado es siempre 1. Dado que este bit nunca cambia, no es necesario almacenarlo. Por lo tanto, podemos expresar 53 bits de precisión en 52 bits de almacenamiento. En lugar de almacenar el significado directamente, almacenamos f , la parte fraccionaria, donde el significado es de la forma $1.f$.

El esquema anterior no explica cómo almacenar 0. Es imposible especificar valores de f y e de modo que $1.f \times 2^e = 0$. El formato de punto flotante hace una excepción a las reglas establecidas anteriormente. Cuando $e = e_{min} - 1$ y $f = 0$, los bits se interpretan como 0. Cuando $e = e_{min} - 1$ y $f \neq 0$, el resultado es un número desnormalizado. Los bits se interpretan como $0.f \times 2^{e_{min}}$. En resumen, el exponente especial reservado debajo de e_{min} se usa para representar 0 y números de punto flotante desnormalizados.

El exponente especial reservado arriba de e_{max} se usa para representar ∞ y NaN . Si $e = e_{max} + 1$ y $f = 0$, los bits se interpretan como ∞ . Pero si $e = e_{max} + 1$ y $f \neq 0$, los bits se interpretan como un NaN o "no es un número" (Not a Number).

Dado que el exponente más grande es 1023 y el significativo más grande es $1.f$ donde f tiene 52 unidades, el número de punto flotante (en C y C++, esta constante se define como `DBL_MAX` definido en `<float.h>`, en Python en `sys.float_info`) más grande es $2^{1023}(2 - 2^{-52}) = 2^{1024} - 2^{971} \approx 2^{1024} \approx$

1.8×10^{308} . Los números mayores que 2^{1024} i.e. $(2 - 2^{52})$ producen un desbordamiento conocido como Overflow.

Dado que el exponente más pequeño es -1022 , el número normalizado positivo más pequeño es $1.0 \times 2^{-1022} \approx 2.2 \times 10^{-308}$ (en C y C++, esta constante se define como *DBL_MIN* definido en *<float.h>*, en Python en *sys.float_info*). Sin embargo, no es el número positivo más pequeño representable como un número de punto flotante, solo el número de punto flotante normalizado más pequeño. Los números más pequeños se pueden expresar en forma desnormalizada, aunque con una pérdida de significado. El número positivo desnormalizado más pequeño ocurre con f que tiene 51 números 0 seguidos de un solo 1. Esto corresponde a $2^{-52} * 2^{-1022} = 2^{-1074} \approx 4.9 \times 10^{-324}$.

Los números que aparecen en los cálculos y tienen magnitud menor que 2^{-1023} i.e. $(1 + 2^{-52})$ producen un desbordamiento de la capacidad mínima o subdesbordamiento también conocido como Underflow y, por lo general se igualan a cero.

C y C++ da el nombre de *DBL_EPSILON* al número positivo más pequeño ϵ tal que $1 + \epsilon \approx 1$, también llamado la precisión de la máquina. Dado que el significativo tiene 52 bits, está claro que $DBL_EPSILON = 2^{-52} \approx 2.2 \times 10^{-16}$. Por eso decimos que un número de punto flotante tiene entre 15 y 16 cifras significativas (decimales).

Formatos de Punto Flotante Se han propuesto varias representaciones diferentes de números reales, pero por mucho la más utilizada es la representación de punto flotante. Las representaciones de punto flotante tienen una base (que siempre se asume que es par) y una precisión p . Si $\beta = 10$ y $p = 3$, entonces el número 0.1 se representa como 1.00×10^{-1} . Si $\beta = 2$ y $p = 24$, entonces el número decimal 0.1 no se puede representar exactamente, pero es aproximadamente $1.10011001100110011001101 \times 2^{-4}$.

En general, un número de punto flotante se representará como $\pm d.dd\dots d \times \beta^e$, donde $d.dd\dots d$ se llama mantisa y tiene p dígitos. De forma más precisa $\pm d_0.d_1d_2\dots d_{p-1} \times \beta^e$ representa el número

$$\pm (d_0 + d_1\beta^{-1} + \dots + d_{p-1}\beta^{-(p-1)})\beta^e, (0 \leq d_i < \beta). \quad (23.1)$$

El término número de punto flotante se utilizará para referirse a un número real que se puede representar exactamente en el formato en discusión. Otros dos parámetros asociados con las representaciones de punto

flotante son los exponentes más grandes y más pequeños permitidos, e_{max} y e_{min} . Dado que hay β^p posibles significados, y $e_{max} - e_{min} + 1$ posibles exponentes, un número de punto flotante se puede codificar en

$$[\log_2 (e_{max} - e_{min} + 1)] + [\log_2 (\beta^p)] + 1$$

bits, donde el +1 final es para el bit de signo. La codificación precisa no es importante por ahora.

Hay dos razones por las que un número real podría no ser exactamente representable como un número de punto flotante. La situación más común se ilustra con el número decimal 0.1. Aunque tiene una representación decimal finita, en binario tiene una representación repetida infinita. Por lo tanto, cuando $\beta = 2$, el número 0.1 se encuentra estrictamente entre dos números de punto flotante y ninguno de ellos lo puede representar exactamente.

Una situación menos común es que un número real esté fuera de rango, es decir, su valor absoluto sea mayor que $\beta \times \beta^{e_{max}}$ o menor que $1.0 \times \beta^{e_{min}}$. La mayor parte de esta sección analiza cuestiones debidas a la primera razón.

Las representaciones de punto flotante no son necesariamente únicas. Por ejemplo, tanto 0.01×10^1 como 1.00×10^{-1} representan 0.1. Si el dígito inicial es distinto de cero ($d_0 \neq 0$ en la Ec. (23.1)), se dice que la representación está normalizada. El número de punto flotante 1.00×10^{-1} está normalizado, mientras que 0.01×10^1 no lo está.

¿Cuándo es Exacta la Conversión de Base de Punto Flotante de Ida y Vuelta? Suponga que almacenamos un número de punto flotante en la memoria, lo imprimimos en base 10 legible por humanos y lo regresamos a memoria. ¿Cuándo se puede recuperar exactamente el número original?

Suponga que comenzamos con la base β con p lugares de precisión y convertimos a la base γ con q lugares de precisión, redondeando al más cercano, luego volvemos a convertir a la base original β . El teorema de Matula dice que si no hay enteros positivos i y j tales que

$$\beta^i = \gamma^j$$

entonces una condición necesaria y suficiente para que la conversión de ida y vuelta sea exacta (suponiendo que no haya desbordamiento o subdesbordamiento) es que

$$\gamma^{q-1} > \beta^p.$$

En el caso de números de punto flotante (por ejemplo doble en C) tenemos $\beta = 2$ y $p = 53$. (ver Anatomía de un Número de Punto Flotante). Estamos imprimiendo a base $\gamma = 10$. Ninguna potencia positiva de 10 también es una potencia de 2, por lo que se mantiene la condición de Matula en las dos bases.

Si imprimimos $q = 17$ decimales, entonces

$$10^{16} > 2^{53}$$

por lo que la conversión de ida y vuelta será exacta si ambas conversiones se redondean al más cercano. Si q es menor, algunas conversiones de ida y vuelta no serán exactas.

También puede verificar que para un número de punto flotante de precisión simple ($p =$ precisión de 24 bits) necesita $q = 9$ dígitos decimales, y para un número de precisión cuádruple (precisión de $p = 113$ bits) necesita $q = 36$ dígitos decimales⁸⁴.

Mirando hacia atrás en el teorema de Matula, claramente necesitamos

$$\gamma^q \geq \beta^p.$$

¿Por qué? Porque el lado derecho es el número de fracciones de base β y el lado izquierdo es el número de fracciones de base γ . No puede tener un mapa uno a uno de un espacio más grande a un espacio más pequeño. Entonces, la desigualdad anterior es necesaria, pero no suficiente. Sin embargo, es casi suficiente. Solo necesitamos una base γ con más cifras significativas, es decir, Matula nos dice

$$\gamma^{q-1} > \beta^p$$

es suficiente. En términos de base 2 y base 10, necesitamos al menos 16 decimales para representar 53 bits. Lo sorprendente es que un decimal más es suficiente para garantizar que las conversiones de ida y vuelta sean exactas. No es obvio a priori que cualquier número finito de decimales adicionales sea siempre suficiente, pero de hecho solo uno más es suficiente.

⁸⁴El número de bits asignados para la parte fraccionaria de un número de punto flotante es 1 menos que la precisión: la cifra inicial es siempre 1, por lo que los formatos IEEE ahorran un bit al no almacenar el bit inicial, dejándolo implícito. Entonces, por ejemplo, un doble en C tiene una precisión de 53 bits, pero 52 bits de los 64 bits en un doble se asignan para almacenar la fracción.

A continuación, se muestra un ejemplo para mostrar que el decimal adicional es necesario. Suponga que $p = 5$. Hay más números de 2 dígitos que números de 5 bits, pero si solo usamos dos dígitos, la conversión de base de ida y vuelta no siempre será exacta. Por ejemplo, el número $17/16$ escrito en binario es 1.0001_{dos} y tiene cinco bits significativos. El equivalente decimal es 1.0625_{diez} , que redondeado a dos dígitos significativos es 1.1_{diez} . Pero el número binario más cercano a 1.1_{diez} con 5 bits significativos es $1.0010_{dos} = 1.125_{diez}$. En resumen, redondeando al más cercano da

$$1.0001_{dos} - > 1.1_{diez} - > 1.0010_{dos}$$

y así no volvemos al punto de partida.

Error de Representación se refiere al hecho de que la mayoría de las fracciones decimales no pueden representarse exactamente como fracciones binarias (en base 2). Esta es la razón principal de por qué muchos lenguajes de programación (Python, Perl, C, C++, Java, Fortran, y tantos otros) frecuentemente no mostrarán el número decimal exacto que esperas.

¿Por qué es eso? $1/10$ no es representable exactamente como una fracción binaria. Casi todas las máquinas de hoy en día usan aritmética de punto flotante: IEEE-754, y casi todas las plataformas mapean los flotantes al «doble precisión» de IEEE-754. Estos «dobles» tienen 53 bits de precisión, por lo tanto en la entrada la computadora intenta convertir 0.1 a la fracción más cercana que puede de la forma $J/(2^N)$ donde J es un entero que contiene exactamente 53 bits. Reescribiendo

$$1/10 \approx J/2^N$$

como

$$J \approx 2^N/10$$

y recordando que J tiene exactamente 53 bits (es $\geq 2^{52}$ pero $< 2^{53}$), el mejor valor para N es 56. O sea, 56 es el único valor para N que deja J con exactamente 53 bits. El mejor valor posible para J es entonces el cociente redondeado, si usamos Python para los cálculos tenemos

```
>>> q, r = divmod(2**56, 10)
>>> r
6
```


Error de Redondeo Comprimir infinitos números reales en un número finito de bits requiere una representación aproximada. Aunque hay un número infinito de enteros, en la mayoría de los programas el resultado de los cálculos de números enteros se puede almacenar en 32 bits o 64 bits. Por el contrario, dado cualquier número fijo de bits, la mayoría de los cálculos con números reales producirán cantidades que no se pueden representar exactamente usando tantos bits. Por lo tanto, el resultado de un cálculo de punto flotante a menudo debe redondearse para volver a ajustarse a su representación finita. Este error de redondeo es el rasgo característico del cálculo de punto flotante.

Dado que la mayoría de los cálculos de punto flotante tienen errores de redondeo de todos modos, ¿importa si las operaciones aritméticas básicas introducen un poco más de error de redondeo de lo necesario? Esa pregunta es un tema principal a lo largo de esta sección. La sección Dígitos de Guarda analiza los dígitos de protección, un medio para reducir el error al restar dos números cercanos. IBM consideró que los dígitos de guarda eran lo suficientemente importantes que en 1968 añadió un dígito de guarda al formato de doble precisión en la arquitectura System / 360 (la precisión simple ya tenía un dígito de guarda) y modernizó todas las máquinas existentes en el campo.

El estándar IEEE va más allá de sólo requerir el uso de un dígito de protección. Proporciona un algoritmo para la suma, resta, multiplicación, división y raíz cuadrada y requiere que las implementaciones produzcan el mismo resultado que ese algoritmo. Por lo tanto, cuando un programa se mueve de una máquina a otra, los resultados de las operaciones básicas serán los mismos en todos los bits si ambas máquinas admiten el estándar IEEE. Esto simplifica enormemente la portabilidad de programas. Otros usos de esta especificación precisa se dan en operaciones exactamente redondeadas.

Error Relativo y Ulp Dado que el error de redondeo es inherente al cálculo de punto flotante, es importante tener una forma de medir este error. Considere el formato de punto flotante con $\beta = 10$ y $p = 3$, que se utilizará en esta sección. Si el resultado de un cálculo de punto flotante es 3.12×10^{-2} , y la respuesta cuando se calcula con precisión infinita es 0.0314, está claro que tiene un error de 2 unidades en el último lugar. De manera similar, si el número real 0.0314159 se representa como 3.14×10^{-2} , entonces tiene un error de 0.159 unidades en el último lugar.

En general, si el número de punto flotante $dd..d \times \beta^e$ se usa para representar z , entonces tiene un error de $|d.d\dots d - (z/\beta^e)|^{\beta^{p-1}}$ unidades en el

último lugar. El término ulps se utilizará como abreviatura de (units in the last place) "unidades en último lugar". Si el resultado de un cálculo es el número de punto flotante más cercano al resultado correcto, aún podría tener un error de hasta 0.5 ulp.

Otra forma de medir la diferencia entre un número de punto flotante y el número real al que se aproxima es el error relativo, que es simplemente la diferencia entre los dos números divididos por el número real. Por ejemplo, el error relativo cometido al aproximar 3.14159 por 3.14×10^0 es $0.00159/3.141590 \approx 0005$.

Para calcular el error relativo que corresponde a .5 ulp, observe que cuando un número real es aproximado por el número de punto flotante

más cercano posible $\overbrace{d.dd\dots dd}^p \times \beta^e$, el error puede ser tan grande como $\overbrace{0.00\dots 00\beta'}^p \times \beta^e$, donde β' es el dígito $\beta/2$, hay p unidades en el significado del número de punto flotante y p unidades de 0 en el significado del error. Este error es $((\beta/2)\beta^{-p}) \times \beta^e$. Dado que los números de la forma $d.dd\dots dd \times \beta^e$ tienen todos el mismo error absoluto, pero tienen valores que oscilan entre β^e y $\beta \times \beta^e$, el error relativo varía entre $((\beta/2)\beta^{-p}) \times \beta^e/\beta^e$ y $((\beta/2)\beta^{-p}) \times \beta^e/\beta^{e+1}$. Eso es,

$$\frac{1}{2}\beta^{-p} \leq \frac{1}{2}ulp \leq \frac{\beta}{2}\beta^{-p} \quad (23.2)$$

En particular, el error relativo correspondiente a 0.5 ulp puede variar en un factor de β . Este factor se llama bamboleo. Estableciendo $\epsilon = (\beta/2)\beta^{-p}$ en el mayor de los límites en Ec. (23.2) anterior, podemos decir que cuando un número real se redondea al número de punto flotante más cercano, el error relativo siempre está limitado por ϵ , que se conoce como épsilon de la máquina.

En el ejemplo anterior, el error relativo fue $0.00159/3.14159 \approx 0005$. Para evitar números tan pequeños, el error relativo normalmente se escribe como factor multiplicado ϵ , que en este caso es $\epsilon = (\beta/2)\beta^{-p} = 5(10)^{-3} = 0.005$. Por lo tanto, el error relativo se expresaría como $(0.00159/3.14159)/0.005\epsilon \approx 0.1\epsilon$.

Para ilustrar la diferencia entre ulps y error relativo, considere el número real $x = 12.35$. Se aproxima por $\tilde{x} = 1.24 \times 10^1$. El error es 0.5 ulps, el error relativo es 0.8ϵ . A continuación, considere el cálculo $8\tilde{x}$. El valor exacto es $8x = 98.8$, mientras que el valor calculado es $8\tilde{x} = 9.92 \times 10^1$. El error ahora

es 4.0 ulps, pero el error relativo sigue siendo 0.8ϵ . El error medido en ulps es 8 veces mayor, aunque el error relativo es el mismo.

En general, cuando la base es β , un error relativo fijo expresado en ulps puede oscilar en un factor de hasta β . Y a la inversa, como muestra la Ec. (23.2) anterior, un error fijo de 0.5 ulps da como resultado un error relativo que puede oscilar por β .

La forma más natural de medir el error de redondeo es en ulps. Por ejemplo, el redondeo al número de punto flotante más cercano corresponde a un error menor o igual a 0.5 ulp. Sin embargo, al analizar el error de redondeo causado por varias fórmulas, el error relativo es una mejor medida. Dado que se puede sobrestimar el efecto de redondear al número de punto flotante más cercano por el factor de oscilación de β , las estimaciones de error de las fórmulas serán más estrictas en máquinas con una pequeña β .

Cuando sólo interesa el orden de magnitud del error de redondeo, ulps y ϵ pueden usarse indistintamente, ya que difieren como máximo en un factor de β . Por ejemplo, cuando un número de punto flotante tiene un error de n ulps, eso significa que el número de dígitos contaminados es $\log_{\beta} n$. Si el error relativo en un cálculo es $n\epsilon$, entonces

$$\text{los dígitos contaminados} \approx \log_{\beta} n. \quad (23.3)$$

Dígito de Guarda Un método para calcular la diferencia entre dos números de punto flotante es calcular la diferencia exactamente y luego redondearla al número de punto flotante más cercano. Esto es muy caro si los operandos difieren mucho en tamaño. Suponiendo que $p = 3$, $2.15 \times 10^{12} - 1.25 \times 10^{-5}$ se calcularía como

$$\begin{aligned} x &= 2.15 \times 10^{12} \\ y &= 0.0000000000000000125 \times 10^{12} \\ x - y &= 2.1499999999999999875 \times 10^{12} \end{aligned}$$

que se redondea a 2.15×10^{12} . En lugar de utilizar todos estos dígitos, el Hardware de punto flotante normalmente funciona con un número fijo de dígitos. Suponga que el número de dígitos que se mantiene es p , y que cuando el operando más pequeño se desplaza hacia la derecha, los dígitos simplemente se descartan (en contraposición al redondeo). Entonces $2.15 \times 10^{12} - 1.25 \times 10^{-5}$ se convierte en

$$\begin{aligned} x &= 2.15 \times 10^{12} \\ y &= 0.00 \times 10^{12} \\ x - y &= 2.15 \times 10^{12} \end{aligned}$$

La respuesta es exactamente la misma que si la diferencia se hubiera calculado exactamente y luego se hubiera redondeado. Tome otro ejemplo: $10.1 - 9.93$. Esto se convierte en

$$\begin{aligned}x &= 1.01 \times 10^1 \\y &= 0 - 99 \times 10^1 \\x - y &= 0.02 \times 10^1\end{aligned}$$

La respuesta correcta es 0.17, por lo que la diferencia calculada está desviada en 30 ulps y es incorrecta en todos los dígitos. ¿Qué tan grave puede ser el error?.

Teorema 127 *Usando un formato de punto flotante con parámetros β y p , y calculando las diferencias usando p dígitos, el error relativo del resultado puede ser tan grande como $\beta - 1$.*

Cuando $\beta = 2$, el error relativo puede ser tan grande como el resultado, y cuando $\beta = 10$, puede ser 9 veces mayor. O para decirlo de otra manera, cuando $\beta = 2$, la Ec. (23.3) muestra que el número de dígitos contaminados es $\log_2(1/\epsilon) = \log_2(2^p) = p$. Es decir, ¡todos los dígitos p del resultado son incorrectos!. Suponga que se agrega un dígito adicional para protegerse contra esta situación (un dígito de guardia). Es decir, el número más pequeño se trunca a $p+1$ dígitos, y luego el resultado de la resta se redondea a p dígitos. Con un dígito de guarda, el ejemplo anterior se convierte en

$$\begin{aligned}x &= 1.010 \times 10^1 \\y &= 0.993 \times 10^1 \\x - y &= 0.017 \times 10^1\end{aligned}$$

y la respuesta es exacta. Con un solo dígito de guarda, el error relativo del resultado puede ser mayor que ϵ , como en $110 - 8.59$.

$$\begin{aligned}x &= 1.10 \times 10^2 \\y &= 0.085 \times 10^2 \\x - y &= 1.015 \times 10^2\end{aligned}$$

Esto se redondea a 102, en comparación con la respuesta correcta de 101.41, para un error relativo de 0.006, que es mayor que $\epsilon = 0.005$. En general, el error relativo del resultado puede ser solo un poco mayor que ϵ . De forma más precisa:

Teorema 128 *Si x y y son números de punto flotante en un formato con parámetros β y p , y si la resta se realiza con $p+1$ dígitos (es decir, un dígito de guarda), entonces el error de redondeo relativo en el resultado es menor que 2ϵ .*

Cancelación La última sección se puede resumir diciendo que sin un dígito de guarda, el error relativo cometido al restar dos cantidades cercanas puede ser muy grande. En otras palabras, la evaluación de cualquier expresión que contenga una resta (o una suma de cantidades con signos opuestos) podría resultar en un error relativo tan grande que todos los dígitos carecen de significado (Teorema (127)). Al restar cantidades cercanas, los dígitos más significativos de los operandos coinciden y se cancelan entre sí. Hay dos tipos de cancelación: catastrófica y benigna.

La cancelación catastrófica ocurre cuando los operandos están sujetos a errores de redondeo. Por ejemplo, en la fórmula cuadrática, aparece la expresión $b^2 - 4ac$. Las cantidades b^2 y $4ac$ están sujetas a errores de redondeo ya que son el resultado de multiplicaciones de punto flotante. Suponga que están redondeados al número de punto flotante más cercano y, por lo tanto, tienen una precisión de 0.5 ulp. Cuando se restan, la cancelación puede hacer que muchos de los dígitos precisos desaparezcan, dejando principalmente dígitos contaminados por errores de redondeo. Por tanto, la diferencia puede tener un error de muchos ulps. Por ejemplo, considere $b = 3.34$, $a = 1.22$ y $c = 2.28$. El valor exacto de $b^2 - 4ac$ es 0.0292. Pero b^2 se redondea a 11.2 y $4ac$ se redondea a 11.1, por lo que la respuesta final es 0.1, que es un error de 70 ulps, aunque $11.2 - 11.1$ es exactamente igual a 0.1. La resta no introdujo ningún error, sino que expuso el error introducido en las multiplicaciones anteriores.

La cancelación benigna ocurre al restar cantidades exactamente conocidas. Si x e y no tienen error de redondeo, entonces, según el Teorema (128), si la resta se realiza con un dígito de guarda, la diferencia $x - y$ tiene un error relativo muy pequeño (menos de 2ϵ).

Una fórmula que presenta una cancelación catastrófica a veces se puede reorganizar para eliminar el problema. Considere nuevamente la fórmula cuadrática

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \text{ y } x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \quad (23.4)$$

Cuando $b^2 \gg 4ac$, entonces $b^2 - 4ac$ no implica una cancelación y

$$\sqrt{b^2 - 4ac} \approx |b|.$$

Pero la otra suma (resta) en una de las fórmulas tendrá una cancelación catastrófica. Para evitar esto, multiplique el numerador y denominador de

x_1 por $-b - \sqrt{b^2 - 4ac}$ (y de manera similar para x_2) para obtener

$$x_1 = \frac{-2c}{b + \sqrt{b^2 - 4ac}} \text{ y } x_2 = \frac{-2c}{b - \sqrt{b^2 - 4ac}} \quad (23.5)$$

Si $b^2 \gg ac$ y $b > 0$, entonces calcular x_1 usando la Ec. (23.4) implicará una cancelación. Por lo tanto, use la Ec. (23.5) para calcular x_1 y (23.4) para x_2 . Por otro lado, si $b < 0$, use (23.4) para calcular x_1 y (23.5) para x_2 .

La expresión $x^2 - y^2$ es otra fórmula que presenta una cancelación catastrófica. Es más exacto evaluarlo como

$$(x - y)(x + y)$$

A diferencia de la fórmula cuadrática, esta forma mejorada todavía tiene una resta, pero es una cancelación benigna de cantidades sin error de redondeo, no catastrófica. Según el Teorema (128), el error relativo en $x - y$ es como máximo 2ϵ . Lo mismo ocurre con $x + y$. Multiplicar dos cantidades con un pequeño error relativo da como resultado un producto con un pequeño error relativo.

23.1 Errores de Redondeo y de Aritmética

La aritmética que realiza una computadora es distinta de la aritmética de nuestros cursos de álgebra o cálculo. En nuestro mundo matemático tradicional consideramos la existencia de números con una cantidad infinita de cifras, en la computadora cada número representable tienen sólo un número finito, fijo de cifras, los cuales en la mayoría de los casos es satisfactoria y se aprueba sin más, aunque a veces esta discrepancia puede generar problemas.

Un ejemplo de este hecho lo tenemos en el cálculo de raíces de:

$$ax^2 + bx + c = 0$$

cuando $a \neq 0$, donde las raíces se calculan comúnmente con el algoritmo Ec. (23.4) o de forma alternativa con el algoritmo que se obtiene mediante la racionalización del numerador Ec. (23.5).

Otro algoritmo que podemos implementar es el método de Newton-Raphson⁸⁵ para buscar raíces, que en su forma iterativa está dado por

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

en el cual se usa x_0 como una primera aproximación a la raíz buscada y x_n es la aproximación a la raíz después de n iteraciones (sí se converge a ella), donde $f(x) = ax^2 + bx + c$ y $f'(x_i) = 2ax + b$.

Salida del Cálculo de Raíces Para resolver el problema, usamos por ejemplo el siguiente código en Python usando programación procedimental:

```
import math
def f(x, a, b, c):
    """ Evalua la Funcion cuadratica """
    return (x * x * a + x * b + c);
def df(x, a, b):
    """ Evalua la derivada de la funcion cuadratica """
    return (2.0 * x * a + b);
def evalua(x, a, b, c):
    """ Evalua el valor X en la función cuadratica """
    print('Raiz (%1.16f), evaluacion raiz: %1.16e' % (x, f(x, a, b, c)))
def metodoNewtonRapson(x, ni, a, b, c):
    """ Metodo Newton-Raphson x = x - f(x)/f'(x) """
    for i in range(ni):
        x = x - (f(x, a, b, c) / df(x, a, b))
    return x
def raices(A, B, C):
    """ Calculo de raices """
    if A == 0.0:
        print("No es una ecuacion cuadratica")
        exit(1)
```

⁸⁵También podemos usar otros métodos, como el de Newton Raphson Modificado para acelerar la convergencia

$$x_{i+1} = x_i - \frac{f(x_i)f'(x_i)}{[f'(x_i)]^2 - f(x_i)f''(x_i)}$$

que involucra la función $f(x)$, la primera derivada $f'(x)$ y a la segunda derivada $f''(x)$.

```

# Calculo del discriminante
d = B * B - 4.0 * A * C
# Raices reales
if d >= 0.0:
    print('\nPolinomio (%f) X^2 + (%f)X + (%f) = 0\n' % (A, B, C))
    print('\nChicharronera 1')
    X1 = (-B + math.sqrt(d)) / (2.0 * A)
    X2 = (-B - math.sqrt(d)) / (2.0 * A)
    evalua(X1, A, B, C)
    evalua(X2, A, B, C)
    print('\nChicharronera 2')
    X1 = (-2.0 * C) / (B + math.sqrt(d))
    X2 = (-2.0 * C) / (B - math.sqrt(d))
    evalua(X1, A, B, C)
    evalua(X2, A, B, C)
    # Metodo Newton-Raphson
    print("\n\nMetodo Newton-Raphson")
    x = X1 - 1.0;
    print("\nValor inicial aproximado de X1 = %1.16f" % x)
    x = metodoNewtonRapson(x, 6, A, B, C)
    evalua(x, A, B, C);
    x = X2 - 1.0;
    print("\nValor inicial aproximado de X2 = %1.16f" % x);
    x = metodoNewtonRapson(x, 6, A, B, C)
    evalua(x, A, B, C)
    print("\n\n")
else:
    # Raices complejas
    print("Raices Complejas ...")
if __name__ == '__main__':
    raices(1.0, 4.0, 1.0)

```

y generan la siguiente salida:

Polinomio (1.000000) X² + (4.000000)X + (1.000000) = 0

Chicharronera 1

Raiz (-0.2679491924311228), evaluacion raiz: -4.4408920985006262e-16

Raiz (-3.7320508075688772), evaluacion raiz: 0.0000000000000000e+00

Chicharronera 2

Raiz (-0.2679491924311227), evaluacion raiz: 0.0000000000000000e+00

Raiz (-3.7320508075688759), evaluacion raiz: -5.3290705182007514e-15

Metodo Newton-Raphson

Valor inicial aproximado de X1 = -1.2679491924311228

Raiz (-0.2679491924311227), evaluacion raiz: 0.0000000000000000e+00

Valor inicial aproximado de X2 = -4.7320508075688759

Raiz (-3.7320508075688772), evaluacion raiz: 0.0000000000000000e+00

En esta salida se muestra la raíz calculada y su evaluación en la ecuación cuadrática, la cual debería de ser cero al ser una raíz, pero esto no ocurre en general por los errores de redondeo. Además, nótese el impacto de seleccionar el algoritmo numérico adecuado a los objetivos que persigamos en la solución del problema planteado.

En cuanto a la implementación computacional, el paradigma de programación seleccionado depende la complejidad del algoritmo a implementar y si necesitamos reusar el código generado o no. Otras implementaciones computacionales se pueden consultar en las ligas, en las cuales se usan distintos lenguajes ([C](#), [C++](#), [Java](#) y [Python](#)) y diferentes paradigmas de programación ([secuencial](#), [procedimental](#) y [orientada a objetos](#)).

Si lo que necesitamos implementar computacionalmente es una fórmula o conjunto de ellas que generen un código de decenas de líneas, la implementación secuencial es suficiente, si es menor a una centena de líneas puede ser mejor opción la implementación procedimental y si el proyecto es grande o complejo, seguramente se optará por la programación orientada a objetos o formulaciones híbridas de las anteriores.

En última instancia, lo que se persigue en la programación es generar un código: correcto, claro, eficiente, de fácil uso y mantenimiento, que sea flexible, reusable y en su caso portable.

23.2 Trabajando con Punto Flotante

Hay varias trampas en las que incluso los programadores muy experimentados caen cuando escriben código que depende de la aritmética de punto flotante. En esta sección explicamos algunas cosas a tener en cuenta al trabajar con

que está cerca pero no es exactamente el valor verdadero de $1/10$.

La mayoría de los usuarios no son conscientes de esta aproximación por la forma en que se muestran los valores. Varios lenguajes de programación como C, C++, Java y Python solamente muestra una aproximación decimal al valor verdadero decimal de la aproximación binaria almacenada por la máquina. En la mayoría de las máquinas, si fuéramos a imprimir el verdadero valor decimal de la aproximación binaria almacenada para 0.1, debería mostrar

0.1000000000000000055511151231257827021181583404541015625

esos son más dígitos que lo que la mayoría de la gente encuentra útil, por lo que los lenguajes de programación mantiene manejable la cantidad de dígitos al mostrar en su lugar un valor redondeado

$1/10$

como

0.1

sólo hay que recordar que, a pesar de que el valor mostrado resulta ser exactamente $1/10$, el valor almacenado realmente es la fracción binaria más cercana posible. Esto queda de manifiesto cuando hacemos

$0.1 + 0.1 + 0.1$ ó $0.1 * 3$ ó $0.1 + 0.2$ ⁸⁶

obtendremos

0.30000000000000004

que es distinto al 0.3 que esperábamos.

Es de hacer notar que hay varios números decimales que comparten la misma fracción binaria más aproximada. Por ejemplo, los números

0.1, 0.10000000000000001 y

⁸⁶Para el caso de $0.1 + 0.2$ podemos hacer un programa que nos de 0.3, usando:

```
double x = 0.1;
double y = 0.1;
double z = (x*10.0 + y*10.0) / 10.0
```

```
0.10000000000000000055511151231257827021181583404541015625
```

son todos aproximados por $3602879701896397/2^{55}$.

Notemos que esta es la verdadera naturaleza del punto flotante binario: no es un error del lenguaje de programación y tampoco es un error en tu código. Verás lo mismo en todos los lenguajes que soportan la aritmética de punto flotante de tu Hardware (a pesar de que en algunos lenguajes por omisión no muestren la diferencia, o no lo hagan en todos los modos de salida). Para una salida más elegante, quizás quieras usar el formateo de cadenas de texto para generar un número limitado de dígitos significativos.

Ejemplo, el número decimal 9.2 se puede expresar exactamente como una relación de dos enteros decimales $92/10$, los cuales se pueden expresar exactamente en binario como $0b1011100/0b1010$. Sin embargo, la misma proporción almacenada como un número de punto flotante nunca es exactamente igual a 9.2:

```
usando 32 bits obtenemos: 9.1999999809265136
usando 64 bits obtenemos: 9.19999999999999928945
```

ya que se guarda como la fracción: $5179139571476070/2^{49}$.

Otro ejemplo, si tomamos el caso del número 0.02 y vemos su representación en el lenguaje de programación Python, tenemos que:

```
import decimal
print(decimal.Decimal(0.02))
```

y el resultado es:

```
0.0200000000000000004163336342344337026588618755340576171875
```

Además, tenemos la no representabilidad de π (y $\pi/2$), esto significa que un intento de cálculo de $\tan(\pi/2)$ no producirá un resultado de infinito, ni siquiera se desbordará en los formatos habituales de punto flotante. Simplemente no es posible que el Hardware de punto flotante estándar intente calcular $\tan(\pi/2)$, porque $\pi/2$ no se puede representar exactamente. Este cálculo en C:

```
doble pi =3.1415926535897932384626433832795;
tan (pi / 2.0);
```

dará un resultado de $1.633123935319537e + 16$. En precisión simple usando tanf, el resultado será -22877332.0 .

De la misma manera, un intento de cálculo de $\sin(\pi)$ no arrojará cero. El resultado será $1.2246467991473532e - 16$ en precisión doble.

Si bien la suma y la multiplicación de punto flotante son conmutativas ($a + b = b + a$ y $a \times b = b \times a$), no son necesariamente asociativas. Es decir, $(a + b) + c$ no es necesariamente igual a $a + (b + c)$. Usando aritmética decimal significativa de 7 dígitos:

$$a = 1234.567, b = 45.67834, c = 0.0004$$

$(a + b) + c$:

$$\begin{array}{ll} 1234.567 + 45.67834 & 1280.24534 \text{ se redondea a } 1280.245 \\ 1280.245 + 0.0004 & 1280.2454 \text{ se redondea a } 1280.245 \end{array}$$

$a + (b + c)$:

$$\begin{array}{ll} 45.67834 + 0.0004 & 45.67874 \\ 1234.567 + 45.6787 & 1280.24574 \text{ se redondea a } 1280.246 \end{array}$$

Tampoco son necesariamente distributivos. Es decir $(a + b) \times c$ puede no ser lo mismo que $a \times c + b \times c$:

$$\begin{array}{l} 1234.567 * 3.333333 = 4115.223, 1.234567 * 3.333333 = 4.115223, \\ 4115.223 + 4.115223 = 4119.338 \end{array}$$

pero

$$1234.567 + 1.234567 = 1235.802, 1235.802 * 3.333333 = 4119.340$$

Algunos Trucos Supongamos que necesitamos hacer el siguiente cálculo:

$$4.56 * 100$$

la respuesta es:

$$455.9999999999994$$

que no es la esperada. ¿Qué podemos hacer para obtener lo que esperamos?. Por ejemplo, con Python, podemos usar:

```
from sympy import *
print(nsimpify(4.56 * 100, tolerance=1e-1))
```

que nos dará el 456 esperado.

SymPy es un paquete matemático simbólico para Python, y `nsimpify` toma un número de punto flotante y trata de simplificarlo como una fracción con un denominador pequeño, la raíz cuadrada de un entero pequeño, una expresión que involucra constantes famosas, etc.

Por ejemplo, supongamos que algún cálculo arrojó 4.242640687119286 y sospechamos que hay algo especial en ese número. Así es como puede probar de dónde vino:

```
from sympy import *
print(nsimpify(4.242640687119286))
```

que nos arrojará:

$$3 * \sqrt{2}$$

Tal vez hagamos un cálculo numérico y encontremos una expresión simple para el resultado y eso sugiera una solución analítica. Creo que una aplicación más común de `nsimpify` podría ser ayudarte a recordar fórmulas medio olvidadas. Por ejemplo, quizás estés oxidado con tus identidades trigonométricas, pero recuerdas que $\cos(\pi/6)$ es algo especial.

```
from sympy import *
print(nsimpify(cos(pi/6)))
```

que nos entregará:

$$\sqrt{3}/2$$

O, para tomar un ejemplo más avanzado, supongamos que recordamos vagamente que la función gamma toma valores reconocibles en valores semienteros, pero no recordamos exactamente cómo. Tal vez algo relacionado con π o e . Puede sugerir que `nsimpify` incluya expresiones con π y e en su búsqueda.

```
from sympy import *
print(nsimpify(gamma(3.5), constants=[pi, E]))
```

obteniendo:

$$15 * \sqrt{\pi}/8$$

También podemos darle a `nsimplify` una tolerancia, pidiéndole que encuentre una representación simple dentro de una vecindad del número. Por ejemplo, aquí hay una manera de encontrar aproximaciones a π .

```
from sympy import *  
print(nsimplify(pi, tolerance=1e-5))
```

obteniendo:

$$355/113$$

con una tolerancia más amplia, devolverá una aproximación más simple.

```
from sympy import *  
print(nsimplify(pi, tolerance=1e-2))
```

obteniendo:

$$22/7$$

finalmente, aquí hay una aproximación de mayor precisión a π que no es exactamente simple:

```
from sympy import *  
print(nsimplify(pi, tolerance=1e-7))
```

obteniendo:

$$\exp(141/895 + \sqrt{780631})/895$$

No Pruebas por la Igualdad Cuando se usa Punto Flotante no es recomendable escribir código como el siguiente⁸⁷:

```
double x;
double y;
...
if (x == y) {...}
```

La mayoría de las operaciones de punto flotante implican al menos una pequeña pérdida de precisión y, por lo tanto, incluso si dos números son iguales para todos los fines prácticos, es posible que no sean exactamente iguales hasta el último bit, por lo que es probable que la prueba de igualdad falle. Por ejemplo:

```
double x = 10;
double y = sqrt(x);
y *= y;
if (x == y)
cout << "La raiz cuadrada es exacta\n";
else
cout << x-y << "\n";
```

el código imprime: $-1.778636e-015$, aunque en teoría, elevar al cuadrado debería deshacer una raíz cuadrada, la operación de ida y vuelta es ligeramente inexacta. En la mayoría de los casos, la prueba de igualdad anterior debe escribirse de la siguiente manera:

```
double tolerancia = ...
if (fabs(x - y) < tolerancia) {...}
```

Aquí la tolerancia es un umbral que define lo que está "lo suficientemente cerca" para la igualdad. Esto plantea la pregunta de qué tan cerca está lo suficientemente cerca. Esto no puede responderse en abstracto; tienes que saber algo sobre tu problema particular para saber qué tan cerca está lo suficientemente cerca en tu contexto.

⁸⁷Sin pérdida de generalidad usamos algún lenguaje de programación en particular para mostrar los ejemplos, pero esto pasa en todos los lenguajes que usan operaciones de Punto Flotante.

Por ejemplo: ¿hay alguna garantía de que la raíz cuadrada de un cuadrado perfecto sea devuelta exactamente?, por ejemplo si hago `sqrt(81.0) == 9.0`, por lo visto anteriormente, la respuesta es no, pero podríamos cambiar la pregunta por `9.0 * 9.0 == 81.0`, esto funcionará siempre que el cuadrado esté dentro de los límites de la magnitud del punto flotante.

Por otro lado, es posible que las expectativas de las matemáticas no se cumplan en el campo del cálculo de punto flotante. Por ejemplo, se sabe que

$$(x + y)(x - y) = x^2 - y^2$$

y esta otra

$$\sin^2 \theta + \cos^2 \theta = 1$$

sin embargo, no se puede confiar en estos hechos cuando las cantidades involucradas son el resultado de un cálculo de punto flotante. Además, el error de redondeo puede afectar la convergencia y precisión de los procedimientos numéricos iterativos.

Aún más y sin pérdida de generalidad, si comparamos usando el lenguaje de programación Python:

```
print(0.1 + 0.2 == 0.3)
print(0.2 + 0.2 + 0.2 == 0.6)
print(1.2 + 2.4 + 3.6 == 7.2)
print(0.1 + 0.2 <= 0.3)
```

en todos los casos dará un resultado de falso, además:

```
print(10.4 + 20.8 > 31.2)
print(0.8 - 0.1 > 0.7)
```

el resultado será verdadero. Por ello es siempre conveniente ver con que números trabajamos usando algo como:

```
print(format(0.1, ".17g"))
print(format(0.2, ".17g"))
print(format(0.3, ".17g"))
```

así cuando sumemos `0.1 + 0.2`, podemos ver el verdadero resultado:

```
print(print(format(0.1 + 0.2, ".17g")))
```

Teniendo esto en cuenta podemos comparar usando:

```
import math
print(math.isclose(0.1 + 0.2, 0.3))
```

nos dará la respuesta esperada. Podemos ajustar la tolerancia relativa usando:

```
import math
print(math.isclose(0.1 + 0.2, 0.3, rel_tol = 1e-20))
```

que en este caso nos dirá que es falso pues no son iguales en los 20 primeros dígitos solicitados.

Para las comparaciones \geq o \leq , por ejemplo para $a + b \leq c$ se debe usar:

```
a, b, c = 0.1, 0.2, 0.3
print(math.isclose(a + b, c) or (a + c < c))
```

Preocúpate más por la suma y la resta que por la multiplicación y la división Los errores relativos en la multiplicación y división son siempre pequeños. La suma y la resta, por otro lado, pueden resultar en una pérdida completa de precisión. Realmente el problema es la resta; la suma sólo puede ser un problema cuando los dos números que se agregan tienen signos opuestos, por lo que puedes pensar en eso como una resta. Aún así, el código podría escribirse con un "+" que sea realmente una resta.

La resta es un problema cuando los dos números que se restan son casi iguales. Cuanto más casi iguales sean los números, mayor será el potencial de pérdida de precisión. Específicamente, si dos números están de acuerdo con n bits, se pueden perder n bits de precisión en la resta. Esto puede ser más fácil de ver en el extremo: si dos números no son iguales en teoría pero son iguales en su representación de máquina, su diferencia se calculará como cero, 100% de pérdida de precisión.

Aquí hay un ejemplo donde tal pérdida de precisión surge a menudo. La derivada de una función f en un punto x se define como el límite de

$$(f(x + h) - f(x))/h$$

cuando h llega a cero. Entonces, un enfoque natural para calcular la derivada de una función sería evaluar

$$(f(x + h) - f(x))/h$$

para alguna h pequeña. En teoría, cuanto menor es h , mejor se aproxima esta fracción a la derivada. En la práctica, la precisión mejora por un tiempo, pero más allá de cierto punto, valores más pequeños de h resultan en peores aproximaciones a la derivada. A medida que h disminuye, el error de aproximación se reduce pero el error numérico aumenta. Esto se debe a que la resta

$$f(x + h) - f(x)$$

se vuelve problemática. Si toma h lo suficientemente pequeño (después de todo, en teoría, más pequeño es mejor), entonces $f(x+h)$ será igual a $f(x)$ a la precisión de la máquina. Esto significa que todas las derivadas se calcularán como cero, sin importar la función, si solo toma h lo suficientemente pequeño. Aquí hay un ejemplo que calcula la derivada de $\sin(x)$ en $x = 1$.

```
cout << std::setprecision(15);
for (int i = 1; i < 20; ++i)
{
double h = pow(10.0, -i);
cout << (sin(1.0+h) - sin(1.0))/h << "\n";
}
cout << "El verdadero resultado es: " << cos(1.0) << "\n";
```

Aquí está la salida del código anterior. Para que la salida sea más fácil de entender, los dígitos después del primer dígito incorrecto se han reemplazado por puntos.

```
0.4.....
0.53.....
0.53.....
0.5402.....
0.5402.....
0.540301.....
0.5403022.....
0.540302302...
```

```
0.54030235....
0.5403022.....
0.540301.....
0.54034.....
0.53.....
0.544.....
0.55.....
0
0
0
0
```

El verdadero resultado es: 0.54030230586814

La precisión⁸⁸ mejora a medida que h se hace más pequeña hasta que $h = 10^{-8}$. Pasado ese punto, la precisión decae debido a la pérdida de precisión en la resta. Cuando $h = 10^{-16}$ o menor, la salida es exactamente cero porque $\sin(1.0 + h)$ es igual a $\sin(1.0)$ a la precisión de la máquina. (De hecho, $1 + h$ equivale a 1 a la precisión de la máquina. Más sobre eso a continuación).

¿Qué haces cuando tu problema requiere resta y va a causar una pérdida de precisión? A veces la pérdida de precisión no es un problema; los doubles comienzan con mucha precisión de sobra. Cuando la precisión es importante, a menudo es posible usar algún truco para cambiar el problema de modo que no requiera resta o no requiera la misma resta con la que comenzaste.

Los Números de Punto Flotante Tienen Rangos Finitos Todos saben que los números de punto flotante tienen rangos finitos, pero esta limitación puede aparecer de manera inesperada. Por ejemplo, puede encontrar sorprendente la salida de las siguientes líneas de código

```
float f = 16777216;
cout << f << " " << f+1 << "\n";
```

⁸⁸Los resultados anteriores se calcularon con Visual C++ 2008. Cuando se compiló con gcc 4.2.3 en Linux, los resultados fueron los mismos, excepto los últimos cuatro números. Donde VC++ produjo ceros, gcc produjo números negativos: -0.017 ..., - 0.17 ..., -1.7 ... y 17

Este código imprime el valor 16777216 dos veces. ¿Que pasó? De acuerdo con la especificación IEEE para aritmética de punto flotante, un tipo flotante tiene 32 bits de ancho. Veinticuatro de estos bits están dedicados al significado (lo que solía llamarse la mantisa) y el resto al exponente. El número 16777216 es 2^{24} y, por lo tanto, a la variable flotante f no le queda precisión para representar $f + 1$. Ocurriría un fenómeno similar para 2^{53} si f fuera del tipo `double` porque un doble de 64 bits dedica 53 bits al significado. El siguiente código imprime 0 en lugar de 1.

```
x = 9007199254740992; // 2^53
cout << ((x+1) - x) << "\n";
```

También podemos quedarnos sin precisión al agregar números pequeños a números de tamaño moderado. Por ejemplo, el siguiente código imprime "¡Lo siento!" porque `DBL_EPSILON` (definido en `float.h`) es el número positivo más pequeño ϵ tal que $1 + \epsilon \neq 1$ cuando se usan tipos dobles.

```
x = 1.0;
y = x + 0.5*DBL_EPSILON;
if (x == y)
    cout << "¡Lo siento!\n";
```

De manera similar, la constante `FLT_EPSILON` es el número positivo más pequeño ϵ tal que $1 + \epsilon$ no es 1 cuando se usan tipos flotantes.

¿Por qué los Flotantes IEEE Tienen Dos Ceros: +0 y -0? Aquí hay un detalle extraño de la aritmética de punto flotante IEEE: las computadoras tienen dos versiones de 0: cero positivo y cero negativo. La mayoría de las veces, la distinción entre +0 y -0 no importa, pero de vez en cuando las versiones firmadas del cero son útiles.

Si una cantidad positiva llega a cero, se convierte en +0. Y si una cantidad negativa llega a cero, se convierte en -0. Podría pensar en +0 (respectivamente, -0) como el patrón de bits para un número positivo (negativo) demasiado pequeño para representarlo.

El estándar de punto flotante IEEE dice que $1/+0$ debería ser *+infinito* y $1/-0$ debería ser *-infinito*. Esto tiene sentido si interpreta $+/-0$ como el fantasma de un número que se desbordó dejando solo su signo. El recíproco de un número positivo (negativo) demasiado pequeño para representarlo es un número positivo (negativo) demasiado grande para representarlo.

Para demostrar esto, ejecute el siguiente código C:

```
int main()
{
    double x = 1e-200;
    double y = 1e-200 * x;
    printf("Reciproco de +0: %g\n", 1/y);
    y = -1e-200*x;
    printf("Reciproco de -0: %g\n", 1/y);
}
```

En Linux con gcc, la salida es:

```
Reciproco de +0: inf
Reciproco de -0: -inf
```

Sin embargo, hay algo acerca de los ceros firmados y las excepciones que no tiene sentido. El informe acertadamente denominado "Lo que todo informático debería saber sobre la aritmética de punto flotante" tiene lo siguiente que decir sobre los ceros con signo.

En aritmética IEEE, es natural definir $\log 0 = -\infty$ y $\log x$ como un *NaN* cuando $x < 0$. Suponga que x representa un pequeño número negativo que se ha desbordado a cero. Gracias al cero con signo, x será negativo, por lo que \log puede devolver un *NaN*. Sin embargo, si no hubiera un cero con signo, la función logarítmica no podría distinguir un número negativo subdesbordado de 0 y, por lo tanto, tendría que devolver $-\infty$.

Esto implica que $\log(-0)$ debe ser *NaN* y $\log(+0)$ debe ser $-\infty$. Eso tiene sentido, pero eso no es lo que sucede en la práctica. La función de \log devuelve $-\infty$ para $+0$ y -0 .

Ejecuté el siguiente código en C:

```
int main()
{
    double x = 1e-200;
    double y = 1e-200 * x;
    printf("Log de +0: %g\n", log(y));
    y = -1e-200*x;
    printf("Log de -0: %g\n", log(y));
}
```

En Linux, el código imprime:

```
Log de +0: -inf
Log de -0: -inf
```

Use Logaritmos para Evitar Desbordamiento y Subdesbordamiento

Las limitaciones de los números de punto flotante descritos en la sección anterior provienen de tener un número limitado de bits en el significado. El desbordamiento y el subdesbordamiento resultan de tener también un número finito de bits en el exponente. Algunos números son demasiado grandes o demasiado pequeños para almacenarlos en un número de punto flotante.

Muchos problemas parecen requerir calcular un número de tamaño moderado como la razón de dos números enormes. El resultado final puede ser representable como un número de punto flotante aunque los resultados intermedios no lo sean. En este caso, los logaritmos proporcionan una salida. Si desea calcular M/N para grandes números M y N , calcule $\log(M) - \log(N)$ y aplique $\exp()$ al resultado. Por ejemplo, las probabilidades a menudo implican proporciones de factoriales, y los factoriales se vuelven astronómicamente grandes rápidamente. Para $N > 170$, $N!$ es mayor que `DBL_MAX`, el número más grande que puede representarse por un doble (sin precisión extendida). Pero es posible evaluar expresiones como $200!/(190!10!)$ Sin desbordamiento de la siguiente manera:

```
x = exp( logFactorial(200) - logFactorial(190) - logFactorial(10) );
```

Una función `logFactorial` simple pero ineficiente podría escribirse de la siguiente manera:

```
double logFactorial(int n)
{
    double sum = 0.0;
    for (int i = 2; i <= n; ++i) sum += log((double)i);
    return sum;
}
```

Un mejor enfoque sería utilizar una función de registro gamma si hay una disponible. Consulte [Cómo calcular las probabilidades binomiales para obtener más información](#).

Las operaciones numéricas no siempre devuelven números Debido a que los números de punto flotante tienen sus limitaciones, a veces las operaciones de punto flotante devuelven "infinito" como una forma de decir "el resultado es más grande de lo que puedo manejar". Por ejemplo, el siguiente código imprime 1. # *INF* en Windows e *inf* en Linux.

```
x = DBL_MAX;
cout << 2*x << "\n";
```

A veces, la barrera para devolver un resultado significativo tiene que ver con la lógica en lugar de la precisión finita. Los tipos de datos de punto flotante representan números reales (a diferencia de los números complejos) y no hay un número real cuyo cuadrado sea -1 . Eso significa que no hay un número significativo que devolver si el código solicita $\text{sqrt}(-2)$, incluso con una precisión infinita. En este caso, las operaciones de punto flotante devuelven *NaN*. Estos son valores de punto flotante que representan códigos de error en lugar de números. Los valores *NaN* se muestran como 1. # *IND* en Windows y *NAN* en Linux.

Una vez que una cadena de operaciones encuentra un *NaN*, todo es un *NaN* de ahí en adelante. Por ejemplo, suponga que tiene un código que equivale a algo como lo siguiente:

```
if (x - x == 0)
    // hacer algo
```

¿Qué podría impedir que se ejecute el código que sigue a la instrucción *if*? Si *x* es un *NaN*, entonces también lo es $x - x$ y los *NaN* no equivalen a nada. De hecho, los *NaN* ni siquiera se igualan. Eso significa que la expresión $x == x$ se puede usar para probar si *x* es un número (posiblemente infinito). Para obtener más información sobre infinitos y *NaN*, consulte las excepciones de punto flotante IEEE en C ++.

Relaciones de Factoriales Los cálculos de probabilidad a menudo implican tomar la razón de números muy grandes para producir un número de tamaño moderado. El resultado final puede caber dentro de un doble con espacio de sobra, pero los resultados intermedios se desbordarían. Por ejemplo, suponga que necesita calcular el número de formas de seleccionar 10 objetos de un conjunto de 200. ¡Esto es $200!/(190!10!)$, Aproximadamente $2.2e16$. Pero $200!$ y $190!$ desbordaría el rango de un doble.

Hay dos formas de solucionar este problema. Ambos usan la siguiente regla: Use trucos algebraicos para evitar el desbordamiento.

El primer truco es reconocer que

$$200! = 200 * 199 * 198 * \dots * 191 * 190!$$

y así

$$200!/(190!10!) = 200 * 199 * 198 * \dots * 191/10!$$

esto ciertamente funciona, pero está limitado a factoriales.

Una técnica más general es usar logaritmos para evitar el desbordamiento: tome el logaritmo de la expresión que desea evaluar y luego exponga el resultado. En este ejemplo

$$\log(200!/(190!10!)) = \text{Log}(200!) - \log(190!) - \log(10!)$$

si tiene un código que calcula el logaritmo de los factoriales directamente sin calcular primero los factoriales, puede usarlo para encontrar el logaritmo del resultado que desea y luego aplicar la función *exp*.

log (1 + x) Ahora veamos el ejemplo del cálculo de $\log(x + 1)$. Considere el siguiente código:

```
double x = 1e-16;
double y = log(1 + x)/x;
```

En este código $y = 0$, aunque el valor correcto sea igual a 1 para la precisión de la máquina.

¿Qué salió mal? los números de doble precisión tienen una precisión de aproximadamente 15 decimales, por lo que $1 + x$ equivale a 1 para la precisión de la máquina. El registro de 1 es cero, por lo que y se establece en cero. Pero para valores pequeños de x , $\log(1 + x)$ es aproximadamente x , por lo que $\log(1 + x)/x$ es aproximadamente 1. Eso significa que el código anterior para calcular $\log(1 + x)/x$ devuelve un resultado con 100% de error relativo. Si x no es tan pequeño que $1 + x$ es igual a 1 en la máquina, aún podemos tener problemas. Si x es moderadamente pequeño, los bits en x no se pierden totalmente al calcular $1 + x$, pero algunos sí. Cuanto más se acerca x a 0, más bits se pierden. Podemos usar la siguiente regla: Utilice aproximaciones analíticas para evitar la pérdida de precisión.

La forma favorita de aproximación de los analistas numéricos es la serie de potencia. ¡La serie de potencia para

$$\log(1 + x) = x + x^2/2! + x^3/3! + \dots$$

Para valores pequeños de x , simplemente devolver x para $\log(1 + x)$ es una mejora. Esto funcionaría bien para los valores más pequeños de x , pero para algunos valores no tan pequeños, esto no será lo suficientemente preciso, pero tampoco lo hará directamente el cálculo del $\log(1 + x)$.

Logit inverso A continuación, veamos el cálculo $f(x) = e^x/(1 + e^x)$. (Los estadísticos llaman a esto la función "logit inverso" porque es el inverso de la función que ellos llaman la función "logit".) El enfoque más directo sería calcular $\exp(x)/(1 + \exp(x))$. Veamos dónde se puede romper eso.

```
double x = 1000;
double t = exp(x);
double y = t/(1.0 + t);
```

Imprimir y da -1. # *IND*. Esto se debe a que el cálculo de t se desbordó, produciendo un número mayor que el que cabía en un doble. Pero podemos calcular el valor de y fácilmente. Si t es tan grande que no podemos almacenarlo, entonces $1 + t$ es esencialmente lo mismo que t y la relación es muy cercana a 1. Esto sugiere que descubramos qué valores de x son tan grandes que $f(x)$ será igual a 1 a la precisión de la máquina, luego solo devuelva 1 para esos valores de x para evitar la posibilidad de desbordamiento. Esta es nuestra siguiente regla: No calcules un resultado que puedas predecir con precisión.

El archivo de encabezado `float.h` tiene un `DBL_EPSILON` constante, que es la precisión doble más pequeña que podemos agregar a 1 sin recuperar 1. Un poco de álgebra muestra que si x es más grande que $-\log(\text{DBL_EPSILON})$, entonces $f(x)$ será igual a 1 a la precisión de la máquina. Así que aquí hay un fragmento de código para calcular $f(x)$ para valores grandes de x :

```
const double x_max = -log(DBL_EPSILON);
if (x > x_max) return 1.0;
```

El código provisto en esta sección calcula siete funciones que aparecen en las estadísticas. Cada uno evita problemas de desbordamiento, subdesbordamiento o pérdida de precisión que podrían ocurrir para grandes argumentos negativos, grandes argumentos positivos o argumentos cercanos a cero:

- LogOnePlusX calcúlese como $\log(1 + x)$ como en ejemplo antes visto
- ExpMinusOne calcúlese como $e^x - 1$
- Logit calcúlese como $\log(x/(1 - x))$
- LogitInverse calcúlese como $e^x/(1 + e^x)$ como se discutió en el ejemplo último
- LogLogitInverse calcúlese como $\log(e^x/(1 + e^x))$
- LogitInverseDifference calcúlese como $\text{LogitInverse}(x) - \text{LogitInverse}(y)$
- LogOnePlusExpX calcúlese como $\log(1 + \exp(x))$
- ComplementaryLogLog calcúlese como $\log(-\log(1 - x))$
- ComplementaryLogLogInverse calcúlese como $1.0 - \exp(-\exp(x))$

Las soluciones presentadas aquí parecen innecesarias o incluso incorrectas al principio. Si este tipo de código no está bien comentado, alguien lo verá y lo "simplificará" incorrectamente. Estarán orgullosos de todo el desorden innecesario que eliminaron. Y si no prueban valores extremos, su nuevo código parecerá funcionar correctamente. Las respuestas incorrectas y los *NaN* solo aparecerán más tarde.

23.3 Aritmética de Baja Precisión

La popularidad de la aritmética de baja precisión para el cómputo de alto rendimiento se ha disparado desde el lanzamiento en 2017 de la GPU Nvidia Volta. Los núcleos tensores de media precisión de Volta ofrecieron una enorme ganancia de rendimiento 16 veces mayor que la doble precisión para operaciones clave. Y el rendimiento del Hardware está mejorando aún más: el FP16 con núcleo tensor Nvidia H100 es 58 veces más rápido que el FP64 estándar.

Esta sorprendente aceleración ciertamente llama la atención. Sin embargo, en el cálculo científico, la aritmética de baja precisión suele considerarse insegura para los códigos de modelado y simulación. De hecho, a veces se puede aprovechar una precisión más baja, comúnmente en una configuración de "precisión mixta" en la que sólo partes del cálculo se realizan

con baja precisión. Sin embargo, en general, cualquier precisión menor que el doble se considera inadecuada para modelar fenómenos físicos complejos con fidelidad.

En respuesta, los desarrolladores han creado herramientas para medir la seguridad de la aritmética de precisión reducida en códigos de aplicación. Algunas herramientas pueden incluso identificar qué variables o matrices se pueden reducir de forma segura a una precisión menor sin perder precisión en el resultado final. Sin embargo, el uso de estas herramientas a ciegas, sin el respaldo de algún tipo de proceso de razonamiento, puede resultar peligroso.

Un ejemplo ilustrará esto:

El método del gradiente conjugado para la resolución y optimización de sistemas lineales y el método de Lanczos, estrechamente relacionado, para la resolución de problemas de valores propios mostraron una gran promesa tras su invención a principios de los años cincuenta. Sin embargo, se consideraban inseguros debido a errores de redondeo catastróficos en la aritmética de punto flotante, que son aún más pronunciados a medida que se reduce la precisión del punto flotante.

No obstante, Chris Paige demostró en su trabajo pionero en la década de 1970 que el error de redondeo, aunque sustancial, no excluye la utilidad de los métodos cuando se utilizan correctamente. El método del gradiente conjugado se ha convertido en un pilar del cómputo científico.

Teniendo en cuenta que ninguna herramienta podría llegar a este hallazgo sin un cuidadoso análisis matemático de los métodos. Una herramienta detectaría inexactitudes en el cálculo pero no podría certificar que estos errores no puedan perjudicar el resultado final.

Algunos podrían proponer en cambio un enfoque puramente basado en datos: simplemente pruebe con baja precisión en algunos casos de prueba; si funciona, utilice baja precisión en producción. Sin embargo, este enfoque está lleno de peligros: es posible que los casos de prueba no capturen todas las situaciones que podrían encontrarse en producción.

Por ejemplo, uno podría probar un código de aerodinámica sólo en regímenes de flujo suaves, pero las series de producción pueden encontrar flujos complejos con gradientes pronunciados, que la aritmética de baja precisión no puede modelar correctamente. Los artículos académicos que prueban

métodos y herramientas de baja precisión deben evaluarse rigurosamente en escenarios desafiantes del mundo real como este.

Lamentablemente, los equipos de ciencia computacional frecuentemente no tienen tiempo para evaluar sus códigos para un uso potencial de aritmética de menor precisión. Las herramientas ciertamente podrían ayudar. Además, las bibliotecas que encapsulan métodos de precisión mixta pueden ofrecer beneficios a muchos usuarios. Una gran historia de éxito son los solucionadores lineales densos de precisión mixta, basados en el sólido trabajo teórico de Nick Highnam y sus colegas, que han llegado a bibliotecas como: Lu, Hao; Matheson, Michael; Wang, Feiyi; Joubert, Wayne; Ellis, Austin; Oles, Vladyslav. "OpenMxP-OpenSource Mixed Precision Computing,".

Entonces la respuesta final es "depende". Cada nuevo caso debe examinarse cuidadosamente y tomar una decisión basada en alguna combinación de análisis y pruebas.

Sí bien, NVIDIA lidera el mercado de las GPU para inteligencia artificial (IA) con una cuota de mercado aproximada del 80%, pero no es en absoluto la única empresa que tiene en su porfolio chips de vanguardia para IA. La compañía californiana Cerebras posee, de hecho, los procesadores para este escenario de uso más complejos que existen. Su chip WSE-2, por ejemplo, aglutina nada menos que 2.6 billones de transistores contabilizados en la escala numérica larga y 850,000 núcleos optimizados para IA.

Cerebras entrega a sus clientes estos procesadores integrados en una plataforma para IA conocida como CS-2, y precisamente uno de ellos es la compañía de Emiratos Árabes G42. Esta última está construyendo seis superordenadores para IA capaces de superar la barrera de la exaescala que aglutinan una gran cantidad de sistemas CS-2. Y según la CIA algunas de estas máquinas irán a parar a las grandes tecnológicas chinas. No obstante, esto no es todo. Y es que Cerebras dió a conocer en 2024 un procesador para IA aún más potente que su WSE-2.

El procesador WSE-3 Cerebras ya tiene listo su procesador WSE-3 (Wafer Scale Engine 3), un producto que, como podemos intuir, está llamado a suceder al también ambicioso WSE-2.

Ambos procesadores se fabrican a partir de una oblea completa de silicio, lo que permite a Cerebras integrar muchos más bloques funcionales y núcleos en la lógica que una GPU convencional como las que fabrican NVIDIA, AMD o Huawei.

Y es que aglutina 4 billones de transistores, tiene una superficie de 46, 225 mm², integra nada menos que 900, 000 núcleos optimizados para IA y tiene una potencia de cálculo, según Cerebras, de 125 petaflops.

Según Cerebras su procesador WSE-3 es el doble de potente que el WSE-2. De hecho, de acuerdo con las especificaciones que ha publicado rinde como 62 GPU H100 de NVIDIA trabajando al unísono, y no debemos pasar por alto que este procesador de la compañía liderada por Jensen Huang es el más potente que tiene hasta que se produzca el lanzamiento de la GPU H200.

Sea como sea Cerebras entrega sus procesadores WSE-3 integrados en un superordenador conocido como CS-3 que es capaz de entrenar grandes modelos de IA con hasta 24 billones de parámetros. El mapa de memoria externa de este superordenador oscila entre 1.5 TB y 1.2 PB, un espacio de almacenamiento descomunal que permite almacenar modelos de lenguaje masivos en un único espacio lógico.

Según informan, el chip WSE-3 optimizado para la IA es capaz de entrenar hasta 24, 000 millones de parámetros, lo que también equivaldría a un rendimiento máximo de IA de 125 petaflops.

24 Apéndice F: Método de Elementos Finitos

En este capítulo se considera el VBVP de la forma

$$\begin{aligned}\mathcal{L}u &= f_\Omega \quad \text{en } \Omega \\ u &= g \quad \text{en } \partial\Omega\end{aligned}\tag{24.1}$$

donde

$$\mathcal{L}u = -\nabla \cdot \underline{a} \cdot \nabla u + cu\tag{24.2}$$

con \underline{a} una matriz positiva definida, simétrica y $c \geq 0$, como un caso particular del operador elíptico definido por la Ec. (4.43) de orden 2, con $\Omega \subset R^2$ un dominio poligonal, es decir, Ω es un conjunto abierto acotado y conexo tal que su frontera $\partial\Omega$ es la unión de un número finito de polígonos.

La sencillez del operador \mathcal{L} nos permite facilitar la comprensión de muchas de las ideas básicas que se expondrán a continuación, pero tengamos en mente que esta es una ecuación que gobierna los modelos de muchos sistemas de la ciencia y la ingeniería, por ello es muy importante su solución.

Si multiplicamos a la ecuación $-\nabla \cdot \underline{a} \cdot \nabla u + cu = f_\Omega$ por $v \in V = H_0^1(\Omega)$, obtenemos

$$-v(\nabla \cdot \underline{a} \cdot \nabla u + cu) = vf_\Omega\tag{24.3}$$

aplicando el teorema de Green (167) obtenemos la Ec. (4.50), que podemos re-escribir como

$$\int_\Omega (\nabla v \cdot \underline{a} \cdot \nabla u + cuv) d\underline{x} = \int_\Omega vf_\Omega d\underline{x}.\tag{24.4}$$

Definiendo el operador bilineal

$$a(u, v) = \int_\Omega (\nabla v \cdot \underline{a} \cdot \nabla u + cuv) d\underline{x}\tag{24.5}$$

y la funcional lineal

$$l(v) = \langle f, v \rangle = \int_\Omega vf_\Omega d\underline{x}\tag{24.6}$$

podemos reescribir el problema dado por la Ec. (24.1) de orden 2 en forma variacional, haciendo uso de la forma bilineal $a(\cdot, \cdot)$ y la funcional lineal $l(\cdot)$.

24.1 Triangulación

El Mallado o triangulación \mathcal{T}_h del dominio Ω es el primer aspecto básico, y ciertamente el más característico, el dominio $\Omega \subset \mathbb{R}^2$ es subdividido en E subdominios o elementos Ω_e llamados elementos finitos, tal que

$$\bar{\Omega} = \bigcup_{e=1}^E \bar{\Omega}_e$$

donde:

- Cada $\Omega_e \in \mathcal{T}_h$ es un polígono (rectángulo o triángulo) con interior no vacío ($\mathring{\Omega}_e \neq \emptyset$) y conexo.
- Cada $\Omega_e \in \mathcal{T}_h$ tiene frontera $\partial\Omega_e$ Lipschitz continua.
- Para cada $\Omega_i, \Omega_j \in \mathcal{T}_h$ distintos, $\mathring{\Omega}_i \cap \mathring{\Omega}_j = \emptyset$.
- El diámetro $h_i = \text{Diam}(\Omega_e)$ de cada Ω_e satisface $\text{Diam}(\Omega_e) \leq h$ para cada $e = 1, 2, \dots, E$.
- Los vértices de cada Ω_e son llamados nodos, teniendo N de ellos por cada elemento Ω_e .

Definición 129 Una familia de triangulaciones \mathcal{T}_h es llamada de forma-regular si existe una constante independiente de h , tal que

$$h_K \leq C\rho_K, \text{ con } K \in \mathcal{T}_h,$$

donde ρ_K es el radio del círculo más grande contenido en K . El radio h_K/ρ_K es llamado el aspect ratio de K .

Definición 130 Una familia de triangulaciones \mathcal{T}_h es llamada cuasi-uniforme si esta es de forma-regular y si existe una constante independiente de h , tal que

$$h_K \leq Ch, \text{ con } K \in \mathcal{T}_h.$$

24.2 Generar Mallas con GMSH

GMSH es un generador para mallar tanto sus superficies como sus volúmenes para Elementos Finitos de código abierto con un motor integrado tipo CAD y un post-procesador. Su diseño tiene como fin proveer una herramienta rápida, ligera y amigable con un editor paramétrico y una gran capacidad de visualización avanzada. GMSH está organizado en cuatro módulos: geometría, malla, motor de cálculo, y post-proceso. La introducción de datos e instrucciones a cualquiera de los módulos puede hacerse ya sea de manera interactiva usando la interfaz gráfica, mediante archivos de texto ASCII empleando el lenguaje propio de GMSH ó a través de la interfaz de programación de aplicaciones C, C++, Python, Julia, Fortran, etc.

Tipos de elementos finitos más comunes

Geometria Grados de libertad Σ	# gdl	Espacio de funciones	Continuidad del espacio MEF	Geometria Grados de libertad Σ	# gdl	Espacio de funciones	Continuidad del espacio MEF
	3	$P_1(K)$	C^0		6	$P_2(K)$	C^0
	10	$P_3(K)$	C^0		10	$P_3(K)$	C^0
	21	$P_5(K)$	C^1		18	$P_5(K)$	C^1

• Valor de la función

↑ → Valores de las derivadas primeras

↗ ↘ Valores de las derivadas segundas

⊗ Valor de la derivada normal al lado

Mallado. Se define como mallado la discretización de una línea, superficie o volumen en porciones de tamaño finito. Las porciones, además de tener un tamaño característico varias veces menor que el espacio discretizado,

serán entidades geométricas elementales como los triángulos o cuadriláteros en dos dimensiones o los tetraedros o prismas en tres dimensiones. Esta discretización en estructuras más elementales es esencial para la resolución de ecuaciones en derivadas parciales en dominios arbitrarios por el método de volúmenes finitos (FVM) o el método de elementos finitos (FEM).

Tipos de elementos finitos más comunes (cont)

Geometría	Grados de libertad Σ	# gdl	Espacio de funciones	Continuidad del espacio MEF	Geometría	Grados de libertad Σ	# gdl	Espacio de funciones	Continuidad del espacio MEF
		4	$Q_1(K)$	C^0			9	$Q_2(K)$	C^0
		16	$Q_3(K)$	C^0			2	$P_1(K)$	C^0
		3	$P_2(K)$	C^0			4	$P_3(K)$	C^1
		4	$P_1(K)$	C^0			10	$P_2(K)$	C^0

•	Valor de la función		Valores de las derivadas segundas
	Valores de las derivadas primeras		Valor de la derivada normal al lado

El proceso de mallado más habitual es el de discretizar sucesivamente entidades de mayor dimensión como si de un problema de contorno se tratara. Por ejemplo, si queremos mallar un cubo primero definiremos los puntos de control en cada una de sus doce aristas. Luego discretizaremos en porciones elementales cada uno de sus seis lados y finalmente discretizaremos el volúmen. Para entender mejor este proceso, y con anterioridad a entender la sintaxis de los archivos GMSH presentamos este ejemplo de mallado de un cubo con tetraedros. El primer paso es definir los puntos básicos de la geometría.

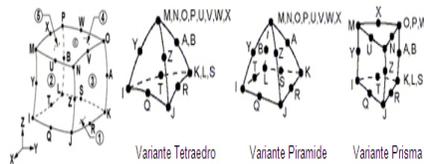
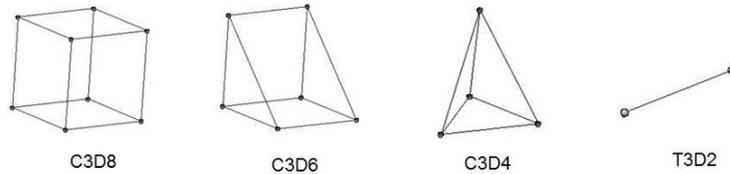


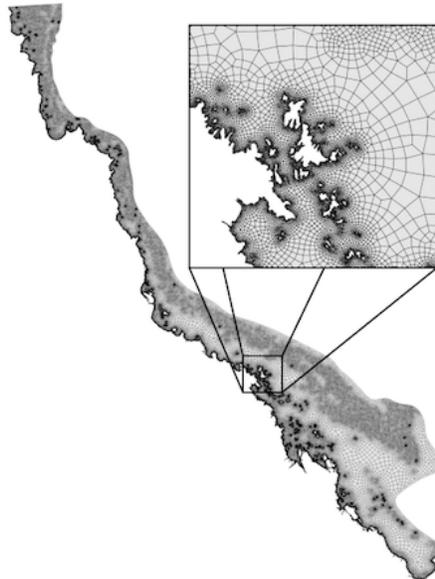
Fig. 3. Geometría del elemento finito sólido estructural

Mallas estructuradas Una malla estructurada es una malla que puede ser transformada, mediante una transformación biyectiva, a una cuadrícula. Esto significa que el problema podría ser resuelto en una malla uniforme y rectangular y devuelto a la geometría original sólo conociendo el jacobiano de la transformación porque este jacobiano se puede invertir.

Mallas no estructuradas Las mallas no estructuradas parten de distribuciones de puntos que cumplen una serie de propiedades dadas, la mayoría de ellas relacionadas con una distribución lo más uniforme posible dentro del contorno. Son adecuadas para geometrías irregulares donde es muy difícil generar una malla estructurada.

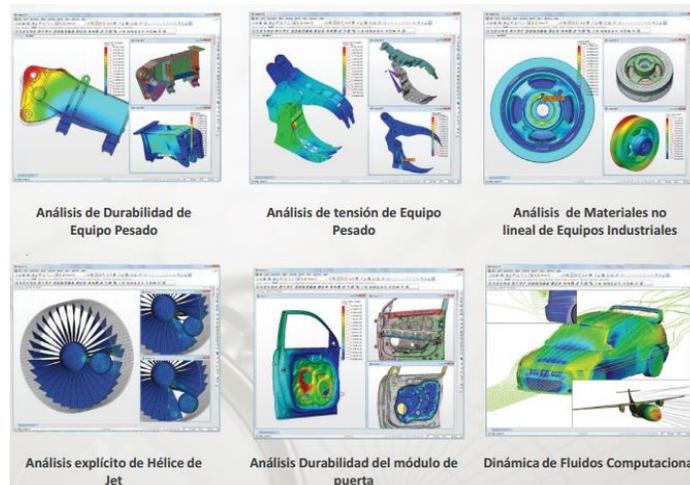
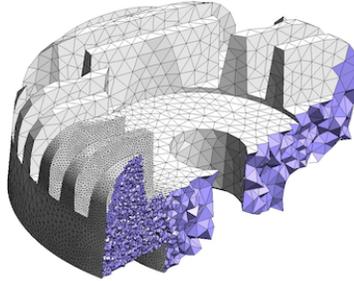
GMSH utiliza un algoritmo de creación de malla basado en la triangulación Delaunay. Dado un contorno se demuestra que sólo existe una triangulación óptima en la que, para cada triángulo formado por tres puntos de la malla, no exista ningún punto dentro de la circunferencia que pasa por los tres puntos.

Si bien esta propiedad cierra el problema de la triangulación dados los puntos de la malla, en la mayoría de los casos sólo dispondremos del contorno. GMSH es capaz de encontrar una distribución no estructurada de puntos cuya triangulación resultante tiene suficiente calidad.



Discretización y calidad de malla. Una de las diferencias esenciales entre las mallas estructuradas y no estructuradas es el proceso de refinado. En algunos problemas la malla utilizada no produce suficiente precisión y es necesario rehacerla para aumentarla. En las mallas estructuradas el aumento de precisión no suele suponer un problema más allá de generar demasiados grados de libertad. Sin embargo en las mallas no estructuradas algunos algoritmos de refinado automático pueden funcionar mal.

Uno de los algoritmos más utilizados es el de partir cada uno de los triángulos o tetraedros en dos o más elementos. Este algoritmo recursivo puede refinar puntos arbitrarios en el espacio produciendo una pérdida de



precisión al introducir gradientes espurios debidos a una mala definición de la geometría.

GMSH se puede usar en conjunto con FreeFem++ que es un Software libre que sirve para la resolución numérica de problemas 2D y 3D utilizando el método de elementos finitos. ¿Qué problemas puedes resolver? De mecánica de fluidos, difusión de calor, elasticidad, electromagnetismo, etc. Todos aquellos que estén modelizados matemáticamente a través de una ecuación en derivadas parciales (EDP), por ejemplo:

<https://www.um.es/freefem/ff++/pmwiki.php?n=Main.Elasticidad3D>

Existe una gran cantidad herramientas para generar mallas en 2D, 3D que se pueden usar desde lenguajes de programación, entre las utilidades destacan:

- <https://www.salome-platform.org/>
- <https://www.paraview.org/>
- <https://www.meshlab.net/>
- <https://wias-berlin.de/Software/index.jsp?id=TetGen&lang=1>
- <http://alice.loria.fr/Software/geogram/doc/html/index.html>

Algo más de Ecuaciones Diferenciales Parciales En la red existen múltiples sitios especializados y una amplia bibliografía para conocer sobre Ecuaciones Diferenciales Parciales, nosotros hemos seleccionado diversos textos que ponemos a su disposición en:

EDP

Además existen una gran cantidad herramientas para para resolver ecuaciones diferenciales parciales que se pueden usar, entre ellas destacan:

- <https://freefem.org/>
- <https://www.opengeosys.org/>
- <https://fenicsproject.org/>
- <https://www.openfoam.com/>
- <https://www.dealii.org/>
- <https://www.dune-project.org/>
- <https://www.aivc.org/resource/domus-20-whole-building-hygrothermal-simulation-program>
- <https://www.ctcms.nist.gov/fipy/>
- <http://onelab.info/>
- <https://www.csc.fi/web/elmer>
- <https://github.com/KratosMultiphysics/Kratos>

- <https://www.gomafem.com/>
- <http://www.juliafem.org/>
- <https://getfem.org/>
- <http://mofem.eng.gla.ac.uk/mofem/html/>
- <http://mech.fsv.cvut.cz/~sifel/>
- <https://dedalus-project.org/>
- <http://www.oofem.org/doku.php?id=en:oofem>
- <http://www.hpfem.org/hermes/>
- <https://cimec.org.ar/foswiki/Main/Cimec/PETScFEM>
- <https://pypi.org/project/HydPy/>
- <http://sfepy.org/doc-devel/index.html>
- <https://www.firedrakeproject.org/>
- <https://github.com/romeric/florence>
- https://gfs.sourceforge.net/wiki/index.php/Main_Page
- <https://www.comsol.com/>
- <https://www.featool.com/>

Una vez que la triangulación \mathcal{T}_h del dominio Ω es establecida, se procede a definir el espacio de elementos finitos $\mathbb{P}^h[k]$ a través del proceso descrito a continuación.

24.3 Interpolación para el Método de Elementos Finitos

Funciones Base A continuación describiremos la manera de construir las funciones base usadas por el método de elemento finito. En este procedimiento debemos tener en cuenta que las funciones base están definidas en un subespacio de $V = H^1(\Omega)$ para problemas de segundo orden que satisfacen las condiciones de frontera.

Las funciones base deberán satisfacer las siguientes propiedades:

- i) Las funciones base ϕ_i son acotadas y continuas, i.e. $\phi_i \in C(\Omega_e)$.
- ii) Existen ℓ funciones base por cada nodo del polígono Ω_e , y cada función ϕ_i es no cero solo en los elementos contiguos conectados por el nodo i .
- iii) $\phi_i = 1$ en cada i nodo del polígono Ω_e y cero en los otros nodos.
- iv) La restricción ϕ_i a Ω_e es un polinomio, i.e. $\phi_i \in \mathbb{P}_k[\Omega_e]$ para alguna $k \geq 1$ donde $\mathbb{P}_k[\Omega_e]$ es el espacio de polinomios de grado a lo más k sobre Ω_e .

Decimos que $\phi_i \in \mathbb{P}_k[\Omega_e]$ es una base de funciones y por su construcción es evidente que estas pertenecen a $H^1(\Omega)$. Al conjunto formado por todas las funciones base definidas para todo Ω_e de Ω será el espacio $\mathbb{P}^h[k]$ de funciones base, i.e.

$$\mathbb{P}^h[k] = \bigcup_{e=1}^E \mathbb{P}_k[\Omega_e]$$

estas formarán las funciones base globales.

24.4 Discretización en 2D Usando Rectángulos

Para resolver la Ec. (24.1), usando una discretización con rectángulos, primero dividimos el dominio $\Omega \subset \mathbb{R}^2$ en N_x nodos horizontales por N_y nodos verticales, teniendo $E = (N_x - 1)(N_y - 1)$ subdominios o elementos rectangulares Ω_e tales que $\bar{\Omega} = \cup_{e=1}^E \bar{\Omega}_e$ y $\bar{\Omega}_i \cap \bar{\Omega}_j \neq \emptyset$ si son adyacentes, con un total de $N = N_x N_y$ nodos.

Donde las funciones lineales definidas por pedazos en Ω_e en nuestro caso serán polinomios de orden uno en cada variable separadamente y cuya restricción de ϕ_i a Ω_e es $\phi_i^{(e)}$. Para simplificar los cálculos en esta etapa, supondremos que la matriz $\underline{a} = a \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, entonces se tiene que la integral del lado izquierdo de la Ec. (5.32) queda escrita como

$$\int_{\Omega} (a \nabla \phi_i \cdot \nabla \phi_j + c \phi_i \phi_j) dx dy = \int_{\Omega} f_{\Omega} \phi_j dx dy \quad (24.7)$$

donde

$$\begin{aligned} K_{ij} &= \int_{\Omega} (a \nabla \phi_i \cdot \nabla \phi_j + c \phi_i \phi_j) dx dy \\ &= \sum_{e=1}^E \int_{\Omega_e} (a \nabla \phi_i^{(e)} \cdot \nabla \phi_j^{(e)} + c \phi_i^{(e)} \phi_j^{(e)}) dx dy \\ &= \sum_{e=1}^E \int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dx dy \end{aligned} \quad (24.8)$$

y el lado derecho como

$$\begin{aligned} F_j &= \int_{\Omega} f_{\Omega} \phi_j dx dy \\ &= \sum_{e=1}^E \int_{\Omega_e} f_{\Omega} \phi_j^{(e)} dx dy. \end{aligned} \quad (24.9)$$

Para cada Ω_e de Ω , la submatriz de integrales (matriz de carga local)

$$K_{ij} = \int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dx dy \quad (24.10)$$

tiene la estructura

$$\begin{bmatrix} K_{1,1}^{(e)} & K_{1,2}^{(e)} & K_{1,3}^{(e)} & K_{1,4}^{(e)} \\ K_{2,1}^{(e)} & K_{2,2}^{(e)} & K_{2,3}^{(e)} & K_{2,4}^{(e)} \\ K_{3,1}^{(e)} & K_{3,2}^{(e)} & K_{3,3}^{(e)} & K_{3,4}^{(e)} \\ K_{4,1}^{(e)} & K_{4,2}^{(e)} & K_{4,3}^{(e)} & K_{4,4}^{(e)} \end{bmatrix}$$

la cual deberá ser ensamblada en la matriz de carga global que corresponda a la numeración de nodos locales del elemento Ω_e con respecto a la numeración global de los elementos en Ω .

De manera parecida, para cada Ω_e de Ω se genera el vector de integrales (vector de carga local)

$$F_j = \int_{\Omega_e} f_{\Omega} \phi_j^{(e)} dx dy \quad (24.11)$$

con la estructura

$$\begin{bmatrix} F_1^{(e)} \\ F_2^{(e)} \\ F_3^{(e)} \\ F_4^{(e)} \end{bmatrix}$$

el cual también deberá ser ensamblado en el vector de carga global que corresponda a la numeración de nodos locales al elemento Ω_e con respecto a la numeración global de los elementos de Ω .

Montando los $K_{ij}^{(e)}$ en la matriz $\underline{\underline{\mathbb{K}}}$ y los $F_j^{(e)}$ en el vector $\underline{\underline{\mathbb{F}}}$ según la numeración de nodos global, se genera el sistema $\underline{\underline{\mathbb{K}}}u_h = \underline{\underline{\mathbb{F}}}$ donde u_h será el vector cuyos valores serán la solución aproximada a la Ec. (24.1) en los nodos interiores de Ω . La matriz $\underline{\underline{\mathbb{K}}}$ generada de esta forma, tiene una propiedad muy importante, es bandada y el ancho de banda es de 9 elementos, esto es muy útil al momento de soportar la matriz en memoria.

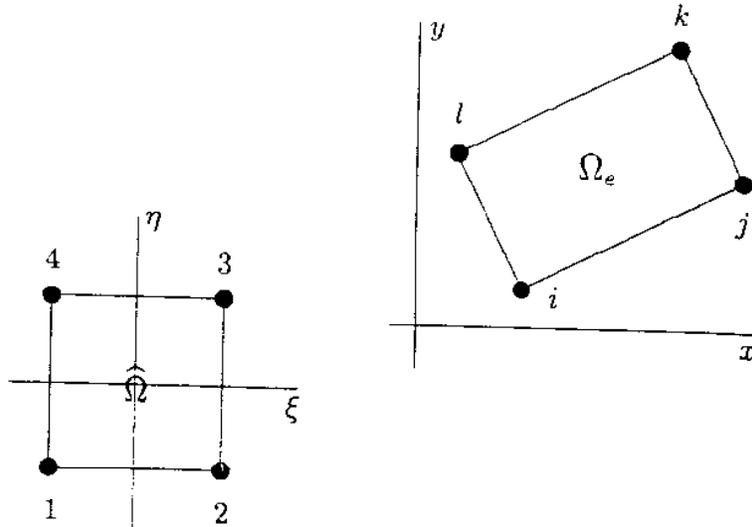
Para implementar numéricamente en cada Ω_e las integrales

$$\int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dx dy \quad (24.12)$$

y

$$\int_{\Omega_e} f_{\Omega} \phi_j^{(e)} dx dy, \quad (24.13)$$

teniendo en mente el simplificar los cálculos computacionales, se considera un elemento de referencia $\hat{\Omega}$ en los ejes coordenados (ε, η) cuyos vértices están el $(-1, -1)$, $(1, -1)$, $(1, 1)$ y $(-1, 1)$ respectivamente, en el cual mediante una función afín será proyectado cualquier elemento rectangular Ω_e cuyos vértices $(x_1^{(e)}, y_1^{(e)})$, $(x_2^{(e)}, y_2^{(e)})$, $(x_3^{(e)}, y_3^{(e)})$ y $(x_4^{(e)}, y_4^{(e)})$ están tomados en sentido contrario al movimiento de las manecillas del reloj como se muestra en la figura



mediante la transformación $f(x, y) = \underline{T}(\varepsilon, \eta) + \underline{b}$, quedando dicha transformación como

$$\begin{aligned} x &= \frac{x_2^{(e)} - x_1^{(e)}}{2} \varepsilon + \frac{y_2^{(e)} - y_1^{(e)}}{2} \eta \\ y &= \frac{x_4^{(e)} - x_1^{(e)}}{2} \varepsilon + \frac{y_4^{(e)} - y_1^{(e)}}{2} \eta \end{aligned} \quad (24.14)$$

en la cual la matriz \underline{T} está dada por

$$\underline{T} = \begin{pmatrix} \frac{x_2^{(e)} - x_1^{(e)}}{2} & \frac{y_2^{(e)} - y_1^{(e)}}{2} \\ \frac{x_4^{(e)} - x_1^{(e)}}{2} & \frac{y_4^{(e)} - y_1^{(e)}}{2} \end{pmatrix} \quad (24.15)$$

y el vector $\underline{b} = (b_1, b_2)$ es la posición del vector centroide del rectángulo Ω_e ,

también se tiene que la transformación inversa es

$$\begin{aligned} \varepsilon &= \frac{x - b_1 - \frac{y_2^{(e)} - y_1^{(e)}}{2} \left[\frac{y - b_2}{\left(\frac{x_4^{(e)} - x_1^{(e)}}{2} \right) \left(\frac{x - b_1 - \frac{y_2^{(e)} - y_1^{(e)}}{2}}{\frac{x_2^{(e)} - x_1^{(e)}}{2}} \right)} \right]}{\frac{x_2^{(e)} - x_1^{(e)}}{2}} \\ \eta &= \frac{y - b_2}{\left(\frac{x_4^{(e)} - x_1^{(e)}}{2} \right) \left(\frac{x - b_1 - \frac{y_2^{(e)} - y_1^{(e)}}{2}}{\frac{x_2^{(e)} - x_1^{(e)}}{2}} \right) + \frac{y_4^{(e)} - y_1^{(e)}}{2}}. \end{aligned} \quad (24.16)$$

Entonces las $\phi_i^{(e)}$ quedan definidas en términos de $\hat{\phi}_i$ cómo

$$\begin{aligned} \hat{\phi}_1(\varepsilon, \eta) &= \frac{1}{4}(1 - \varepsilon)(1 - \eta) \\ \hat{\phi}_2(\varepsilon, \eta) &= \frac{1}{4}(1 + \varepsilon)(1 - \eta) \\ \hat{\phi}_3(\varepsilon, \eta) &= \frac{1}{4}(1 + \varepsilon)(1 + \eta) \\ \hat{\phi}_4(\varepsilon, \eta) &= \frac{1}{4}(1 - \varepsilon)(1 + \eta) \end{aligned} \quad (24.17)$$

y las funciones $\phi_i^{(e)}$ son obtenidas por el conjunto $\phi_i^{(e)}(x, y) = \hat{\phi}_i(\varepsilon, \eta)$ con (x, y) y (ε, η) relacionadas por la Ec. (24.14), entonces se tendrían las siguientes integrales

$$\begin{aligned} K_{ij}^{(e)} &= \int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dx dy \\ &= \int_{\hat{\Omega}} \left(\left[a \left(\frac{\partial \hat{\phi}_i}{\partial \varepsilon} \frac{\partial \varepsilon}{\partial x} + \frac{\partial \hat{\phi}_i}{\partial \eta} \frac{\partial \eta}{\partial x} \right) \left(\frac{\partial \hat{\phi}_j}{\partial \varepsilon} \frac{\partial \varepsilon}{\partial x} + \frac{\partial \hat{\phi}_j}{\partial \eta} \frac{\partial \eta}{\partial x} \right) + \right. \right. \\ &\quad \left. \left. \left(\frac{\partial \hat{\phi}_i}{\partial \varepsilon} \frac{\partial \varepsilon}{\partial y} + \frac{\partial \hat{\phi}_i}{\partial \eta} \frac{\partial \eta}{\partial y} \right) \left(\frac{\partial \hat{\phi}_j}{\partial \varepsilon} \frac{\partial \varepsilon}{\partial y} + \frac{\partial \hat{\phi}_j}{\partial \eta} \frac{\partial \eta}{\partial y} \right) \right] + c \hat{\phi}_i \hat{\phi}_j \right) |J| d\varepsilon d\eta \end{aligned} \quad (24.18)$$

donde el índice i y j varía de 1 a 4. En esta última usamos la regla de la cadena y $dx dy = |J| d\varepsilon d\eta$ para el cambio de variable en las integrales,

aquí $|J| = \det T$, donde T está dado como en la Ec. (24.15). Para resolver $\int_{\Omega_e} f_{\Omega} \phi_j^{(e)} dx dy$ en cada Ω_e se genera las integrales

$$\begin{aligned} F_j^{(e)} &= \int_{\Omega_e} f_{\Omega} \phi_j^{(e)} dx dy \\ &= \int_{\hat{\Omega}} f_{\Omega} \hat{\phi}_j |J| d\varepsilon d\eta \end{aligned} \quad (24.19)$$

donde el índice i y j varía de 1 a 4.

Para realizar el cálculo numérico de las integrales en el rectángulo de referencia $\hat{\Omega} = [-1, 1] \times [-1, 1]$, debemos conocer $\frac{\partial \phi_i}{\partial \varepsilon}$, $\frac{\partial \phi_i}{\partial \eta}$, $\frac{\partial \varepsilon}{\partial x}$, $\frac{\partial \varepsilon}{\partial y}$, $\frac{\partial \eta}{\partial x}$ y $\frac{\partial \eta}{\partial y}$, entonces realizando las operaciones necesarias a la Ec. (24.17) obtenemos

$$\begin{aligned} \frac{\partial \phi_1}{\partial \varepsilon} &= -\frac{1}{4}(1 - \eta) & \frac{\partial \phi_1}{\partial \eta} &= -\frac{1}{4}(1 - \varepsilon) \\ \frac{\partial \phi_2}{\partial \varepsilon} &= \frac{1}{4}(1 - \eta) & \frac{\partial \phi_2}{\partial \eta} &= -\frac{1}{4}(1 + \varepsilon) \\ \frac{\partial \phi_3}{\partial \varepsilon} &= \frac{1}{4}(1 + \eta) & \frac{\partial \phi_3}{\partial \eta} &= \frac{1}{4}(1 + \varepsilon) \\ \frac{\partial \phi_4}{\partial \varepsilon} &= -\frac{1}{4}(1 + \eta) & \frac{\partial \phi_4}{\partial \eta} &= \frac{1}{4}(1 - \varepsilon) \end{aligned} \quad (24.20)$$

y también

$$\begin{aligned} \frac{\partial \varepsilon}{\partial x} &= \left(\frac{y_4^{(e)} - y_1^{(e)}}{2 \det T} \right) & \frac{\partial \varepsilon}{\partial y} &= \left(\frac{x_4^{(e)} - x_1^{(e)}}{2 \det T} \right) \\ \frac{\partial \eta}{\partial x} &= \left(\frac{y_2^{(e)} - y_1^{(e)}}{2 \det T} \right) & \frac{\partial \eta}{\partial y} &= \left(\frac{x_2^{(e)} - x_1^{(e)}}{2 \det T} \right) \end{aligned} \quad (24.21)$$

las cuales deberán de ser sustituidas en cada $\underline{K}_{ij}^{(e)}$ y $\underline{F}_j^{(e)}$ para calcular las integrales en el elemento Ω_e . Estas integrales se harán en el programa usando cuadratura Gaussiana, permitiendo reducir el número de cálculos al mínimo pero manteniendo el balance entre precisión y número bajo de operaciones necesarias para realizar las integraciones.

Suponiendo que Ω fue dividido en E elementos, estos elementos generan N nodos en total, de los cuales N_d son nodos desconocidos y N_c son nodos conocidos con valor γ_j , entonces el algoritmo de ensamble de la matriz \underline{K} y el vector \underline{F} se puede esquematizar como:

$$\begin{aligned} K_{i,j} &= (\phi_i, \phi_j) \quad \forall i = 1, 2, \dots, E, j = 1, 2, \dots, E \\ F_j &= (f_{\Omega}, \phi_j) \quad \forall j = 1, 2, \dots, E \\ \forall j &= 1, 2, \dots, N_d : \end{aligned}$$

$$b_j = b_j - \gamma_i K_{i,j} \quad \forall i = 1, 2, \dots, E$$

Así, se construye una matriz global en la cual están representados los nodos conocidos y los desconocidos, tomando sólo los nodos desconocidos de la matriz \underline{K} formaremos una matriz \underline{A} , haciendo lo mismo al vector \underline{F} formamos el vector \underline{b} , entonces la solución al problema será la resolución del sistema de ecuaciones lineales $\underline{Ax} = \underline{b}$, este sistema puede resolverse usando por ejemplo el método de Gradiente Conjugado. El vector \underline{x} contendrá la solución buscada en los nodos desconocidos N_d .

24.5 Discretización en 2D Usando Triángulos

Para resolver la Ec. (5.1), usando una discretización con triángulos, primero dividimos el dominio $\Omega \subset \mathbb{R}^2$ en N_x nodos horizontales por N_y nodos verticales, teniendo $E = 2(N_x - 1)(N_y - 1)$ subdominios o elementos triangulares Ω_e tales que $\bar{\Omega} = \cup_{e=1}^E \bar{\Omega}_e$ y $\bar{\Omega}_i \cap \bar{\Omega}_j \neq \emptyset$ si son adyacentes, con un total de $N = N_x N_y$ nodos.

Donde las funciones lineales definidas por pedazos en Ω_e en nuestro caso serán polinomios de orden uno en cada variable separadamente y cuya restricción de ϕ_i a Ω_e es $\phi_i^{(e)}$. Para simplificar los cálculos en esta etapa, supondremos que la matriz $\underline{a} = a \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, entonces se tiene que la integral del lado izquierdo de la Ec. (5.32) queda escrita como

$$\int_{\Omega} (a \nabla \phi_i \cdot \nabla \phi_j + c \phi_i \phi_j) dx dy = \int_{\Omega} f_{\Omega} \phi_j dx dy \quad (24.22)$$

donde

$$\begin{aligned} K_{ij} &= \int_{\Omega} (a \nabla \phi_i \cdot \nabla \phi_j + c \phi_i \phi_j) dx dy \\ &= \sum_{e=1}^E \int_{\Omega_e} (a \nabla \phi_i^{(e)} \cdot \nabla \phi_j^{(e)} + c \phi_i^{(e)} \phi_j^{(e)}) dx dy \\ &= \sum_{e=1}^E \int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dx dy \end{aligned} \quad (24.23)$$

y el lado derecho como

$$\begin{aligned} \underline{F}_j &= \int_{\Omega} f_{\Omega} \phi_j dx dy & (24.24) \\ &= \sum_{e=1}^E \int_{\Omega_e} f_{\Omega} \phi_j^{(e)} dx dy. \end{aligned}$$

Para cada Ω_e de Ω la submatriz de integrales (matriz de carga local)

$$\underline{\underline{K}}_{ij} = \int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dx dy \quad (24.25)$$

tiene la estructura

$$\begin{bmatrix} k_{1,1}^{(e)} & k_{1,2}^{(e)} & k_{1,3}^{(e)} \\ k_{2,1}^{(e)} & k_{2,2}^{(e)} & k_{2,3}^{(e)} \\ k_{3,1}^{(e)} & k_{3,2}^{(e)} & k_{3,3}^{(e)} \end{bmatrix}$$

la cual deberá ser ensamblada en la matriz de carga global que corresponda a la numeración de nodos locales del elemento Ω_e con respecto a la numeración global de los elementos en Ω .

De manera parecida, para cada Ω_e de Ω se genera el vector de integrales (vector de carga local)

$$F_j = \int_{\Omega_e} f_{\Omega} \phi_j^{(e)} dx dy \quad (24.26)$$

con la estructura

$$\begin{bmatrix} F_1^{(e)} \\ F_2^{(e)} \\ F_3^{(e)} \end{bmatrix}$$

el cual también deberá ser ensamblado en el vector de carga global que corresponda a la numeración de nodos locales al elemento Ω_e con respecto a la numeración global de los elementos de Ω .

Montando los $\underline{\underline{K}}_{ij}^{(e)}$ en la matriz $\underline{\underline{K}}$ y los $F_j^{(e)}$ en el vector \underline{F} según la numeración de nodos global, se genera el sistema $\underline{\underline{K}} \underline{u}_h = \underline{F}$ donde \underline{u}_h será el vector cuyos valores serán la solución aproximada a la Ec. (5.1) en los nodos interiores de Ω . La matriz $\underline{\underline{K}}$ generada de esta forma, tiene una propiedad muy importante, es bandada y el ancho de banda es de 7 elementos, esto es muy útil al momento de soportar la matriz en memoria.

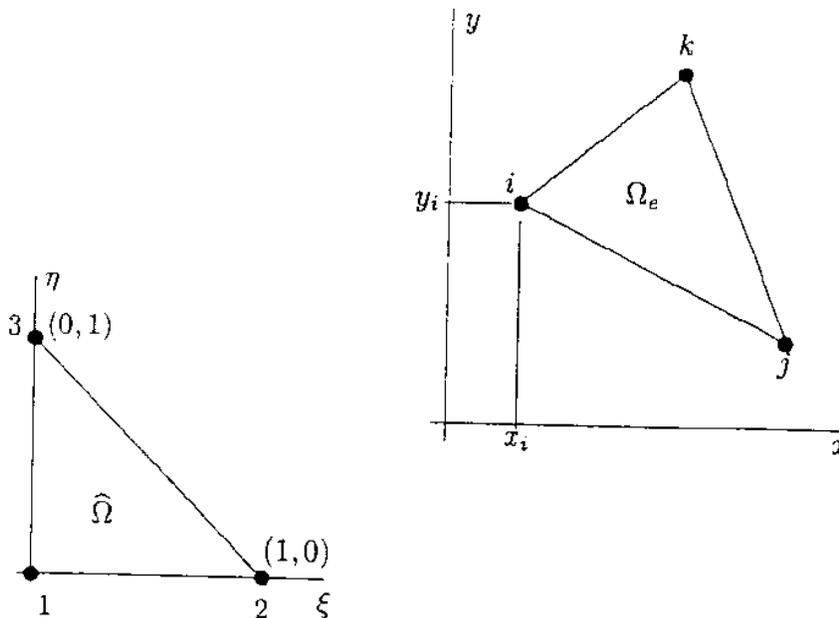
Para implementar numéricamente en cada Ω_e las integrales

$$\int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dx dy \quad (24.27)$$

y

$$\int_{\Omega_e} f_{\Omega} \phi_j^{(e)} dx dy$$

teniendo en mente el simplificar los cálculos computacionales se considera a un elemento de referencia $\hat{\Omega}$ en los ejes coordenados (ε, η) cuyos vertices estan en $(0, 0)$, $(1, 0)$ y $(0, 1)$ y en el cual mediante un mapeo afín será proyectado cualquier elemento triangular Ω_e cuyos vértices $(x_1^{(e)}, y_1^{(e)})$, $(x_2^{(e)}, y_2^{(e)})$, $(x_3^{(e)}, y_3^{(e)})$ están tomados en el sentido contrario al movimiento de las manecillas del reloj como se muestra en la figura



mediante la transformación $f(\varepsilon, \eta) = \underline{T}(\varepsilon, \eta) + \underline{b}$, quedando dicha transformación como

$$\begin{aligned} x &= x_1^{(e)}(1 - \varepsilon - \eta) + x_2^{(e)}\varepsilon + x_3^{(e)}\eta \\ y &= y_1^{(e)}(1 - \varepsilon - \eta) + y_2^{(e)}\varepsilon + y_3^{(e)}\eta \end{aligned} \quad (24.28)$$

y en la cual la matriz \underline{T} está dada por

$$\underline{T} = \begin{pmatrix} x_2^{(e)} - x_1^{(e)} & x_3^{(e)} - x_1^{(e)} \\ y_2^{(e)} - y_1^{(e)} & y_3^{(e)} - y_1^{(e)} \end{pmatrix} \quad (24.29)$$

donde \underline{b} es un vector constante

$$\underline{b} = \begin{pmatrix} x_1^{(e)} \\ y_1^{(e)} \end{pmatrix} \quad (24.30)$$

también se tiene que la transformación inversa es

$$\begin{aligned} \varepsilon &= \frac{1}{2A_{\Omega_e}} \left[\left(y_3^{(e)} - y_1^{(e)} \right) \left(x - x_1^{(e)} \right) - \left(x_3^{(e)} - x_1^{(e)} \right) \left(y - y_1^{(e)} \right) \right] \\ \eta &= \frac{1}{2A_{\Omega_e}} \left[- \left(y_2^{(e)} - y_1^{(e)} \right) \left(x - x_1^{(e)} \right) - \left(x_2^{(e)} - x_1^{(e)} \right) \left(y - y_1^{(e)} \right) \right] \end{aligned} \quad (24.31)$$

donde

$$A_{\Omega_e} = \left| \det \begin{bmatrix} 1 & x_1^{(e)} & y_1^{(e)} \\ 1 & x_2^{(e)} & y_2^{(e)} \\ 1 & x_3^{(e)} & y_3^{(e)} \end{bmatrix} \right|. \quad (24.32)$$

Entonces las $\phi_i^{(e)}$ quedan definidas en términos de $\hat{\phi}_i$ cómo

$$\begin{aligned} \hat{\phi}_1(\varepsilon, \eta) &= 1 - \varepsilon - \eta \\ \hat{\phi}_2(\varepsilon, \eta) &= \varepsilon \\ \hat{\phi}_3(\varepsilon, \eta) &= \eta \end{aligned} \quad (24.33)$$

entonces las funciones $\phi_i^{(e)}$ son obtenidas por el conjunto $\phi_i^{(e)}(x, y) = \hat{\phi}_i(\varepsilon, \eta)$ con (x, y) y (ε, η) relacionadas por la Ec. (24.28), entonces se tendrían las siguientes integrales

$$\begin{aligned} k_{ij}^{(e)} &= \int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dx dy \\ &= \int_{\hat{\Omega}} \left(\left[a \left(\frac{\partial \hat{\phi}_i}{\partial \varepsilon} \frac{\partial \varepsilon}{\partial x} + \frac{\partial \hat{\phi}_i}{\partial \eta} \frac{\partial \eta}{\partial x} \right) \left(\frac{\partial \hat{\phi}_j}{\partial \varepsilon} \frac{\partial \varepsilon}{\partial x} + \frac{\partial \hat{\phi}_j}{\partial \eta} \frac{\partial \eta}{\partial x} \right) + \right. \right. \\ &\quad \left. \left(\frac{\partial \hat{\phi}_i}{\partial \varepsilon} \frac{\partial \varepsilon}{\partial y} + \frac{\partial \hat{\phi}_i}{\partial \eta} \frac{\partial \eta}{\partial y} \right) \left(\frac{\partial \hat{\phi}_j}{\partial \varepsilon} \frac{\partial \varepsilon}{\partial y} + \frac{\partial \hat{\phi}_j}{\partial \eta} \frac{\partial \eta}{\partial y} \right) \right] + c \hat{\phi}_i \hat{\phi}_j \Big| J \Big| d\varepsilon d\eta \end{aligned} \quad (24.34)$$

donde el índice i y j varía de 1 a 3. En esta última usamos la regla de la cadena y $dxdy = |J| d\varepsilon d\eta$ para el cambio de variable en las integrales, aquí $|J| = \det T$, donde T está dado como en la Ec. (24.29). Para resolver $\int_{\Omega_e} f_{\Omega} \phi_j^{(e)} dxdy$ en cada Ω_e se genera las integrales

$$\begin{aligned} F_j^{(e)} &= \int_{\Omega_e} f_{\Omega} \phi_j^{(e)} dxdy \\ &= \int_{\hat{\Omega}} f_{\Omega} \hat{\phi}_j |J| d\varepsilon d\eta \end{aligned} \quad (24.35)$$

donde el índice i y j varía 1 a 3.

Para realizar el cálculo numérico de las integrales en el triángulo de referencia $\hat{\Omega}$, debemos conocer $\frac{\partial \phi_i}{\partial \varepsilon}$, $\frac{\partial \phi_i}{\partial \eta}$, $\frac{\partial \varepsilon}{\partial x}$, $\frac{\partial \varepsilon}{\partial y}$, $\frac{\partial \eta}{\partial x}$ y $\frac{\partial \eta}{\partial y}$, entonces realizando las operaciones necesarias a las Ec. (24.33) obtenemos

$$\begin{aligned} \frac{\partial \phi_1}{\partial \varepsilon} &= -1 & \frac{\partial \phi_1}{\partial \eta} &= -1 \\ \frac{\partial \phi_2}{\partial \varepsilon} &= 1 & \frac{\partial \phi_2}{\partial \eta} &= 0 \\ \frac{\partial \phi_3}{\partial \varepsilon} &= 0 & \frac{\partial \phi_3}{\partial \eta} &= 1 \end{aligned} \quad (24.36)$$

y también

$$\begin{aligned} \frac{\partial \varepsilon}{\partial x} &= \frac{(y_3^{(e)} - y_1^{(e)})}{2A_{\Omega_e}} & \frac{\partial \varepsilon}{\partial y} &= -\frac{(x_3^{(e)} - x_1^{(e)})}{2A_{\Omega_e}} \\ \frac{\partial \eta}{\partial x} &= -\frac{(y_2^{(e)} - y_1^{(e)})}{2A_{\Omega_e}} & \frac{\partial \eta}{\partial y} &= \frac{(x_2^{(e)} - x_1^{(e)})}{2A_{\Omega_e}} \end{aligned} \quad (24.37)$$

las cuales deberán de ser sustituidas en cada $\underline{K}_{ij}^{(e)}$ y $\underline{F}_j^{(e)}$ para calcular las integrales en el elemento Ω_e .

Suponiendo que Ω fue dividido en E elementos, estos elementos generan N nodos en total, de los cuales N_d son nodos desconocidos y N_c son nodos conocidos con valor γ_j , entonces el algoritmo de ensamble de la matriz \underline{K} y el vector \underline{F} se puede esquematizar como:

$$\begin{aligned} K_{i,j} &= (\phi_i, \phi_j) \quad \forall i = 1, 2, \dots, E, j = 1, 2, \dots, E \\ F_j &= (f_{\Omega}, \phi_j) \quad \forall j = 1, 2, \dots, E \\ \forall j &= 1, 2, \dots, N_d : \end{aligned}$$

$$b_j = b_j - \gamma_i K_{i,j} \quad \forall i = 1, 2, \dots, E$$

Así, se construye una matriz global en la cual están representados los nodos conocidos y los desconocidos, tomando sólo los nodos desconocidos

de la matriz \underline{K} formaremos una matriz \underline{A} , haciendo lo mismo al vector \underline{F} formamos el vector \underline{b} , entonces la solución al problema será la resolución del sistema de ecuaciones lineales $\underline{Ax} = \underline{b}$, este sistema puede resolverse usando por ejemplo el método de gradiente conjugado. El vector \underline{x} contendrá la solución buscada en los nodos desconocidos N_d .

25 Apéndice G: Nociones de Álgebra Lineal

En este apéndice se darán algunas definiciones que se usan a lo largo del presente trabajo, así como se detallan algunos resultados generales de álgebra lineal y análisis funcional (en espacios reales) que se enuncian sin demostración pero se indica en cada caso la bibliografía correspondiente donde se encuentran estas y el desarrollo en detalle de cada resultado.

25.1 Nociones de Álgebra Lineal

A continuación detallaremos algunos resultados de álgebra lineal, las demostraciones de los siguientes resultados puede ser consultada en [12].

Definición 131 Sea V un espacio vectorial y sea $f(\cdot) : V \rightarrow \mathbb{R}$, f es llamada funcional lineal si satisface la condición

$$f(\alpha v + \beta w) = \alpha f(v) + \beta f(w) \quad \forall v, w \in V \quad \text{y} \quad \alpha, \beta \in \mathbb{R}. \quad (25.1)$$

Definición 132 Si V es un espacio vectorial, entonces el conjunto V^* de todas las funcionales lineales definidas sobre V es un espacio vectorial llamado espacio dual de V .

Teorema 133 Si $\{v_1, \dots, v_n\}$ es una base para el espacio vectorial V , entonces existe una única base $\{v_1^*, \dots, v_n^*\}$ del espacio vectorial dual V^* llamado la base dual de $\{v_1, \dots, v_n\}$ con la propiedad de que $V_i^* = \delta_{ij}$. Por lo tanto V es isomorfo a V^* .

Definición 134 Sea $D \subset V$ un subconjunto del espacio vectorial V . El nulo de D es el conjunto $N(D)$ de todas las funcionales en V^* tal que se nulifican en todo el subconjunto D , es decir

$$N(D) = \{f \in V^* \mid f(v) = 0 \quad \forall v \in D\}. \quad (25.2)$$

Teorema 135 Sea V un espacio vectorial y V^* el espacio dual de V , entonces

- a) $N(D)$ es un subespacio de V^*
- b) Si $M \subset V$ es un subespacio de dimensión m , V tiene dimensión n , entonces $N(M)$ tiene dimensión $n - m$ en V^* .

Corolario 136 Si $V = L \oplus M$ (suma directa) entonces $V^* = N(L) \oplus N(M)$.

Teorema 137 Sean V y W espacios lineales, si $T(\cdot) : V \rightarrow W$ es lineal, entonces el adjunto T^* de T es un operador lineal $T^* : W^* \rightarrow V^*$ definido por

$$T^*(w^*)(u) = w^*(Tu). \quad (25.3)$$

Teorema 138 Si H es un espacio completo con producto interior, entonces $H^* = H$.

Definición 139 Si V es un espacio vectorial con producto interior y $T(\cdot) : V \rightarrow V$ es una transformación lineal, entonces existe una transformación asociada a T llamada la transformación auto-adjunta T^* definida como

$$\langle Tu, v \rangle = \langle u, T^*v \rangle. \quad (25.4)$$

Definición 140 Sea V un espacio vectorial sobre los reales. Se dice que una función $\tau(\cdot, \cdot) : V \times V \rightarrow \mathbb{R}$ es una forma bilineal sobre V , si para toda $x, y, z \in V$ y $\alpha, \beta \in \mathbb{R}$ se tiene

$$\begin{aligned} \tau(\alpha x + \beta y, z) &= \alpha\tau(x, z) + \beta\tau(y, z) \\ \tau(x, \alpha y + \beta z) &= \alpha\tau(x, y) + \beta\tau(x, z). \end{aligned} \quad (25.5)$$

Definición 141 Si $\tau(\cdot, \cdot)$ es una forma bilineal sobre V , entonces la función $q_\tau(\cdot) : V \rightarrow \mathbb{R}$ definida por

$$q_\tau(x) = \tau(x, x) \quad \forall x \in V \quad (25.6)$$

se le llama la forma cuadrática asociada a τ .

Notemos que para una forma cuadrática $q_\tau(\cdot)$ se tiene que $q_\tau(\alpha x) = |\alpha|^2 q_\tau(x) \forall x \in V$ y $\alpha \in \mathbb{R}$.

Definición 142 Sea $V \subset \mathbb{R}^n$ un subespacio, $P \in \mathbb{R}^n \times \mathbb{R}^n$

25.2 σ -Álgebra y Espacios Medibles

A continuación detallaremos algunos resultados conjuntos de espacios σ -álgebra, conjuntos de medida cero y funciones medibles, las demostraciones de los siguientes resultados puede ser consultada en [71] y [55].

Definición 143 Una σ -álgebra sobre un conjunto Ω es una familia ξ de subconjuntos de Ω que satisface

- $\emptyset \in \xi$
- Si $\psi_n \in \xi$ entonces $\bigcup_{n=1}^{\infty} \psi_n \in \xi$
- Si $\psi \in \xi$ entonces $\psi^c \in \xi$.

Definición 144 Si Ω es un espacio topológico, la familia de Borel es el conjunto σ -álgebra más pequeño que contiene a los abiertos del conjunto Ω .

Definición 145 Una medida μ sobre Ω es una función no negativa real valuada cuyo dominio es una σ -álgebra ξ sobre Ω que satisface

- $\mu(\emptyset) = 0$ y
- Si $\{\psi_n\}$ es una sucesión de conjuntos ajenos de ξ entonces

$$\mu\left(\bigcup_{n=1}^{\infty} \psi_n\right) = \sum_{n=1}^{\infty} \mu(\psi_n). \quad (25.7)$$

Teorema 146 Existe una función de medida μ sobre el conjunto de Borel de \mathbb{R} llamada la medida de Lebesgue que satisface $\mu([a, b]) = b - a$.

Definición 147 Una función $f : \Omega \rightarrow \mathbb{R}$ es llamada medible si $f^{-1}(U)$ es un conjunto medible para todo abierto U de \mathbb{R} .

Definición 148 Sea $E \subset \Omega$ un conjunto, se dice que el conjunto E tiene medida cero si $\mu(E) = 0$.

Teorema 149 Si α es una medida sobre el espacio X y β es una medida sobre el espacio Y , podemos definir una medida μ sobre $X \times Y$ con la propiedad de que $\mu(A \times B) = \alpha(A)\beta(B)$ para todo conjunto medible $A \in X$ y $B \in Y$.

Teorema 150 (Fubini)

Si $f(x, y)$ es medible en $X \times Y$ entonces

$$\int_{X \times Y} f(x, y) d\mu = \int_X \int_Y f(x, y) d\beta d\alpha = \int_Y \int_X f(x, y) d\alpha d\beta \quad (25.8)$$

en el sentido de que cualquiera de las integrales existe y son iguales.

Teorema 151 Una función f es integrable en el sentido de Riemann en Ω si y sólo si el conjunto de puntos donde $f(\underline{x})$ es no continua tiene medida cero.

Observación 13 Sean f y g dos funciones definidas en Ω , decimos que f y g son iguales salvo en un conjunto de medida cero si $f(x) \neq g(x)$ sólo en un conjunto de medida cero.

Definición 152 Una propiedad P se dice que se satisface en casi todos lados, si existe un conjunto E con $\mu(E) = 0$ tal que la propiedad se satisface en todo punto de E^c .

25.3 Espacios L^p

Las definiciones y material adicional puede ser consultada en [2], [54] y [55].

Definición 153 Una función medible $f(\cdot)$ (en el sentido de Lebesgue) es llamada integrable sobre un conjunto medible $\Omega \subset \mathbb{R}^n$ si

$$\int_{\Omega} |f| d\underline{x} < \infty. \quad (25.9)$$

Definición 154 Sea p un número real con $p \geq 1$. Una función $u(\cdot)$ definida sobre $\Omega \subset \mathbb{R}^n$ se dice que pertenece al espacio $L^p(\Omega)$ si

$$\int_{\Omega} |u(\underline{x})|^p d\underline{x} < \infty \quad (25.10)$$

es integrable.

Al espacio $L^2(\Omega)$ se le llama cuadrado integrable.

Definición 155 La norma $L^2(\Omega)$ se define como

$$\|u\|_{L^2(\Omega)} = \left(\int_{\Omega} |u(\underline{x})|^2 d\underline{x} \right)^{\frac{1}{2}} < \infty \quad (25.11)$$

y el producto interior en la norma $L^2(\Omega)$ como

$$\langle u, v \rangle_{L^2(\Omega)} = \int_{\Omega} u(\underline{x})v(\underline{x})d\underline{x}. \quad (25.12)$$

Definición 156 Si $p \rightarrow \infty$, entonces definimos al espacio $L^\infty(\Omega)$ como el espacio de todas las funciones medibles sobre $\Omega \subset \mathbb{R}^n$ que sean acotadas en casi todo Ω (excepto posiblemente sobre un conjunto de medida cero), es decir,

$$L^\infty(\Omega) = \{u \mid |u(x)| \leq k\} \quad (25.13)$$

definida en casi todo Ω , para algún $k \in \mathbb{R}$.

25.4 Distribuciones

La teoría de distribuciones es la base para definir a los espacios de Sobolev, ya que permiten definir las derivadas parciales de funciones no continuas, pero esta es coincidente con las derivadas parciales clásica si las funciones son continuas, para mayor referencia de estos resultados ver [2], [54] y [55]

Definición 157 Sea $\Omega \subset \mathbb{R}^n$ un dominio, al conjunto de todas las funciones continuas definidas en Ω se denotará por $C^0(\Omega)$, o simplemente $C(\Omega)$.

Definición 158 Sea u una función definida sobre un dominio Ω la cual es no cero sólo en los puntos pertenecientes a un subconjunto propio $K \subset \Omega$. Sea \bar{K} la clausura de K . Entonces \bar{K} es llamado el soporte de u . Decimos que u tiene soporte compacto sobre Ω si su soporte \bar{K} es compacto. Al conjunto de funciones continuas con soporte compacto se denota por $C_0(\Omega)$.

Definición 159 Sea \mathbb{Z}_+^n el conjunto de todas las n -dúplas de enteros no negativos, un miembro de \mathbb{Z}_+^n se denota usualmente por α ó β (por ejemplo $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$). Denotaremos por $|\alpha|$ la suma $|\alpha| = \alpha_1 + \alpha_2 + \dots + \alpha_n$ y por $D^\alpha u$ la derivada parcial

$$D^\alpha u = \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \dots \partial x_n^{\alpha_n}} \quad (25.14)$$

así, si $|\alpha| = m$, entonces $D^\alpha u$ denota la m -ésima derivada parcial de u .

Definición 160 Sea $C^m(\Omega)$ el conjunto de todas las funciones $D^\alpha u$ tales que sean funciones continuas con $|\alpha| = m$. Y $C^\infty(\Omega)$ como el espacio de funciones en el cual todas las derivadas existen y sean continuas en Ω .

Definición 161 El espacio $\mathcal{D}(\Omega)$ será el subconjunto de funciones infinitamente diferenciables con soporte compacto, algunas veces se denota también como $C_0^\infty(\Omega)$.

Definición 162 Una distribución sobre un dominio $\Omega \subset \mathbb{R}^n$ es toda funcional lineal continua sobre $\mathcal{D}(\Omega)$.

Definición 163 El espacio de distribuciones es el espacio de todas las funcionales lineales continuas definidas en $\mathcal{D}(\Omega)$, denotado como $\mathcal{D}^*(\Omega)$, es decir el espacio dual de $\mathcal{D}(\Omega)$.

Definición 164 Un función $f(\cdot)$ es llamada localmente integrable, si para todo subconjunto compacto $K \subset \Omega$ se tiene

$$\int_K |f(x)| dx < \infty. \quad (25.15)$$

Ejemplo de una distribución es cualquier función $f(\cdot)$ localmente integrable en Ω . La distribución F asociada a f se puede definir de manera natural como $F : \mathcal{D}(\Omega) \rightarrow \mathbb{R}$ como

$$\langle F, \phi \rangle = \int_{\Omega} f \phi dx \quad (25.16)$$

con $\phi \in \mathcal{D}(\Omega)$.

Si el soporte de ϕ es $K \subset \Omega$, entonces

$$|\langle F, \phi \rangle| = \left| \int_{\Omega} f \phi dx \right| = \left| \int_K f \phi dx \right| \leq \sup_{x \in K} |\phi| \int_{\Omega} |f(x)| dx \quad (25.17)$$

la integral es finita y $\langle F, \phi \rangle$ tiene sentido. Bajo estas circunstancias F es llamada una distribución generada por f .

Otro ejemplo de distribuciones es el generado por todas las funciones continuas acotadas, ya que estas son localmente integrables y por lo tanto generan una distribución.

Definición 165 Si una distribución es generada por funciones localmente integrables es llamada una distribución regular. Si una distribución no es generada por una función localmente integrable, es llamada distribución singular (ejemplo de esta es la delta de Dirac).

Es posible definir de manera natural el producto de una función y una distribución. Específicamente, si $\Omega \subset \mathbb{R}^n$, u pertenece a $C^\infty(\Omega)$, y si $f(\cdot)$ es

una distribución sobre Ω , entonces entenderemos uf por la distribución que satisface

$$\langle (uf), \phi \rangle = \langle f, u\phi \rangle \quad (25.18)$$

para toda $\phi \in \mathcal{D}(\Omega)$. Notemos que la anterior ecuación es una generalización de la identidad

$$\int_{\Omega} [u(x) f(x)] \phi(x) dx = \int_{\Omega} f(x) [u(x) \phi(x)] dx \quad (25.19)$$

la cual se satisface si f es localmente integrable.

Derivadas de Distribuciones Funciones como la delta de Dirac y la Heaviside no tienen derivada en el sentido ordinario, sin embargo, si estas funciones son tratadas como distribuciones es posible extender el concepto de derivada de tal forma que abarque a dichas funciones, para ello recordemos que:

Teorema 166 *La versión clásica del teorema de Green es dada por la identidad*

$$\int_{\Omega} u \frac{\partial v}{\partial x_i} d\mathbf{x} = \int_{\partial\Omega} u v n_i d\mathbf{s} - \int_{\Omega} v \frac{\partial u}{\partial x_i} d\mathbf{x} \quad (25.20)$$

que se satisface para todas las funciones u, v en $C^1(\bar{\Omega})$, donde n_i es la i -ésima componente de la derivada normal del vector n en la frontera $\partial\Omega$ de un dominio Ω .

Una versión de la Ec. (25.20) en una dimensión se obtiene usando la fórmula de integración por partes, quedando como

$$\int_a^b uv' dx = [uv] \Big|_a^b - \int_a^b vu' dx, \quad u, v \in C^1[a, b] \quad (25.21)$$

como un caso particular de la Ec. (25.20).

Este resultado es fácilmente generalizable a un resultado usando derivadas parciales de orden m de funciones $u, v \in C^m(\bar{\Omega})$ pero reemplazamos u por $D^\alpha u$ en la Ec. (25.20) y con $|\alpha| = m$, entonces se puede mostrar que:

Teorema 167 *Otra versión del teorema de Green es dado por*

$$\int_{\Omega} (D^\alpha u) v d\mathbf{x} = (-1)^{|\alpha|} \int_{\Omega} u D^\alpha v d\mathbf{x} + \int_{\partial\Omega} h(u, v) d\mathbf{s} \quad (25.22)$$

donde $h(u, v)$ es una expresión que contiene la suma de productos de derivadas de u y v de orden menor que m .

Ahora reemplazando v en la Ec. (25.22) por ϕ perteneciente a $\mathcal{D}(\Omega)$ y como $\phi = 0$ en la frontera $\partial\Omega$ tenemos

$$\int_{\Omega} (D^{\alpha}u) \phi d\underline{x} = (-1)^{|\alpha|} \int_{\Omega} u D^{\alpha} \phi d\underline{x} \quad (25.23)$$

ya que u es m -veces continuamente diferenciable, esta genera una distribución denotada por u , tal que

$$\langle u, \phi \rangle = \int_{\Omega} u \phi d\underline{x} \quad (25.24)$$

o, como $D^{\alpha}\phi$ también pertenece a $\mathcal{D}(\Omega)$, entonces

$$\langle u, D^{\alpha}\phi \rangle = \int_{\Omega} u D^{\alpha}\phi d\underline{x} \quad (25.25)$$

además, $D^{\alpha}u$ es continua, así que es posible generar una distribución regular denotada por $D^{\alpha}u$ satisfaciendo

$$\langle D^{\alpha}u, \phi \rangle = \int_{\Omega} (D^{\alpha}u) \phi d\underline{x} \quad (25.26)$$

entonces la Ec. (25.23) puede reescribirse como

$$\langle D^{\alpha}u, \phi \rangle = (-1)^{|\alpha|} \langle u, D^{\alpha}\phi \rangle, \quad \forall \phi \in \mathcal{D}(\Omega). \quad (25.27)$$

Definición 168 *La derivada de cualquier distribución $f(\cdot)$ se define como: La α -ésima derivada parcial distribucional o derivada generalizada de una distribución f es definida por una distribución denotada por $D^{\alpha}f$, que satisface*

$$\langle D^{\alpha}f, \phi \rangle = (-1)^{|\alpha|} \langle f, D^{\alpha}\phi \rangle, \quad \forall \phi \in \mathcal{D}(\Omega).$$

Nótese que si f pertenece a $C^m(\bar{\Omega})$, entonces la derivada parcial distribucional coincide con la derivada parcial α -ésima para $|\alpha| \leq m$.

Derivadas Débiles Supóngase que una función $u(\cdot)$ es localmente integrable que genere una distribución, también denotada por u , que satisface

$$\langle u, \phi \rangle = \int_{\Omega} u \phi dx \quad (25.28)$$

para toda $\phi \in \mathcal{D}(\Omega)$.

Además la distribución u posee derivada distribucional de todos los órdenes, en particular la derivada $D^\alpha u$ es definida por

$$\langle D^\alpha u, \phi \rangle = (-1)^{|\alpha|} \langle u, D^\alpha \phi \rangle, \quad \forall \phi \in D(\Omega). \quad (25.29)$$

por supuesto $D^\alpha u$ puede o no ser una distribución regular. Si es una distribución regular, entonces es generada por una función localmente integrable tal que

$$\langle D^\alpha u, \phi \rangle = \int_{\Omega} D^\alpha u(x) \phi(x) d\underline{x} \quad (25.30)$$

y se sigue que la función u y $D^\alpha u$ están relacionadas por

$$\int_{\Omega} D^\alpha u(x) \phi(x) d\underline{x} = (-1)^{|\alpha|} \int_{\Omega} u(x) D^\alpha \phi(x) d\underline{x} \quad (25.31)$$

para $|\alpha| \leq m$.

Definición 169 *Llamamos a la función (o más precisamente, a la equivalencia de clases de funciones) $D^\alpha u$ obtenida en la Ec. (25.31), la α -ésima derivada débil de la función u .*

Notemos que si u pertenece a $C^m(\bar{\Omega})$, entonces la derivada $D^\alpha u$ coincide con la derivada clásica para $|\alpha| \leq m$.

26 Bibliografía

Este texto es una recopilación de múltiples fuentes, mi aportación -si es que puedo llamarla así- es plasmarlo en este documento, en el que trato de dar coherencia a mi visión de los temas desarrollados.

En la realización de este texto se han revisado -en la mayoría de los casos indico la referencia, pero pude omitir varias de ellas, por lo cual pido una disculpa- múltiples páginas Web, artículos técnicos, libros, entre otros materiales bibliográficos, los más representativos y de libre acceso los pongo a su disposición en la siguiente liga:

Herramientas
<http://132.248.181.216/Herramientas/>

Referencias

- [1] K. Hutter y K. Jöhnk, *Continuum Methods of Physical Modeling*, Springer-Verlag Berlin Heidelberg New York, 2004. 8, 11
- [2] J. L. Lions y E. Magenes, *Non-Homogeneous Boundary Value Problems and Applications* Vol. I, Springer-Verlag Berlin Heidelberg New York, 1972. 44, 58, 69, 77, 233, 564, 565
- [3] A. Quarteroni y A. Valli, *Domain Decomposition Methods for Partial Differential Equations*. Clarendon Press Oxford, 1999. 9, 10, 236, 258
- [4] A. Quarteroni y A. Valli; *Numerical Approximation of Partial Differential Equations*. Springer, 1994. 296
- [5] B. Dietrich, *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*, Cambridge University, 2001. 296

- [6] B. F. Smith, P. E. Bjørstad, W. D. Gropp; *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996. [9](#), [10](#), [233](#), [296](#)
- [7] Fuzhen Zhang, *The Schur Complement and its Applications*, Springer, Numerical Methods and Algorithms, Vol. 4, 2005. [371](#)
- [8] B. I. Wohlmuth; *Discretization Methods and Iterative Solvers Based on Domain Decomposition*. Springer, 2003. [9](#), [120](#), [388](#)
- [9] L. F. Pavarino, A. Toselli; *Recent Developments in Domain Decomposition Methods*. Springer, 2003. [11](#), [296](#), [388](#)
- [10] M.B. Allen III, I. Herrera & G. F. Pinder; *Numerical Modeling in Science And Engineering*. John Wiley & Sons, Inc . 1988. [8](#), [28](#), [36](#), [78](#), [344](#), [383](#), [384](#), [387](#), [388](#), [409](#)
- [11] R. L. Burden y J. D. Faires; *Análisis Numérico*. Math Learning, 7 ed. 2004. [383](#), [409](#)
- [12] S. Friedberg, A. Insel, and L. Spence; *Linear Algebra*, 4th Edition, Prentice Hall, Inc. 2003. [383](#), [409](#), [561](#)
- [13] Y. Saad; *Iterative Methods for Sparse Linear Systems*. SIAM, 2 ed. 2000. [372](#), [378](#), [380](#), [384](#), [387](#), [391](#)
- [14] Y. Skiba; *Métodos y Esquemas Numéricos, un Análisis Computacional*. UNAM, 2005. [383](#), [388](#), [394](#), [409](#)
- [15] W. Gropp, E. Lusk, A. Skjelle, *Using MPI, Portable Parallel Programming With the Message Passing Interface*. Scientific and Engineering Computation Series, 2ed, 1999. [297](#), [302](#), [491](#), [492](#), [498](#)
- [16] I. Foster; *Designing and Building Parallel Programs*. Addison-Wesley Inc., Argonne National Laboratory, and the NSF, 2004. [297](#), [302](#), [491](#), [492](#), [498](#)
- [17] Jorge L. Ortega-Arjona, *Patterns for Parallel Software Design*, Wiley series in Software Design Patterns, 2010. [11](#), [297](#), [302](#), [491](#)

- [18] DDM Organization, *Proceedings of International Conferences on Domain Decomposition Methods*, 1988-2012. [11](#), [12](#), [13](#), [238](#), [239](#), [345](#)
<http://www.ddm.org> and <http://www.domain-decomposition.com>
- [19] Toselli, A., and Widlund O. *Domain decomposition methods- Algorithms and theory*, Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2005, 450p. [10](#), [11](#), [12](#), [258](#), [277](#), [320](#)
- [20] Farhat, C. and Roux, F. X. *A Method of Finite Element Tearing and Interconnecting and its Parallel Solution Algorithm*. Int. J. Numer. Meth. Engng., 32:1205-1227, 1991. [12](#)
- [21] Mandel J. & Tezaur R. *Convergence of a Substructuring Method with Lagrange Multipliers*, Numer. Math. 73 (1996) 473-487. [12](#)
- [22] Farhat C., Lesoinne M. Le Tallec P., Pierson K. & Rixen D. *FETI-DP a Dual-Primal Unified FETI method, Part 1: A Faster Alternative to the two-level FETI Method*, Int. J. Numer. Methods Engng. 50 (2001) 1523-1544. [12](#)
- [23] Farhat C., Lesoinne M., Pierson K. *A Scalable Dual-Primal Domain Decomposition Method*, Numer. Linear Algebra Appl. 7 (2000) 687-714.
- [24] Mandel J. & Tezaur R. *On the Convergence of a Dual-Primal Substructu-ring Method*, Numer. Math. 88(2001), pp. 5443-558. [12](#)
- [25] Mandel, J. *Balancing Domain Decomposition*. Comm. Numer. Meth. Engng., 9:233-241, 1993. [12](#)
- [26] Mandel J., & Brezina M., *Balancing Domain Decomposition for Problems with Large Jumps in Coefficients*, Math. Comput. 65 (1996) 1387-1401. [12](#)
- [27] Dohrmann C., *A Preconditioner for Substructuring Based on Constrained Energy Minimization*. SIAM J. Sci. Comput. 25 (2003) 246-258. [12](#), [348](#)
- [28] Mandel J. & Dohrmann C., *Convergence of a Balancing Domain Decomposition by Constraints and Energy Minimization*. Numer. Linear Algebra Appl. 10 (2003) 639-659.

- [29] Da Conceição, D. T. Jr., *Balancing Domain Decomposition Preconditioners for Non-symmetric Problems*, Instituto Nacional de Matemática pura e Aplicada, Agencia Nacional do Petróleo PRH-32, Rio de Janeiro, May. 9, 2006. [12](#), [17](#), [251](#), [280](#), [281](#), [323](#), [328](#)
- [30] J. Li and O. Widlund, *FETI-DP, BDDC and block Cholesky Methods*, Int. J. Numer. Methods Engrg. 66, 250-271, 2005. [12](#)
- [31] Herrera, I., *Theory of Differential Equations in Discontinuous Piecewise-Defined-Functions*, NUMER METH PART D.E., 23(3): 597-639, 2007 OI 10.1002/num.20182. [233](#), [279](#), [280](#)
- [32] Herrera, I. *New Formulation of Iterative Substructuring Methods Without Lagrange Multipliers: Neumann-Neumann and FETI*, NUMER METH PART D.E. 24(3) pp 845-878, May 2008 DOI 10.1002/num.20293.
- [33] Herrera I. and R. Yates, *Unified Multipliers-Free Theory of Dual Primal Domain Decomposition Methods*. NUMER. METH. PART D. E. 25(3): 552-581, May 2009, (Published on line May 13, 2008) DOI 10.1002/num.20359. [283](#)
- [34] Herrera, I. & Yates R. A., *The Multipliers-Free Domain Decomposition Methods*, NUMER. METH. PART D. E., 26(4): pp 874-905, July 2010. (Published on line: 23 April 2009, DOI 10.1002/num.20462) [15](#), [17](#), [247](#), [249](#), [253](#), [259](#), [260](#), [274](#), [283](#), [319](#), [320](#), [347](#)
- [35] Herrera, I., Yates R. A., *The multipliers-Free Dual Primal Domain Decomposition Methods for Nonsymmetric Matrices*. NUMER. METH. PART D. E. DOI 10.1002/num.20581 (Published on line April 28, 2010). [15](#), [17](#), [249](#), [253](#), [259](#), [260](#), [347](#)
- [36] Herrera, I., Carrillo-Ledesma A. and Alberto Rosas-Medina, *A Brief Overview of Non-Overlapping Domain Decomposition Methods*, Geofisica Internacional, Vol, 50, 4, October-December, 2011. [13](#), [14](#), [16](#), [238](#), [245](#), [251](#), [256](#), [259](#), [263](#), [295](#), [316](#), [344](#), [345](#)
- [37] Herrera, I. and Pinder, G. F., *Mathematical Modelling in Science and Engineering: An Axiomatic Approach*, Wiley, 243p., 2012. [8](#), [32](#), [36](#), [344](#)

- [38] Ismael Herrera and Alberto A. Rosas-Medina, *The Derived-Vector Space Framerwork and Four General purposes massively parallel DDM Algorithms*, Engineering Analysis with Boundary Elements, 20013, in press. [14](#), [16](#), [238](#), [245](#), [285](#), [295](#), [296](#), [316](#), [344](#), [345](#)
- [39] Antonio Carrillo-Ledesma, Herrera, I, Luis M. de la Cruz, *Parallel Algorithms for Computational Models of Geophysical Systems*, Geofísica Internacional, en prensa, 2013. [13](#), [16](#), [263](#), [285](#), [295](#), [296](#), [306](#), [316](#), [345](#)
- [40] Farhat Ch., Lesoinne M., Le Tallec P., Pierson K. and Rixen D. *FETI-DP: A Dual-Primal Unified FETI Method-Part I: A Faster Alternative to the Two Level FETI Method*. Internal. J. Numer. Methods Engrg., 50:1523-1544, 2001. [239](#)
- [41] Rixen, D. and Farhat Ch. *A Simple and Efficient Extension of a Class of Substructure Based Preconditioners to Heterogeneous Structural Mechanics Problems*. Internal. J. Numer. Methods Engrg., 44:489-516, 1999. [239](#)
- [42] J. Mandel, C. R. Dohrmann, and R. Tezaur, *An Algebraic Theory for Primal and Dual Substructuring Methods by Constraints*, Appl. Numer. Math., 54 (2005), pp. 167-193. [12](#), [13](#), [238](#), [239](#), [345](#)
- [43] A. Klawonn, O. B. Widlund, and M. Dryja, *Dual-primal FETI Methods for Three-Dimensional Elliptic Problems with Heterogeneous Coefficients*, SIAM J. Numer. Anal., 40 (2002), pp. 159-179. [12](#), [13](#), [238](#), [239](#), [345](#)
- [44] Alberto Rosas Medina, *Métodos de Estabilización para Problemas de Advección-Difusión*, Trabajo de Investigación para Sustentar el Examen de Candidatura al Doctorado, Postgrado en Ciencias de la Tierra, UNAM, 2011. [323](#), [325](#), [326](#), [327](#)
- [45] Iván Germán Contreras Trejo, *Métodos de Precondicionamiento para Sistemas de Ecuaciones Diferenciales Parciales*, Trabajo de Tesis Doctoral en Proceso, Postgrado en Ciencias e Ingeniería de la Computación, UNAM, 2012. [328](#), [356](#)
- [46] Klawonn A. and Widlund O.B., *FETI and Neumann-Neumann Iterative Substructuring Methods: Connections and New Results*. Comm. Pure and Appl. Math. 54(1): 57-90, 2001. [348](#)

- [47] X. O. Olivella, C. A. de Sacribar; *Mecánica de Medios Continuos para Ingenieros*. Ediciones UPC, 2000. 36
- [48] K. Hutter & K. Jöhnk; *Continuum Methods of Physical Modeling*. Springer-Verlag Berlin Heidelberg New York 2004. 36
- [49] Tezaur R., *Analysis of Lagrange Multipliers Based Domain Decomposition*. P.H. D. Thesis, University of Colorado, Denver, 1998. 348
- [50] Herrera I. & Rubio E., *Unified Theory of Differential Operators Acting on Discontinuous Functions and of Matrices Acting on Discontinuous Vectors*, 19th International Conference on Domain Decomposition Methods, Zhangjiajie, China 2009. (Oral presentation). Internal report #5, GMMC-UNAM, 2011. 258, 270
- [51] Valeri I. Agoshkov, *Poincaré-Steklov Operators and Domain Decomposition Methods in Finite Dimensional Spaces*. First International Symposium on Domain Decomposition Methods for Partial Differential Equations, pages 73-112, Philadelphia, PA, 1988. SIAM. Paris, France, January 7-9, 1987. 236, 270
- [52] Toselli, A., *FETI Domain Decomposition Methods for Escalar Advection-Diffusion Problems*. Computational Methods Appl. Mech. Engrg. 190. (2001), 5759-5776. 17, 325, 326, 328
- [53] C.T. Keller, *Iterative Methods for Linear and Nonlinear Equations*, Society for Industrial and Applied Mathematics, 1995. 391, 394
- [54] P.G. Ciarlet, J. L. Lions; *Handbook of Numerical Analysis, Vol. II*. North-Holland, 1991. 44, 564, 565
- [55] B. D. Reddy; *Introductory Functional Analysis - With Applications to Boundary Value Problems and Finite Elements*. Springer 1991. 44, 58, 69, 77, 562, 564, 565
- [56] Manoj Bhardwaj, David Day, Charbel Farhat, Michel Lesoinne, Kendall Pierson, and Daniel Rixen. *Application of the PETI Method to ASCI Problems: Scalability Results on One Thousand Processors and Discussion of Highly Heterogeneous Problems*. Intemat. J. Numer. Methods Engrg., 47:513-535, 2000. 11

- [57] Zdeněk Dostál and David Hordk. *Scalability and FETI Based Algorithm for Large Discretized Variational Inequalities*. Math. Comput. Simulation, 61(3-6): 347-357, 2003. MODELLING 2001 (Pilsen). [11](#), [12](#), [13](#), [238](#), [239](#), [345](#)
- [58] A. Toselli, O. Widlund; *Domain Decomposition Methods - Algorithms and Theory*. Springer, 2005. [120](#)
- [59] I. Herrera, R. Yates y E. Rubio. "More Efficient Procedures for Applying Collocation". Advances in Engineering Software 38 (2007) 657-667.
- [60] I. Herrera. "Theory of Differential Equations in Discontinuous Piecewise-Defined Functions". Wiley InterScience, 2006. [219](#)
- [61] I. Herrera. "New Formulation of Iterative Substructuring Methods Without Lagrange Multipliers Neumann-Neumann and FETI". Wiley InterScience, 2007. [219](#)
- [62] I. Herrera y R. Yates, "Unified Multipliers-Free Theory of Dual-Primal Domain Decomposition Methods", Numerical Methods For Partial Differential Equations, Wiley InterScience, 2008. [147](#), [219](#)
- [63] B. Cockburn, G. E. Karniadakis y C. W. Shu; Discontinuous Galerkin Methods: Theory, Computation and Applications. Springer, 2000. [82](#), [83](#)
- [64] C. Farhat, I Harari, L. P. Franca; The Discontinuous Enrichment Method. Computer Methods in Applied mechanics and Engineering, 190 (6455-6479), 2001. [93](#)
- [65] F. Brezzi y M. Fortin; Mixed and Hybrid Finite Element Methods, Springer, 1991. [44](#), [104](#)
- [66] J. II. Bramble, J. E. Pasciak and A. II Schatz. *The Construction of Preconditioners for Elliptic Problems by Substructuring*. I. Math. Comput., 47, 103-134,1986.
- [67] Charbel Farhat, Michel Lesoinne, and Kendall Pierson. *A Scalable Dual-Primal Domain Decomposition Method*. Numer. Linear Algebra Appl., 7(7-8):687-714, 2000. [11](#), [12](#), [13](#), [238](#), [239](#), [345](#)

- [68] Yannis Fragakis and Manolis Papadrakakis, *The Mosaic of High Performance Domain Decomposition Methods for Structural Mechanics: Formulation, Interrelation and Numerical Efficiency of Primal and Dual Methods*. Comput. Methods Appl. Mech. Engrg, 192(35-36):3799-3830, 2003. [11](#), [12](#), [13](#), [239](#), [345](#)
- [69] Kendall H. Pierson, *A family of Domain Decomposition Methods for the Massively Parallel Solution of Computational Mechanics Problems*. PhD thesis, University of Colorado at Boulder, Aerospace Engineering, 2000. [11](#)
- [70] Manoj Bhardwaj, Kendall H. Pierson, Garth Reese, Tim Walsh, David Day, Ken Alvin, James Peery, Charbel Farhat, and Michel Lesoinne. Salinas, *A Scalable Software for High Performance Structural and Mechanics Simulation*. In ACM/IEEE Proceedings of SC02: High Performance Networking and Computing. Gordon Bell Award, pages 1-19, 2002. [11](#), [12](#), [13](#), [239](#), [345](#)
- [71] W. Rudin; *Principles of Mathematical Analysis*. McGraw-Hill International Editions, 1976. [562](#)
- [72] Klawonn, A.; Rheinbach, O., *Highly Scalable Parallel Domain Decomposition Methods with an Application to Biomechanics*, Journal of Applied Mathematics and Mechanics 90 (1): 5-32, doi:10.1002/zamm.200900329. [11](#), [12](#), [13](#), [238](#), [239](#), [345](#)
- [73] Petter E. Bjørstad and Morten Skogen. *Domain Decomposition Algorithms of Schwarz Type, Designed for Massively Parallel Computers*. In David E. Keyes, Tony F. Chan, Gerard A. Meurant, Jeffrey S. Scroggs, and Robert G. Voigt, editors. Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, pages 362-375, Philadelphia, PA, 1992. SIAM. Norfolk, Virginia, May 6-8, 1991. [11](#)
- [74] Yau Shu Wong and Guangrui Li. *Exact Finite Difference Schemes for Solving Helmholtz Equation at any Wavenumber*. International Journal of Numerical Analysis and Modeling, Series B, Volume 2, Number 1, Pages 91-108, 2011. [322](#)

- [75] Holger Brunst, Bernd Mohr. *Performance Analysis of Large-Scale OpenMP and Hybrid MPI/OpenMP Applications with Vampir NG*. IWOMP 2005: 5-14. **313, 499**
- [76] S.J. Pennycook, S.D. Hammond, S.A. Jarvis and G.R. Mudalige, *Performance Analysis of a Hybrid MPI/CUDA Implementation of the NAS-LU Benchmark*. ACM SIGMETRICS Perform. Eval. Rev. 38 (4). ISSN 0163-5999, (2011). **313, 499**
- [77] Doxygen, *Generate Documentation from Source Code*. **298**
<http://www.stack.nl/~dimitri/doxygen/>
- [78] XMPI, *A Run/Debug GUI for MPI*. **306, 491**
<http://www.lam-mpi.org/Software/xmpi/>
- [79] VAMPIR, *Performance Optimization*. **306**
<http://www.vampir.eu/>
- [80] Handbook of Floating-Point Arithmetic 2010th Edition, Jean-Michel Muller, Nicolas Brisebarre, Et alii, Birkhäuser, 2010.
- [81] What Every Computer Scientist Should Know About Floating-Point Arithmetic, David Goldberg, ACM Computing Surveys, Vol 23, No 1, March 1991
- [82] 754-2008 - IEEE Standard for Floating-Point Arithmetic. IEEE. 29 de agosto de 2008. ISBN 978-0-7381-5752-8. doi:10.1109/IEEESTD.2008.4610935. (NB. Superseded by IEEE Std 754-2019, a revision of IEEE 754-2008.)
- [83] Stochastic Rounding and Its Probabilistic Backward Error Analysis, Michael P. Connolly, Nicholas J. Higham, Methods and Algorithms for Scientific Computing, SIAM Journal on Scientific Computing Vol. 43, Iss. 1 (2021)
- [84] IEEE, <https://www.ieee.org/>
- [85] Intel, <https://www.intel.la>
- [86] AMD, <https://www.amd.com>

[87] ARM, <https://www.arm.com/>

[88] Cerebras, <https://www.cerebras.net/>



Declaro terminado este trabajo sufrido, ideado y llevado a cabo entre los años 2007 al 2025, aún y a pesar de impedimentos tales como: la mala suerte, la desventura, el infortunio, la incomprensión, la gripe, el Covid-19, las horas de frío y de calor, la tristeza, la desesperanza, el cansancio, el presente, el pasado y mi futuro, el que dirán, la vergüenza, mis propias incapacidades y limitaciones, mis aversiones, mis temores, mis dudas y en fin, todo aquello que pudiera ser tomado por mi, o por cualquiera, como obstáculo en este tiempo de mentiras, verdades, de incredulidad e ignorancia o negación de la existencia real y física de la mala fe.

Atentamente

Antonio Carrillo Ledesma

