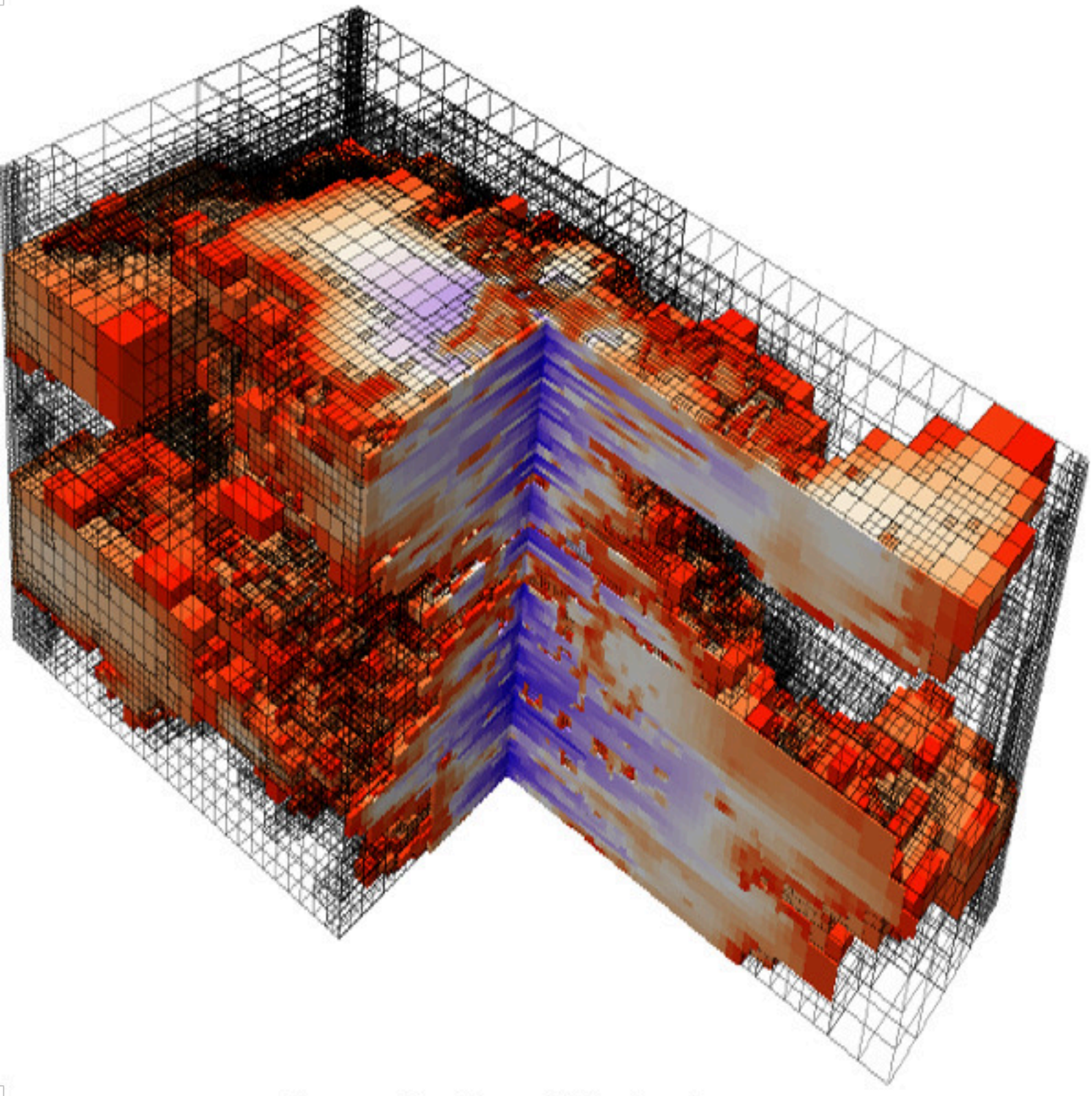
Introducción al Método de Elemento Finito



Antonio Carrillo Ledesma

Introducción al Método de Elemento Finito

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 1.0α[1]

[1]El 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

Los sistemas continuos -sistemas físicos macroscópicos (véase [10])-, 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 contienen un gran número de grados de libertad[1].

Los modelos matemáticos de los sistemas continuos (véase [74] y [19]) 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-

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

reducen el problema a la generación y solución de un -cada vez más grande- sistema algebraico de ecuaciones (véase [13]). 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 [16] y [2]).

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

En la actualidad, los sistemas computacionales paralelos son ubicuos. En ellos es posible encontrar más de una unidad de procesamiento, conocidas como Núcleo (Core). El número de Cores es creciente conforme avanza la tecnología, esto tiene una gran importancia en el desarrollo eficiente de algoritmos que resuelvan sistemas algebraicos en implementaciones paralelas. Actualmente la gran mayoría de los algoritmos desarrollados son algoritmos secuenciales y su implantación en equipos paralelos no es óptima, pero es una práctica común usar diversas técnicas de pseudoparalelización -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 [16] y [2]). La idea básica detrás de los métodos de descomposición de dominio es que en lugar de resolver un enorme problema sobre un dominio, puede

[2] Como pueden ser las bibliotecas ATLAS -http://math-atlas.sourceforge.net- y HYPRE -https://computation.llnl.gov/projects/hypre-scalable-linear-solvers-multigrid-methods- entre muchas otras.

ser conveniente -o necesario- resolver múltiples problemas -usando Diferencias Finitas, Volumen Finito y Elemento Finito- de tamaño menor sobre un solo subdominio un cierto número de veces. Mucho del trabajo en la descomposición de dominio se relaciona con la selección de subproblemas que aseguren que la razón de convergencia del nuevo método iterativo sea rápida. En otras palabras, los métodos de descomposición de dominio proveen precondicionadores a priori que puedan acelerarse por métodos en el espacio de Krylov (véase [31]).

1.2 Objetivos

El presente trabajo tiene por objetivo el hacer una introducción al Método de Elemento Finito (FEM) y su implementación computacional, este método es de carácter general que permite la resolución aproximada de ecuaciones diferenciales en derivadas parciales definidas en un dominio finito. Es de una gran sencillez conceptual y constituye un procedimiento muy adecuado para la resolución de una ecuación en una, dos o tres dimensiones.

El método consiste en una aproximación de las derivadas parciales por expresiones algebraicas con los valores de la variable dependiente en un número finito de puntos seleccionados en el dominio. Como resultado de la aproximación, la ecuación diferencial parcial que describe el problema es reemplazada por un número finito de ecuaciones algebraicas, en términos de los valores de la variable dependiente en los puntos seleccionados. El valor de los puntos seleccionados se convierten en las incógnitas. La solución del sistema de ecuaciones algebraico permite obtener la solución aproximada en cada punto seleccionado de la malla.

1.3 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/

Agradecemos 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, estan 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 na-turales 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, porqué 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 estan 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.

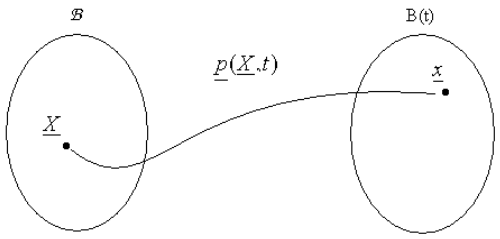


Figura 1: Representación del movimiento de partículas de un cuerpo $\mathcal{B}$, para un tiempo dado.

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 $\underline{p}(\underline{X}, 0) \equiv \underline{X}$.

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

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

el vector posición $\underline{p}(\underline{X}, t)$ es función del vector tridimensional $\underline{X}$ y del tiempo. Si fijamos el tiempo t, $\underline{p}(\underline{X}, t)$ define una transformación del espacio Euclidiano $\mathbb{R}^3$ 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

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

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 $\phi : \mathcal{B} \to \mathbb{R}^1$, 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 $\psi : \mathcal{B}(t) \to \mathbb{R}^1$ a la cual se denomina representación Euleriana de la función considerada. Estas dos representaciones de una misma propiedad están relacionadas por la siguiente identidad

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

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

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

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

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

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

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

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

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

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

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

o, más brevemente,

$$\frac{D\psi}{Dt} = \frac{\partial \psi}{\partial t} + \underline{\mathrm{v}} \cdot \nabla \psi \tag{2.9}$$

utilizando esta notación, se puede escribir

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

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

$$\frac{D\underline{\mathrm{v}}}{Dt} = \frac{\partial \underline{\mathrm{v}}}{\partial t} + \underline{\mathrm{v}} \cdot \nabla \underline{\mathrm{v}} \tag{2.11}$$

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

$$\frac{D\mathrm{v}_i}{Dt} = \frac{\partial \mathrm{v}_i}{\partial t} + \underline{\mathrm{v}} \cdot \nabla \mathrm{v}_i. \tag{2.12}$$

Desde luego, la aceleración, en representación Lagrangiana es simplemente

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

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

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

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, porqué 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 [19]. 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

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

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 formula 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

$$\Delta\mathbb{E} = \mathbb{P} + \mathbb{I} \tag{2.16}$$

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

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

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

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

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

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

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 porqué 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 [74].

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*

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

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

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

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

ii) La ecuación

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

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

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

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

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

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

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

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

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

porqué 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

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

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

$$\nabla \cdot \underline{v} = 0. \tag{2.28}$$

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

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

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

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

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

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

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 'isocorico'. Es oportuno mencionar que si bien es cierto que cuando un fluido tiene la propiedad de ser incompresible, todos sus movimientos son isocoricos, lo inverso no es cierto: un fluido compresible en ocasiones puede efectuar movimientos isocoricos.

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

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

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 estan fijas en el espacio; así, $\underline{v}_\Sigma = 0$ y la Ec. (2.32) se reduce a

$$[\phi \underline{v}] \cdot \underline{n} = 0 \tag{2.33}$$

o, de otra manera

$$\phi_+ v_{n_+} = \phi_- v_{n_-}. \tag{2.34}$$

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 lado 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

$$\underline{U} = \phi \underline{v} \tag{2.35}$$

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

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

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 [74], [19], [73] y [71].

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 estan 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

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

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

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

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

$$u(x, y) = f(mx + y) \tag{3.3}$$

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

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

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

$$\left(am^2 + bm + c\right) f'' = 0 \tag{3.5}$$

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

$$am^2 + bm + c = 0 \tag{3.6}$$

resolviendo esta ecuación cuadrática para m obtenemos las dos soluciones

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

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 $m_1 x + y$ ó $m_2 x + y$ resuelven a la Ec. (3.2). Por lo tanto la solución general de la Ec. (3.2) es

$$u(x, y) = \mathcal{F}\left(m_1 x + y\right) + \mathcal{G}(m_2 x + y) \tag{3.8}$$

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

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

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

$$u(x, y) = \mathcal{F}(m_1 x + y) + y\mathcal{G}(m_1 x + y) \tag{3.10}$$

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

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

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

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

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, $m_2 = m_1$. La solución general puede ser escrita en la forma

$$u(x, y) = \mathcal{F}(m_1 x + y) + \mathcal{G}(m_2 x + y) \tag{3.13}$$

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

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

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

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

y consideremos también la ecuación homogénea asociada a este operador

$$\mathcal{L}u = 0 \tag{3.16}$$

además, sea $\underline{x} \in \Omega$ un punto del espacio Euclidiano y $V(x)$ 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 $V(\underline{x})\backslash\Sigma$. 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

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

para alguna pareja $i, j = 1, ..., n$. 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*

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

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

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

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)

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

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 $\partial\Omega$

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

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 $\partial\Omega$

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

3) **Condiciones tipo Robin**

esta condición es una combinación de las dos anteriores

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

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

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 liguen 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 $x \equiv (x_1, ..., x_n)$, mientras que cuando se considera cualquiera de las otras dos, u es una función del vector $x \equiv (x_1, ..., x_n, t)$. 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 $n \to n+1$ 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 $\frac{\partial u}{\partial \underline{n}} = g_\partial$ 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

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

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

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

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 [11], [7], [1] y [9].

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 $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$. Denotaremos por $|\alpha|$ la suma $|\alpha| = \alpha_1 + \alpha_2 + ... + \alpha_n$ y por $D^\alpha u$ la derivada parcial*

$$D^\alpha u = \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} ... \partial x_n^{\alpha_n}} \tag{4.1}$$

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

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

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

$$\mathcal{L}u = f, \tag{4.3}$$

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

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

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

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

$$\sum_{|\alpha|,|\beta|=m} a_{\alpha\beta}\left(\underline{x_o}\right) \xi^{\alpha+\beta} \neq 0 \qquad \forall \xi \neq 0; \tag{4.5}$$

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

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

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

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

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

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

$$\sum_{i,j=1}^{n} a_{ij}(\underline{x_0})\xi_i\xi_j \neq 0 \qquad \forall \xi \neq 0 \tag{4.8}$$

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

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

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*

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

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*

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

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

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

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*

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

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*

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

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 $(\xi_1,...,\xi_n)$ tal que el segmento $\partial\Omega \cap B(x_0,\varepsilon)$ pueda expresarse como una función

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

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 $C^\infty(\overline{\Omega})$.

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

$$\|u - f\|_{H^m} < \varepsilon \tag{4.16}$$

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 $C^\infty(\overline{\Omega})$.*

Otra propiedad, se tiene al considerar a cualquier miembro de $u \in H^m(\Omega)$, este puede ser identificado con una función en $C^m(\overline{\Omega})$, 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 $f : X \to Y$ 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*

$$\|f(u)\|_Y = \|u\|_Y \leq K \|u\|_X \tag{4.17}$$

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 $C^k(\overline{\Omega})$, es decir, hay un miembro que pertenece a $C^k(\overline{\Omega})$. Además, la inmersión

$$H^m(\Omega) \subset C^k(\overline{\Omega}) \tag{4.18}$$

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 actué en $\overline{\Omega}$ tal que $tr(u) = u_{|\partial\Omega}$.

El siguiente lema nos dice que el operador $tr(\cdot)$ es un operador lineal continuo de $C^1\left(\overline{\Omega}\right)$ 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*

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

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

Ahora, para el caso $tr(\cdot) : H^1(\Omega) \to L^2(\partial\Omega)$, 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 $tr(\cdot) : H^1(\Omega) \to L^2(\partial\Omega)$, tal que

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

con la propiedad que si $u \in C^1\left(\overline{\Omega}\right)$, 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

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

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

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

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

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

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

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

Teorema 21 *(Desigualdad de Poincaré-Friedrichs)*

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

$$\int_\Omega |u|^2 \, dx \leq C \int_\Omega |\nabla u|^2 \, dx \tag{4.25}$$

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*

$$|u|_m^2 = \sum_{|\alpha|=m} \int_\Omega |D^\alpha u|^2 \, dx. \tag{4.26}$$

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*

$$\|u\|_{L^2}^2 \leq C\,|u|_m^2 \tag{4.27}$$

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*

$$q = \sum_{|\alpha|<m} D^\alpha q_\alpha \tag{4.28}$$

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 $\gamma_0 : H^1(\Omega) \to L^2(\Omega)$ lineal y continuo, tal que $\gamma_0 v = tr(v)$ 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

$$H^1(\partial\Omega) \subset \gamma_0\left(H^1(\Omega)\right) \subset L^2(\partial\Omega) \equiv H^0(\partial\Omega) \tag{4.29}$$

donde cada inclusión es estricta.

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

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

con

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

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

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

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

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

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

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

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

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

4.2.3 Espacios $H(div, \Omega)$

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

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

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

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

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

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

Cuando $H(div,\Omega)$ es equipada con la norma $\|\cdot\|^2_{H(div,\Omega)}$ 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 $\underline{q}\in H(div,\Omega)$, entonces los vectores de $H(div,\Omega)$ 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

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

para toda $\underline{q}\in H(div,\Omega)$ 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 $\left\langle v,\underline{q}\cdot\underline{n}\right\rangle$.

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

$$\int_\Omega div\ \underline{q}v dx+\int_\Omega \underline{q}\cdot grad\ v dx=\left\langle v,\underline{q}\cdot\underline{n}\right\rangle \tag{4.40}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Antes de poder ver las condiciones bajo las cuales se garantice 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*

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

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 (82) a la integral

$$\int_\Omega v \mathcal{L} u d\underline{x} \tag{4.48}$$

obtenemos

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

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 (82) y obtenemos, para i y j fijos

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

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

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

y

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

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 $g_0, g_1, ..., g_{m-1} = 0$ (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

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

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 $B_j : H^s(\Omega) \to L^2(\Omega)$, entonces

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

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

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

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

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

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

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

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

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

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

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

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

De esta forma tenemos que

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

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

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

y definimos

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

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

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

Definiendo para el operador $\mathcal{L}^*$

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

y

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

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 [1] y [11].

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

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

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

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

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

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

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

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

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*

$$w\mathcal{L}u - u\mathcal{L}^*w = \nabla \cdot \underline{\mathfrak{D}}\,(u, w) \tag{3.1}$$

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

$$\frac{d^n u}{dx^n} = u \tag{4.70}$$

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

$$\mathcal{L}u = u \tag{4.71}$$

de la definición de operador adjunto tenemos que

$$w\mathcal{L}u = u\mathcal{L}^*w + \nabla \cdot \underline{\mathfrak{D}}\,(u, w) \tag{4.72}$$

entonces el término izquierdo es

$$w\mathcal{L}u = wu \tag{4.73}$$

de aquí

$$u\mathcal{L}^*w = uw \tag{4.74}$$

por lo tanto el operador adjunto formal es

$$\mathcal{L}^*w = w \tag{4.75}$$

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

$$\mathcal{L}u = c\frac{du}{dx} \tag{4.76}$$

de la definición de operador adjunto tenemos

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

desarrollando el lado izquierdo

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

por lo tanto, el operador adjunto formal es

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

y los términos de frontera son

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

C) **Operador Elíptico**

El operador elíptico más sencillo es el Laplaciano

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

de la ecuación del operador adjunto formal tenemos

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

entonces, el operador adjunto formal es

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

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

$$\underline{\mathfrak{D}}\,(u, w) = u \frac{\partial w}{\partial x_i} - w \frac{\partial u}{\partial x_i}. \tag{4.84}$$

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

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

de la definición de operador adjunto formal tenemos que

$$w \mathcal{L} u = u \mathcal{L}^* w + \nabla \cdot \underline{\mathfrak{D}}\,(u, w) \tag{4.86}$$

desarrollando el lado derecho de la ecuación anterior

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

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

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

reordenando términos se tiene

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

por lo tanto, el operador adjunto formal es

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

y el término correspondiente a valores en la frontera es

$$\underline{\mathfrak{D}}\left(u,w\right)=\underline{\underline{a}}\cdot\left(u\nabla w-w\nabla u\right)+\left(w\underline{b}u\right). \tag{4.91}$$

E) **La ecuación bi-armónica**

Consideremos el operador diferencial bi-armónico

$$\mathcal{L}u=\Delta^2 u \tag{4.92}$$

entonces se tiene que

$$w\mathcal{L}u=u\mathcal{L}^*w+\nabla\cdot\underline{\mathfrak{D}}\left(u,w\right) \tag{4.93}$$

desarrollemos el término del lado derecho

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

utilizando la igualdad de divergencia

$$\nabla\cdot\left(sV\right)=s\nabla\cdot V+V\cdot\nabla s \tag{4.95}$$

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

$$\begin{aligned} & w\nabla\cdot\left(\nabla\Delta u\right) \\ = \;& \nabla\cdot\left(w\nabla\Delta u\right)-\nabla\Delta u\cdot\nabla w \end{aligned} \tag{4.96}$$

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

$$\begin{aligned} & \nabla\cdot\left(w\nabla\Delta u\right)-\nabla\Delta u\cdot\nabla w \\ = \;& \nabla\cdot\left(w\nabla\Delta u\right)+\Delta u\nabla\cdot\nabla w-\nabla\cdot\left(\Delta u\nabla w\right) \\ = \;& \Delta w\nabla\cdot\nabla u+\nabla\cdot\left(w\nabla\Delta u-\Delta u\nabla w\right) \end{aligned} \tag{4.97}$$

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

$$\begin{aligned} & \Delta w\nabla\cdot\nabla u+\nabla\cdot\left(w\nabla\Delta u-\Delta u\nabla w\right) \\ = \;& \nabla\cdot\left(\Delta w\nabla u\right)-\nabla u\cdot\nabla\left(\Delta w\right)+\nabla\cdot\left(w\nabla\Delta u-\Delta u\nabla w\right) \\ = \;& -\nabla u\cdot\nabla\left(\Delta w\right)+\nabla\cdot\left(w\nabla\Delta u+\Delta w\nabla u-\Delta u\nabla w\right) \end{aligned} \tag{4.98}$$

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

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

reordenando términos

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

entonces se tiene que el operador adjunto formal es

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

y los términos de frontera son

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

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*

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

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

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

de la definición de operador adjunto formal tenemos que

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

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

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

utilizando la igualdad de divergencia tenemos

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

reordenado términos tenemos que la ecuación anterior es

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

en notación simbólica tenemos que

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

por lo tanto el operador adjunto formal es

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

y los términos de frontera son

$$\underline{\mathfrak{D}}\left(\underline{u},\underline{w}\right)=\underline{u}\cdot\underline{\underline{\underline{C}}}:\nabla\underline{w}-\underline{w}\cdot\underline{\underline{\underline{C}}}:\nabla\underline{u} \tag{4.111}$$

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

B) **Métodos Mixtos a la Ecuación de Laplace**

Operador Laplaciano

$$\underline{\underline{\mathcal{L}}}\ \underline{u}=\Delta\underline{u}=\underline{f} \tag{4.112}$$

escrito en un sistema de ecuaciones se obtiene

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

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

$$\underline{u} \equiv \left\{\underline{p}, u\right\} \text{ y } \underline{w} = \left\{\underline{q}, w\right\} \tag{4.114}$$

ahora el operador diferencial vector-valuado es el siguiente

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

entonces

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

utilizando la definición de operador adjunto

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

haciendo el desarrollo del término izquierdo se tiene que

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

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

$$\begin{aligned} & \underline{qp} - \underline{q}\nabla\cdot u + w\nabla\cdot\underline{p} \\ = \;& \underline{qp} + u\nabla\cdot\underline{q} - \nabla\cdot\left(\underline{q}u\right) - \underline{p}\cdot\nabla w + \nabla\cdot\left(w\underline{p}\right) \\ = \;& \underline{p}\left(\underline{q} - \nabla w\right) + u\nabla\cdot\underline{q} + \nabla\cdot\left(w\underline{p} - u\underline{q}\right) \end{aligned} \tag{4.119}$$

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

$$\begin{aligned} & \underline{p}\left(\underline{q}-\nabla w\right)+u\nabla\cdot\underline{q}+\nabla\cdot\left(w\underline{p}-u\underline{q}\right) \\ = & \left[\begin{array}{c}\underline{p}\\ u\end{array}\right]\left[\begin{array}{c}\underline{q}-\nabla w\\ w\end{array}\right]+\nabla\cdot\left(w\underline{p}-u\underline{q}\right) \\ = & \left[\begin{array}{c}\underline{p}\\ u\end{array}\right]\cdot\left[\begin{array}{cc}1 & -\nabla\\ \nabla\cdot & 0\end{array}\right]\cdot\left[\begin{array}{c}\underline{q}\\ w\end{array}\right]+\nabla\cdot\left(w\underline{p}-u\underline{q}\right) \end{aligned} \tag{4.120}$$

por lo tanto, el operador adjunto formal es

$$\begin{aligned} \underline{\underline{\mathcal{L}}}^{*}\underline{w} &= \left[\begin{array}{cc}1 & -\nabla\\ \nabla\cdot & 0\end{array}\right]\cdot\left[\begin{array}{c}\underline{q}\\ w\end{array}\right] \\ &= \left[\begin{array}{c}\underline{q}-\nabla w\\ \nabla\cdot\underline{q}\end{array}\right] \end{aligned} \tag{4.121}$$

y el término correspondiente a valores en la frontera es

$$\underline{\mathfrak{D}}\left(\underline{u},\underline{w}\right)=w\underline{p}-u\underline{q}. \tag{4.122}$$

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

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

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

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

ahora el operador diferencial vector-valuado es el siguiente

$$\underline{\underline{\mathcal{L}}}\underline{U}=\left[\begin{array}{cc}-\Delta & \nabla\\ -\nabla\cdot & 0\end{array}\right]\cdot\left[\begin{array}{c}\underline{u}\\ p\end{array}\right] \tag{4.125}$$

el desarrollo se hará en notación indicial, entonces tenemos que

$$\underline{W}\underline{\underline{\mathcal{L}}}\underline{U}=\left\{\begin{array}{c}-\underline{w}\Delta\underline{u}+\underline{w}\nabla p\\ -q\nabla\cdot\underline{u}\end{array}\right. \tag{4.126}$$

usando notación indicial se obtiene

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

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

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

reordenando términos tenemos

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

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

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

en notación índicial se tiene

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

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

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

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

ordenando los términos tenemos

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

escribiendo la ecuación anterior en notación simbólica, se obtiene

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

por lo tanto, el operador adjunto formal es

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

y el término de valores de frontera es

$$\underline{\mathfrak{D}}(\underline{u},\underline{w})=\underline{u}\nabla\underline{w}-\underline{w}\nabla\underline{u}+\underline{w}p-\underline{u}q. \qquad (4.136)$$

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 [1] y [11].

Si el operador $\mathcal{L}$ está definido por

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

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

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

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

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

aplicando el teorema de Green (82) obtenemos la Ec. (4.50), que podemos re-escribir como

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

Definiendo el operador bilineal

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

y la funcional lineal

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

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

$$a(u,v) = \langle f, v\rangle \tag{4.143}$$

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*

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

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

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

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

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

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

$$a(u,v) = \langle f, v\rangle - a(u_0, v) \tag{4.147}$$

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

Entonces, la cuestión fundamental, es conocer bajo qué condiciones el pro-blema 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 $a(\cdot,\cdot) : V \times V \to \mathbb{R}$ una forma bilineal continua V-elíptico sobre V. Además, sea $l(\cdot) : V \to \mathbb{R}$ una funcional lineal continua sobre V. Entonces

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

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

tiene una y sólo una solución;

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

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

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 $a(\cdot,\cdot) : V\times V \to \mathbb{R}$ una forma bilineal continua V-elíptico sobre V y sea $l(\cdot) : V \to \mathbb{R}$ una funcional lineal continua sobre V. Sea P un subespacio cerrado de V tal que*

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

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

$$Q = \left\{ v \in V \mid \int_\Omega upd\underline{x} = 0 \quad \forall p \in P \right\}, \tag{4.151}$$

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

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

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

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

si y sólo si las condiciones de compatibilidad

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

se satisfacen.

ii) La solución u satisface

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

(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*

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

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

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

se satisface.

5 Métodos de Solución Aproximada para EDP

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). Ya que en general encontrar la solución a problemas con geometría diversa es difícil y en algunos casos imposible usando métodos analíticos.

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

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

aplicando el teorema de Green (82) obtenemos la Ec. (4.50), que podemos re-escribir como

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

Definiendo el operador bilineal

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

y la funcional lineal

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

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

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

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 $\phi_1, \phi_2, ..., \phi_N$ en V y definimos el espacio V^h a partir del subespacio dimensionalmente finito de V generado por las funciones ϕ_i, es decir,

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

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 $N \to \infty$, $h \to 0$.

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

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

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

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

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

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

o más concisamente, como

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

en la cual

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

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

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

o más compactamente

$$\underline{\underline{\mathbb{K}}}\underline{u} = \underline{F} \tag{5.15}$$

en la cual $\underline{\underline{\mathbb{K}}}$ y $\underline{F}$ son la matriz y el vector cuyas entradas son K_{ij} y F_j. 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 simétria son obvias, mientras que la propiedad de V- elíticidad de $a(\cdot,\cdot)$ es por

$$a(v, v) \geq \alpha \|v\|^2 > 0 \qquad \forall v \neq 0, \tag{5.16}$$

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 sean ortogonales entre sí, entonces el sistema (5.12) se simplifica considerablemente, ya que

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

y

$$K_{ii}c_i = F_i \qquad ó \qquad c_i = F_i/K_{ii}. \tag{5.18}$$

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

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

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

$$a(u_h, v_h) = \langle f, v_h \rangle - a(u_0, v_h) \tag{5.20}$$

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

$$\int_\Omega \left(\nabla v_h \cdot \underline{\underline{a}} \cdot \nabla u_h + c u_h v_h \right) dxdy = \int_\Omega f_\Omega v_h dxdy \tag{5.21}$$

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 sigui-ente resultado

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

$$a(u_h, v_h) = \langle l, v_h \rangle \qquad \forall v_h \in V^h. \tag{5.22}$$

Además, si $l(\cdot)$ es de la forma

$$\langle l, v_h \rangle = \int_\Omega f_\Omega v_h d\underline{x} \tag{5.23}$$

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

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

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 [11] y [1].

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

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

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 converge 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

$$\inf_{v_h \in V^h} \|u - v_h\|_V \to 0 \quad \textit{cuando} \quad h \to 0. \tag{5.26}$$

5.2 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

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

entonces

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

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

Sustituyendo esta última expresión Ec. (5.28) en

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

se obtienen N ecuaciones simultáneas

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

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

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

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

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

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

5.3 Método de Elementos Finitos

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 [19]), 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

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

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 $h_i = Diam(\Omega_e)$ de cada Ω_e satisface $Diam(\Omega_e) \leq h$ para cada $e = 1, 2, ..., E$.
- 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

$$V^h = Generado\left\{\phi_i \in \mathbb{P}^h[k] \;\middle|\; \phi_i(x) = 0 \text{ en } \partial\Omega\right\}. \tag{5.31}$$

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

$$\int_\Omega \left(\nabla\phi_i \cdot \underline{\underline{a}} \cdot \nabla\phi_j - c\phi_i\phi_j\right) dxdy = \int_\Omega f_\Omega \phi_j dxdy \tag{5.32}$$

si definimos el operador bilineal

$$K_{ij} \equiv a\left(\phi_i, \phi_j\right) = \int_\Omega \left(\nabla\phi_i \cdot a_{ij} \cdot \nabla\phi_j - c\phi_i\phi_j\right) dxdy \tag{5.33}$$

y la funcional lineal

$$F_j \equiv \left\langle f, \phi_j \right\rangle = \int_\Omega f_\Omega \phi_j dxdy \tag{5.34}$$

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

$$\underline{\underline{K}}\,\underline{u} = \underline{F} \tag{5.35}$$

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

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

donde

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

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

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

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

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

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

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

Entonces, reescribimos la Ec. (5.38) como

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

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

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

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

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

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

$$\underline{\underline{K}}\underline{u} = \underline{F} \tag{5.43}$$

donde

$$K_{ij} = a\left(\phi_i, \phi_j\right) \tag{5.44}$$

y

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

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

6 Método de Elementos Finitos

En este capítulo se considera el VBVP de la forma

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

donde

$$\mathcal{L}u = -\nabla \cdot \underline{\underline{a}} \cdot \nabla u + cu \tag{6.2}$$

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

$$-v\left(\nabla \cdot \underline{\underline{a}} \cdot \nabla u + cu\right) = v f_\Omega \tag{6.3}$$

aplicando el teorema de Green (82) obtenemos la Ec. (4.50), que podemos re-escribir como

$$\int_\Omega \left(\nabla v \cdot \underline{\underline{a}} \cdot \nabla u + cuv\right) d\underline{x} = \int_\Omega v f_\Omega d\underline{x}. \tag{6.4}$$

Definiendo el operador bilineal

$$a(u, v) = \int_\Omega \left(\nabla v \cdot \underline{\underline{a}} \cdot \nabla u + cuv\right) d\underline{x} \tag{6.5}$$

y la funcional lineal

$$l(v) = \langle f, v \rangle = \int_\Omega v f_\Omega d\underline{x} \tag{6.6}$$

podemos reescribir el problema dado por la Ec. (6.1) de orden 2 en forma variacional, haciendo uso de la forma bilineal $a(\cdot, \cdot)$ y la funcional lineal $l(\cdot)$.

6.1 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

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

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 $h_i = Diam(\Omega_e)$ de cada Ω_e satisface $Diam(\Omega_e) \leq h$ para cada $e = 1, 2, ..., E$.
- Los vértices de cada Ω_e son llamados nodos, teniendo N de ellos por cada elemento Ω_e.

Definición 44 *Una familia de triangulaciones $\mathcal{T}_h$ es llamada de forma-regular si existe una constante independiente de h, tal que*

$$h_K \leq C\rho_K, \ \ con\ K \in \mathcal{T}_h,$$

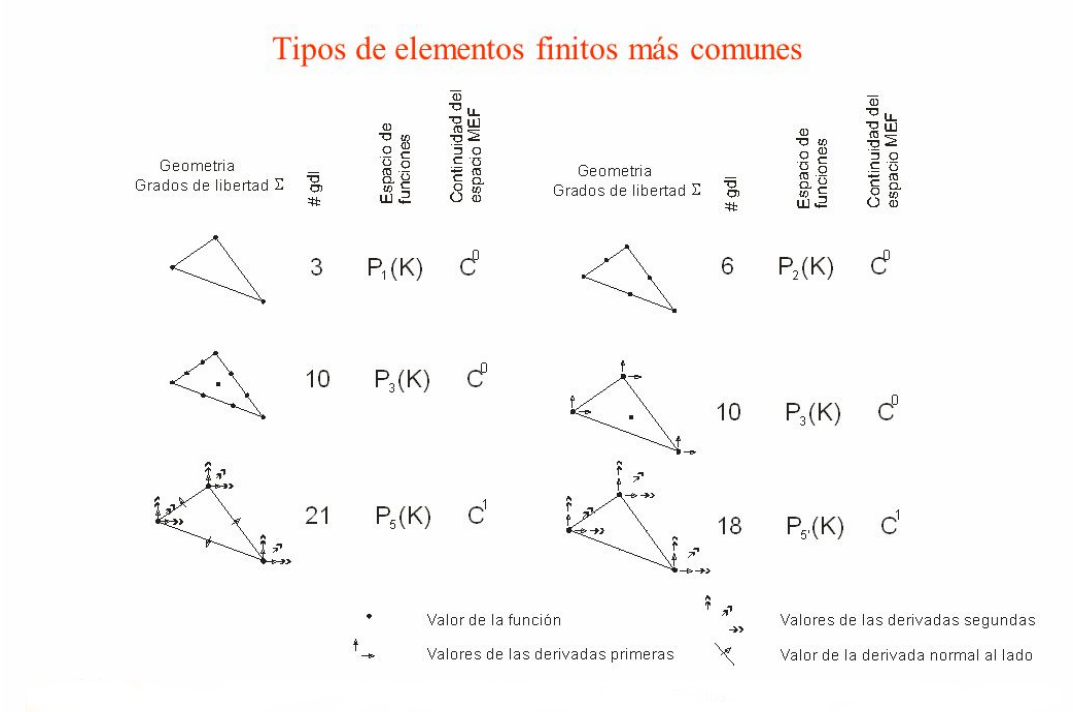
donde ρ_K es el radio del círculo más grande contenido en K. El radio h_K/ρ_K es llamado el aspect ratio de K.

Definición 45 *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*

$$h_K \leq Ch, \ \ con\ K \in \mathcal{T}_h.$$

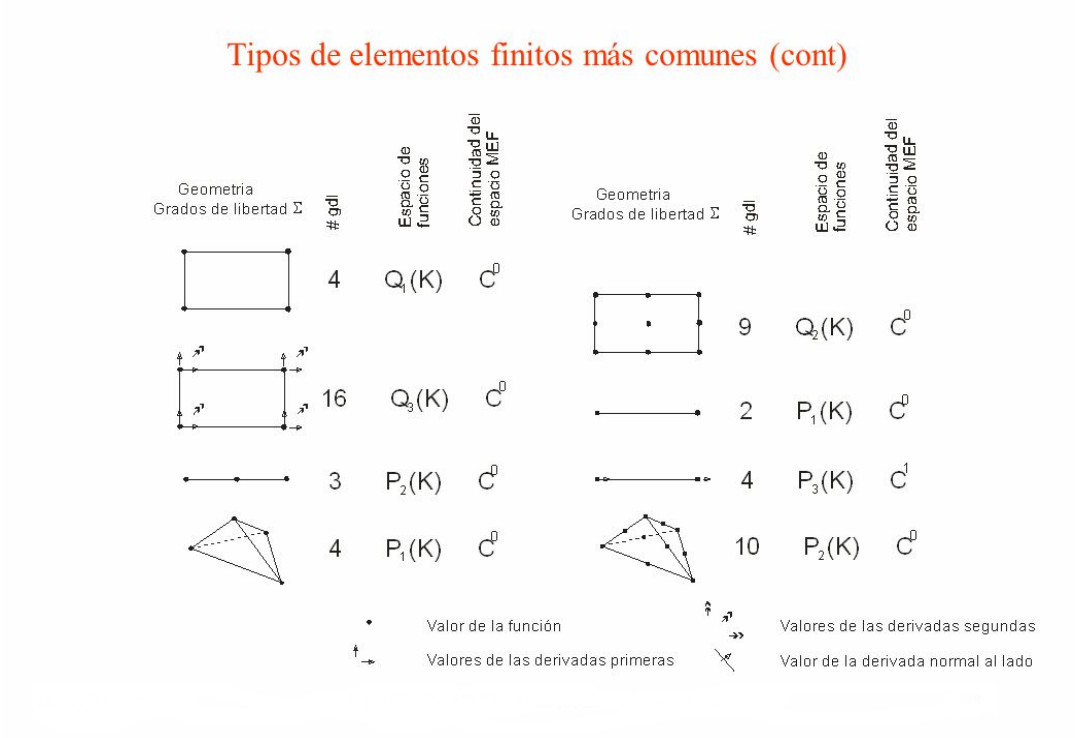
6.2 Generar Mallas con GMSH

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.



Mallado. Se define como mallado la discretización de una línea, superficie o volúmen 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).



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.

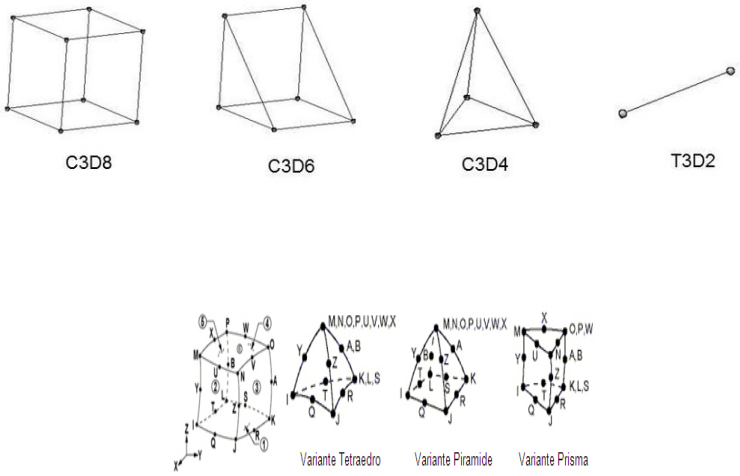


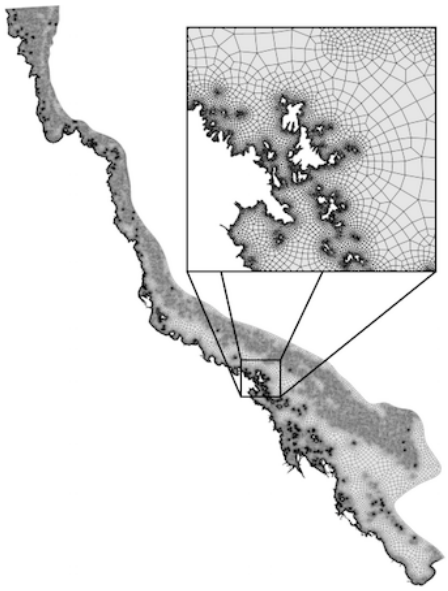
Fig. 3. Geometría del elemento finito sólido estructural

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.

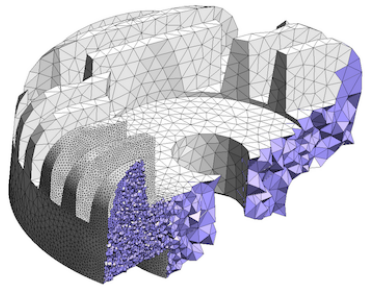
GMSH utiliza un algoritmo de creación de malla basado en la triangulación Delaunay. Dado un contorno se demuestra que sólo existe una triangulación óptima en la que, para cada triángulo formado por tres puntos de la malla, no exista ningún punto dentro de la circunferencia que pasa por los tres puntos.

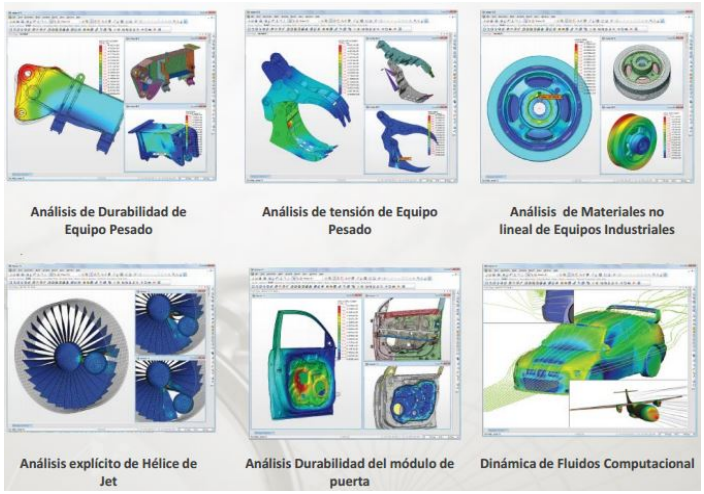
Si bien esta propiedad cierra el problema de la triangulación dados los puntos de la malla, en la mayoría de los casos sólo dispondremos del contorno. GMSH es capaz de encontrar una distribución no estructurada de puntos cuya triangulación resultante tiene suficiente calidad.



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.

6.3 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.

6.4 Discretización en 2D Usando Rectángulos

Para resolver la Ec. (6.1), 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 $\overline{\Omega} = \cup_{e=1}^{E} \overline{\Omega_e}$ y $\overline{\Omega_i} \cap \overline{\Omega_j} \neq \varnothing$ 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 $\underline{\underline{a}} = a \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, entonces se tiene que la integral del lado izquierdo de la Ec. (5.32) queda escrita como

$$\int_\Omega \left(a \nabla \phi_i \cdot \nabla \phi_j + c \phi_i \phi_j \right) dxdy = \int_\Omega f_\Omega \phi_j dxdy \tag{6.7}$$

donde

$$\begin{aligned} K_{ij} &= \int_\Omega \left(a \nabla \phi_i \cdot \nabla \phi_j + c \phi_i \phi_j \right) dxdy \\ &= \sum_{e=1}^{E} \int_{\Omega_e} \left(a \nabla \phi_i^{(e)} \cdot \nabla \phi_j^{(e)} + c \phi_i^{(e)} \phi_j^{(e)} \right) dxdy \\ &= \sum_{e=1}^{E} \int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dxdy \end{aligned} \tag{6.8}$$

y el lado derecho como

$$\begin{aligned} F_j &= \int_\Omega f_\Omega \phi_j dxdy \\ &= \sum_{e=1}^{E} \int_{\Omega_e} f_\Omega \phi_j^{(e)} dxdy. \end{aligned} \tag{6.9}$$

Para cada Ω_e de Ω, la submatriz de integrales (matriz de carga local)

$$K_{ij} = \int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dxdy \tag{6.10}$$

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)

$$F_j = \int_{\Omega_e} f_\Omega \phi_j^{(e)} dxdy \tag{6.11}$$

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 $\underline{\mathbb{F}}$ según la numeración de nodos global, se genera el sistema $\underline{\underline{\mathbb{K}}}\underline{u_h}=\underline{\mathbb{F}}$ donde $\underline{u_h}$ será el vector cuyos valores serán la solución aproximada a la Ec. (6.1) 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.

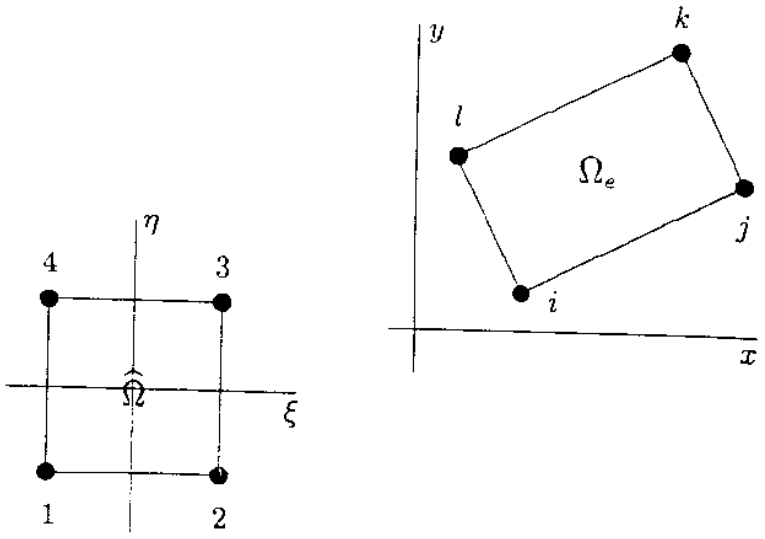
Para implementar numéricamente en cada Ω_e las integrales

$$\int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c\phi_i^{(e)} \phi_j^{(e)} \right) dxdy \tag{6.12}$$

y

$$\int_{\Omega_e} f_\Omega \phi_j^{(e)} dxdy, \tag{6.13}$$

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 $f(x,y) = \underline{\underline{T}}(\varepsilon,\eta)+\underline{b}$, quedando dicha transformación como

$$\begin{aligned} x &= \frac{x_2^{(e)} - x_1^{(e)}}{2}\varepsilon + \frac{y_2^{(e)} - y_1^{(e)}}{2}\eta \\ y &= \frac{x_4^{(e)} - x_1^{(e)}}{2}\varepsilon + \frac{y_4^{(e)} - y_1^{(e)}}{2}\eta \end{aligned} \tag{6.14}$$

en la cual la matriz $\underline{\underline{T}}$ está dada por

$$\underline{\underline{T}} = \begin{pmatrix} \frac{x_2^{(e)}-x_1^{(e)}}{2} & \frac{y_2^{(e)}-y_1^{(e)}}{2} \\ \frac{x_4^{(e)}-x_1^{(e)}}{2} & \frac{y_4^{(e)}-y_1^{(e)}}{2} \end{pmatrix} \tag{6.15}$$

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

$$\begin{aligned}\varepsilon &= \frac{x - b_1 - \frac{y_2^{(e)} - y_1^{(e)}}{2}\left[\frac{y - b_2}{\left(\frac{x_4^{(e)} - x_1^{(e)}}{2}\right)\left(\frac{x - b_1 - \frac{y_2^{(e)} - y_1^{(e)}}{2}}{\frac{x_2^{(e)} - x_1^{(e)}}{2}}\right)}\right]}{\frac{x_2^{(e)} - x_1^{(e)}}{2}} \\ \eta &= \frac{y - b_2}{\left(\frac{x_4^{(e)} - x_1^{(e)}}{2}\right)\left(\frac{x - b_1 - \frac{y_2^{(e)} - y_1^{(e)}}{2}}{\frac{x_2^{(e)} - x_1^{(e)}}{2}}\right) + \frac{y_4^{(e)} - y_1^{(e)}}{2}}.\end{aligned} \tag{6.16}$$

Entonces las $\phi_i^{(e)}$ quedan definidas en términos de $\hat{\phi}_i$ cómo

$$\begin{aligned}\hat{\phi}_1(\varepsilon, \eta) &= \frac{1}{4}(1 - \varepsilon)(1 - \eta) \\ \hat{\phi}_2(\varepsilon, \eta) &= \frac{1}{4}(1 + \varepsilon)(1 - \eta) \\ \hat{\phi}_3(\varepsilon, \eta) &= \frac{1}{4}(1 + \varepsilon)(1 + \eta) \\ \hat{\phi}_4(\varepsilon, \eta) &= \frac{1}{4}(1 - \varepsilon)(1 + \eta)\end{aligned} \tag{6.17}$$

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. (6.14), entonces se tendrían las siguientes integrales

$$\begin{aligned}K_{ij}^{(e)} &= \int_{\Omega_e}\left(a\left[\frac{\partial\phi_i^{(e)}}{\partial x}\frac{\partial\phi_j^{(e)}}{\partial x} + \frac{\partial\phi_i^{(e)}}{\partial y}\frac{\partial\phi_j^{(e)}}{\partial y}\right] + c\phi_i^{(e)}\phi_j^{(e)}\right)dxdy \\ &= \int_{\hat{\Omega}}\left(\left[a\left(\frac{\partial\hat{\phi}_i}{\partial\varepsilon}\frac{\partial\varepsilon}{\partial x} + \frac{\partial\hat{\phi}_i}{\partial\eta}\frac{\partial\eta}{\partial x}\right)\left(\frac{\partial\hat{\phi}_j}{\partial\varepsilon}\frac{\partial\varepsilon}{\partial x} + \frac{\partial\hat{\phi}_j}{\partial\eta}\frac{\partial\eta}{\partial x}\right) + \right.\right. \\ &\quad \left.\left.\left(\frac{\partial\hat{\phi}_i}{\partial\varepsilon}\frac{\partial\varepsilon}{\partial y} + \frac{\partial\hat{\phi}_i}{\partial\eta}\frac{\partial\eta}{\partial y}\right)\left(\frac{\partial\hat{\phi}_j}{\partial\varepsilon}\frac{\partial\varepsilon}{\partial y} + \frac{\partial\hat{\phi}_j}{\partial\eta}\frac{\partial\eta}{\partial y}\right)\right] + c\hat{\phi}_i\hat{\phi}_j\right)|J|\,d\varepsilon d\eta\end{aligned} \tag{6.18}$$

donde el índice i y j varía de 1 a 4. En está 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. (6.15). Para resolver $\int_{\Omega_e} f_\Omega \phi_j^{(e)} dxdy$ en cada Ω_e se genera las integrales

$$\begin{aligned} F_j^{(e)} &= \int_{\Omega_e} f_\Omega \phi_j^{(e)} dxdy \\ &= \int_{\hat{\Omega}} f_\Omega \hat{\phi}_j \, |J| \, d\varepsilon d\eta \end{aligned} \tag{6.19}$$

donde el índice i y j varía de 1 a 4.

Para realizar el cálculo numérico de las integrales en el rectángulo de refe-rencia $\hat{\Omega} = [-1, 1] \times [-1, 1]$, 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. (6.17) obtenemos

$$\begin{array}{ll} \frac{\partial \phi_1}{\partial \varepsilon} = -\frac{1}{4}(1-\eta) & \frac{\partial \phi_1}{\partial \eta} = -\frac{1}{4}(1-\varepsilon) \\ \frac{\partial \phi_2}{\partial \varepsilon} = \frac{1}{4}(1-\eta) & \frac{\partial \phi_2}{\partial \eta} = -\frac{1}{4}(1+\varepsilon) \\ \frac{\partial \phi_3}{\partial \varepsilon} = \frac{1}{4}(1+\eta) & \frac{\partial \phi_3}{\partial \eta} = \frac{1}{4}(1+\varepsilon) \\ \frac{\partial \phi_4}{\partial \varepsilon} = -\frac{1}{4}(1+\eta) & \frac{\partial \phi_4}{\partial \eta} = \frac{1}{4}(1-\varepsilon) \end{array} \tag{6.20}$$

y también

$$\begin{array}{ll} \frac{\partial \varepsilon}{\partial x} = \begin{pmatrix} \frac{y_4^{(e)} - y_1^{(e)}}{2 \det T} \end{pmatrix} & \frac{\partial \varepsilon}{\partial y} = \begin{pmatrix} \frac{x_4^{(e)} - x_1^{(e)}}{2 \det T} \end{pmatrix} \\ \frac{\partial \eta}{\partial x} = \begin{pmatrix} \frac{y_2^{(e)} - y_1^{(e)}}{2 \det T} \end{pmatrix} & \frac{\partial \eta}{\partial y} = \begin{pmatrix} \frac{x_2^{(e)} - x_1^{(e)}}{2 \det T} \end{pmatrix} \end{array} \tag{6.21}$$

las cuales deberán de ser sustituidas en cada $\underline{\underline{K_{ij}^{(e)}}}$ 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 $\underline{\underline{K}}$ y el vector $\underline{F}$ se puede esquematizar como:

$K_{i,j} = (\phi_i, \phi_j) \quad \forall i = 1, 2, ..., E, j = 1, 2, ..., E$
$F_j = (f_\Omega, \phi_j) \quad \forall j = 1, 2, ..., E$
$\forall j = 1, 2, ..., N_d$:

$$b_j = b_j - \gamma_i K_{i,j} \quad \forall i = 1, 2, ..., E$$

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 $\underline{\underline{K}}$ formaremos una matriz $\underline{\underline{A}}$, 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 $\underline{\underline{A}}\underline{x}= \underline{b}$, 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.

6.5 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 triángulares Ω_e tales que $\overline{\Omega} = \cup_{e=1}^{E} \overline{\Omega_e}$ y $\overline{\Omega_i} \cap \overline{\Omega_j} \neq \varnothing$ 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 $\underline{\underline{a}} = a \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, entonces se tiene que la integral del lado izquierdo de la Ec. (5.32) queda escrita como

$$\int_\Omega \left(a \nabla \phi_i \cdot \nabla \phi_j + c \phi_i \phi_j \right) dxdy = \int_\Omega f_\Omega \phi_j dxdy \tag{6.22}$$

donde

$$\begin{aligned} K_{ij} &= \int_\Omega \left(a \nabla \phi_i \cdot \nabla \phi_j + c \phi_i \phi_j \right) dxdy \qquad (6.23) \\ &= \sum_{e=1}^{E} \int_{\Omega_e} \left(a \nabla \phi_i^{(e)} \cdot \nabla \phi_j^{(e)} + c \phi_i^{(e)} \phi_j^{(e)} \right) dxdy \\ &= \sum_{e=1}^{E} \int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dxdy \end{aligned}$$

y el lado derecho como

$$\begin{aligned} \underline{F_j} &= \int_{\Omega} f_{\Omega}\phi_j dxdy \\ &= \sum_{e=1}^{E} \int_{\Omega_e} f_{\Omega}\phi_j^{(e)} dxdy. \end{aligned} \tag{6.24}$$

Para cada Ω_e de Ω la submatriz de integrales (matriz de carga local)

$$\underline{\underline{K_{ij}}} = \int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c\phi_i^{(e)}\phi_j^{(e)} \right) dxdy \tag{6.25}$$

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 carla gocal)

$$F_j = \int_{\Omega_e} f_{\Omega}\phi_j^{(e)} dxdy \tag{6.26}$$

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 $K_{ij}^{(e)}$ en la matriz $\underline{\underline{\mathbb{K}}}$ y los $\underline{F_j^{(e)}}$ en el vector $\underline{\mathbb{F}}$ según la numeración de nodos global, se genera el sistema $\underline{\underline{\mathbb{K}}}\underline{u_h}=\underline{\mathbb{F}}$ 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 $\underline{\underline{\mathbb{K}}}$ 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.

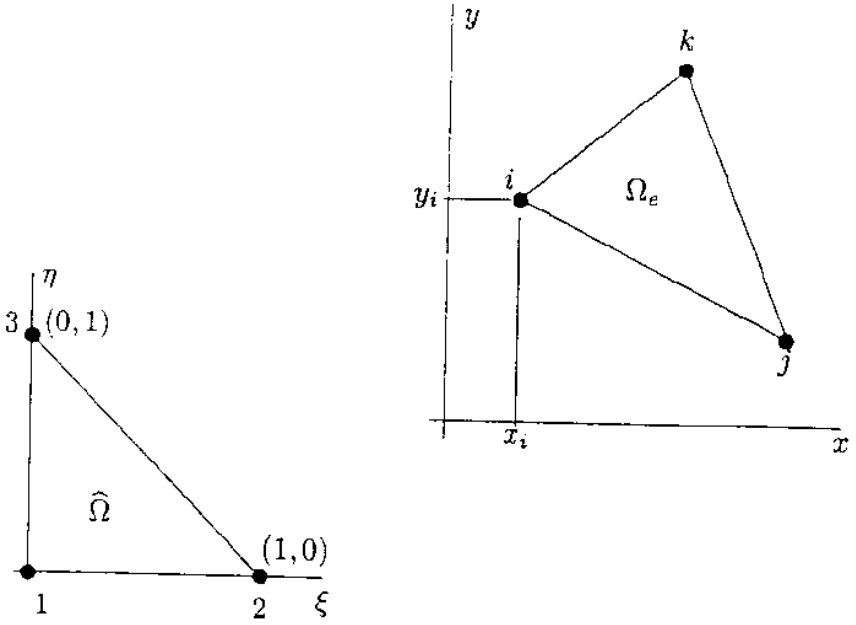
Para implementar numéricamente en cada Ω_e las integrales

$$\int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dxdy \tag{6.27}$$

y

$$\int_{\Omega_e} f_\Omega \phi_j^{(e)} dxdy$$

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 $f(\varepsilon, \eta) = \underline{\underline{T}}(\varepsilon, \eta) + \underline{b}$, quedando dicha transformación como

$$\begin{aligned} x &= x_1^{(e)}(1 - \varepsilon - \eta) + x_2^{(e)}\varepsilon + x_3^{(e)}\eta \\ y &= y_1^{(e)}(1 - \varepsilon - \eta) + y_2^{(e)}\varepsilon + y_3^{(e)}\eta \end{aligned} \tag{6.28}$$

y en la cual la matriz $\underline{\underline{T}}$ está dada por

$$\underline{\underline{T}} = \begin{pmatrix} x_2^{(e)} - x_1^{(e)} & x_3^{(e)} - x_1^{(e)} \\ y_2^{(e)} - y_1^{(e)} & y_3^{(e)} - y_1^{(e)} \end{pmatrix} \tag{6.29}$$

donde $\underline{b}$ es un vector constante

$$\underline{b} = \begin{pmatrix} x_1^{(e)} \\ y_1^{(e)} \end{pmatrix} \tag{6.30}$$

también se tiene que la transformación inversa es

$$\begin{aligned} \varepsilon &= \frac{1}{2A_{\Omega_e}} \left[\left(y_3^{(e)} - y_1^{(e)} \right) \left(x - x_1^{(e)} \right) - \left(x_3^{(e)} - x_1^{(e)} \right) \left(y - y_1^{(e)} \right) \right] \\ \eta &= \frac{1}{2A_{\Omega_e}} \left[- \left(y_2^{(e)} - y_1^{(e)} \right) \left(x - x_1^{(e)} \right) - \left(x_2^{(e)} - x_1^{(e)} \right) \left(y - y_1^{(e)} \right) \right] \end{aligned} \tag{6.31}$$

donde

$$A_{\Omega_e} = \left| \det \begin{bmatrix} 1 & x_1^{(e)} & y_1^{(e)} \\ 1 & x_2^{(e)} & y_2^{(e)} \\ 1 & x_3^{(e)} & y_3^{(e)} \end{bmatrix} \right| . \tag{6.32}$$

Entoces las $\phi_i^{(e)}$ quedan definidas en términos de $\hat{\phi}_i$ cómo

$$\begin{aligned} \hat{\phi}_1(\varepsilon, \eta) &= 1 - \varepsilon - \eta \\ \hat{\phi}_2(\varepsilon, \eta) &= \varepsilon \\ \hat{\phi}_3(\varepsilon, \eta) &= \eta \end{aligned} \tag{6.33}$$

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. (6.28), entonces se tendrían las siguientes integrales

$$\begin{aligned} k_{ij}^{(e)} &= \int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dx dy \\ &= \int_{\hat{\Omega}} \left(\left[a \left(\frac{\partial \hat{\phi}_i}{\partial \varepsilon} \frac{\partial \varepsilon}{\partial x} + \frac{\partial \hat{\phi}_i}{\partial \eta} \frac{\partial \eta}{\partial x} \right) \left(\frac{\partial \hat{\phi}_j}{\partial \varepsilon} \frac{\partial \varepsilon}{\partial x} + \frac{\partial \hat{\phi}_j}{\partial \eta} \frac{\partial \eta}{\partial x} \right) + \right. \right. \\ &\quad \left. \left. \left(\frac{\partial \hat{\phi}_i}{\partial \varepsilon} \frac{\partial \varepsilon}{\partial y} + \frac{\partial \hat{\phi}_i}{\partial \eta} \frac{\partial \eta}{\partial y} \right) \left(\frac{\partial \hat{\phi}_j}{\partial \varepsilon} \frac{\partial \varepsilon}{\partial y} + \frac{\partial \hat{\phi}_j}{\partial \eta} \frac{\partial \eta}{\partial y} \right) \right] + c \hat{\phi}_i \hat{\phi}_j \right) |J| \, d\varepsilon d\eta \end{aligned} \tag{6.34}$$

donde el índice i y j varía de 1 a 3. En está 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. (6.29). Para resolver $\int_{\Omega_e} f_\Omega \phi_j^{(e)} dxdy$ en cada Ω_e se genera las integrales

$$\begin{aligned} F_j^{(e)} &= \int_{\Omega_e} f_\Omega \phi_j^{(e)} dxdy \\ &= \int_{\hat{\Omega}} f_\Omega \hat{\phi}_j \, |J| \, d\varepsilon d\eta \end{aligned} \tag{6.35}$$

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. (6.33) obtenemos

$$\begin{array}{ll} \frac{\partial \phi_1}{\partial \varepsilon} = -1 & \frac{\partial \phi_1}{\partial \eta} = -1 \\ \frac{\partial \phi_2}{\partial \varepsilon} = 1 & \frac{\partial \phi_2}{\partial \eta} = 0 \\ \frac{\partial \phi_3}{\partial \varepsilon} = 0 & \frac{\partial \phi_3}{\partial \eta} = 1 \end{array} \tag{6.36}$$

y también

$$\begin{array}{ll} \frac{\partial \varepsilon}{\partial x} = \frac{\left(y_3^{(e)} - y_1^{(e)}\right)}{2A_{\Omega_e}} & \frac{\partial \varepsilon}{\partial y} = -\frac{\left(x_3^{(e)} - x_1^{(e)}\right)}{2A_{\Omega_e}} \\ \frac{\partial \eta}{\partial x} = -\frac{\left(y_2^{(e)} - y_1^{(e)}\right)}{2A_{\Omega_e}} & \frac{\partial \eta}{\partial y} = \frac{\left(x_2^{(e)} - x_1^{(e)}\right)}{2A_{\Omega_e}} \end{array} \tag{6.37}$$

las cuales deberán de ser sustituidas en cada $\underline{\underline{K_{ij}^{(e)}}}$ 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 $\underline{\underline{K}}$ y el vector $\underline{F}$ se puede esquematizar como:

$K_{i,j} = (\phi_i, \phi_j) \quad \forall i = 1, 2, ..., E, j = 1, 2, ..., E$
$F_j = (f_\Omega, \phi_j) \quad \forall j = 1, 2, ..., E$
$\forall j = 1, 2, ..., N_d$:

$$b_j = b_j - \gamma_i K_{i,j} \quad \forall i = 1, 2, ..., E$$

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 $\underline{\underline{K}}$ formaremos una matriz $\underline{\underline{A}}$, 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 $\underline{\underline{A}}\underline{x}= \underline{b}$, 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.

6.6 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++ u-sando 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 (FEM, DDM secuencial y paralelo) y como es posible optimizar los recursos computacionales con los que se cuente.

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 usó 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 interfases 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.

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:

$$\begin{aligned} -\nabla^2 u &= f_\Omega \text{ en } \Omega \\ u &= g_{\partial\Omega} \text{ en } \partial\Omega. \end{aligned} \tag{6.38}$$

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, entre ellos el flujo de agua subterránea a través de un acuífero isotrópico, homogéneo bajo condiciones de equilibrio y es muy usada 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 consideramos el problema con Ω definido en:

$$\Omega = [-1, 1] \times [0, 1] \tag{6.39}$$

donde

$$f_\Omega = 2n^2\pi^2 \sin(n\pi x) * \sin(n\pi y) \quad \text{y} \quad g_{\partial\Omega} = 0 \tag{6.40}$$

cuya solución es

$$u(x, y) = \sin(n\pi x) * \sin(n\pi y). \tag{6.41}$$

Para las pruebas de rendimiento en las cuales se evalúa el desempeño de los programas realizados se usa $n = 10$, pero es posible hacerlo con $n \in \mathbb{N}$ grande. 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:

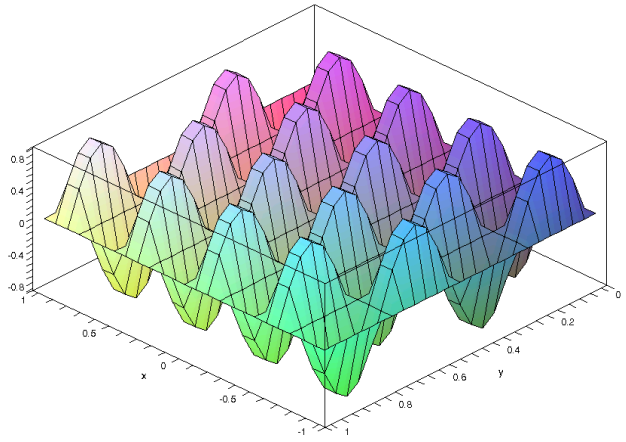


Figura 2: Solución a la ecuación de Poisson para n=4.

Hay que hacer notar que al implementar la solución numérica por el método del elemento finito y el método de subestructuración secuencial en un procesador, un factor limitante para su operación es la cantidad de memoria disponible en la computadora, ya que el sistema algebraico de ecuaciones asociado a esté problema crece muy rápido (del orden de n^2), donde n es el número de nodos en la partición.

En todos los cálculos de los métodos numéricos usados para resolver el sistema lineal algebraico asociado se usó una tolerancia mínima de 1×10^{-10}. Ahora, veremos la implementación del método de elemento finito secuencial para después continuar con el método de descomposición de dominio tanto secuencial como paralelo y poder analizar en cada caso los requerimientos de cómputo, necesarios para correr eficientemente un problema en particular.

Método del Elemento Finito Secuencial A partir de la formulación del método de elemento finito visto en la sección (5.3), la implementación computacional que se desarrolló tiene la jerarquía de clases siguiente:

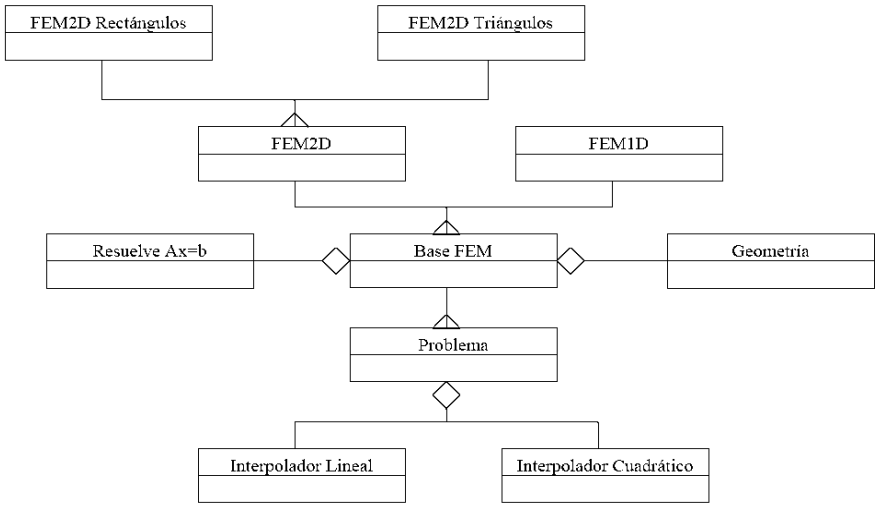


Figura 3: Jerarquía de clases para el método de elemento finito

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 *Geo-metría* 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.

Como ya se mencionó, el método de elemento finito es un algoritmo secuencial, por ello se implementa para que use un solo procesador y un factor limitante para su operación es la cantidad de memoria disponible en la computadora, por ejemplo:

Resolver la Ec. (6.38) con una partición rectangular de 513×513 nodos, genera 262144 elementos rectangulares con 263169 nodos en total, donde 261121 son desconocidos; así el sistema algebraico de ecuaciones asociado a este problema es de dimensión 261121×261121.

Usando el equipo secuencial, primeramente evaluaremos el desempeño del método de elemento finito con los distintos métodos para resolver el sistema algebraico de ecuaciones, encontrando los siguientes resultados:

Método Iterativo	Iteraciones	Tiempo Total
Jacobi	865037	115897 seg.
Gauss-Seidel	446932	63311 seg.
Gradiente Conjugado	761	6388 seg.

Como se observa el uso del método de Gradiente Conjugado es por mucho la mejor elección. En principio, podríamos quedarnos solamente con el método de Gradiente Conjugado sin hacer uso de precondicionadores por los buenos rendimientos encontrados hasta aquí, pero si se desea resolver un problema con un gran número de nodos, es conocido el aumento de eficiencia al hacer uso de precondicionadores.

Ahora, si tomamos ingenuamente el método de elemento finito conjuntamente con el método de Gradiente Conjugado con precondicionadores a posteriori (los más sencillos de construir) para resolver el sistema algebraico de ecuaciones, encontraremos los siguientes resultados:

Precondicionador	Iteraciones	Tiempo Total
Jacobi	760	6388 seg.
SSOR	758	6375 seg.
Factorización Incompleta	745	6373 seg.

Como es notorio el uso del método de Gradiente Conjugado precondicionado con precondicionadores a posteriori no ofrece una ventaja significativa que compense el esfuerzo computacional invertido al crear y usar un precondicionador en los cálculos por el mal condicionamiento del sistema algebraico. Existen también precondicionadores a priori para el método de elemento finito, pero no es costeable en rendimiento su implementación.

6.6.1 Procedimiento General para Programar FEM

Como para casi todo problema no trivial que se desee programar, es necesario tener claro el procedimiento matemático del problema en cuestión. A partir de los modelos matemáticos y los modelos numéricos se plasmará el modelo computacional en algún lenguaje de programación usando algún paradigma de programación soportado por el lenguaje seleccionado.

En particular, para programar el método de Elemento Finito que no es un problema trivial, debemos empezar seleccionado la ecuación diferencial parcial más sencilla, con una malla lo más pequeña posible pero adecuada a la dimensión de nuestro problema, esto nos facilitará hacer los cálculos entendiendo a cabalidad el problema. Además de esta forma podemos encontrar errores lógicos, conceptuales, de cálculo en las integrales, ensamble en las matrices y solución del sistema lineal $Ax = b$ asociado, de otra forma se esta complicando innecesariamente el proceso de programación y depuración. Además tratar de rastrear errores sin tener primero claro los cálculos a los que debe llegar el programa es una complicación innecesaria.

Este procedimiento sirve tanto para *1D, 2D o 3D*, pero lo recomendable es iniciar en *1D* para adquirir la pericia necesaria para las dimensiones mayores, mediante el siguiente procedimiento:

a) Supondremos la ecuación diferencial parcial más sencilla, la ecuación de Poisson, con condiciones de frontera Dirichlet, definido en Ω como:

$$\begin{aligned} -\nabla^2 u &= f_\Omega \text{ en } \Omega \\ u &= g_{\partial\Omega} \text{ en } \partial\Omega. \end{aligned} \tag{6.42}$$

b) Generar la matriz de carga correspondiente y compararla con la calculada por nosotros

c) Si la matriz es correcta, ahora usar una ecuación no homogénea constante para calcular el vector de carga y compararlo con lo calculado por nosotros, de ser necesario ahora podemos cambiar a una función del lado derecho cualquiera

d) En la malla homogénea más sencilla aplicable a la dimensión del problema -para *1D* una malla de 3 nodos, para *2D* una malla de *3x3* nodos (2 elementos por 2 elementos) y para *3D* una malla de *3x3x3* nodos (2 elementos por 2 elementos por 2 elementos)- ahora podemos hacer el ensamble de la matriz global A y el vector b y compararla con la calculada por nosotros

e) Si es correcto el ensambla ahora podemos variar las dimensiones de nuestro dominio y la cantidad de nodos usados en la descretización -en *2D* ahora podemos usar una malla del tipo *2x3* y *3x2*, algo similar en *3D* se puede hacer- para validar el ensamble correcto de la matriz, superado este punto ahora podemos ampliar la malla y ver que el ensamble se mantiene bien hecho -en *2D* la malla mínima adecuada es *3x3* nodos, en *3D* será de *3x3x3* nodos-

f) Regresando a la malla mínima aplicable a la dimensión del problema, ahora solucionar el sistema $Ax = b$ por inversa y revisar que concuerde con la solución calculada por nosotros. Si es correcta, ahora podemos usar algún método del tipo *CGM* o *GMRES* según sea simétrica o no simétrica la matriz A

g) Si contamos con la solución analítica de alguna ecuación, ahora podemos calcular la solución numérica por FEM para distintas mallas y compararla con la analítica y poder ver si la convergencia es adecuada al aumentar la malla

h) llegando a este punto, podemos ahora ampliar los términos de la ecuación diferencial parcial y bastará con revisar que la matriz y vector de carga sea bien generada para tener certeza de que todo funcionará

7 Apéndice A: Algunos Conceptos Básicos

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.

7.1 Nociones de Álgebra Lineal

A continuación detallaremos algunos resultados de álgebra lineal, las demostraciones de los siguientes resultados puede ser consultada en [21].

Definición 46 *Sea V un espacio vectorial y sea $f(\cdot) : V \to \mathbb{R}$, f es llamada funcional lineal si satisface la condición*

$$f(\alpha v + \beta w) = \alpha f(v) + \beta f(w) \quad \forall v, w \in V \quad y \quad \alpha, \beta \in \mathbb{R}. \tag{7.1}$$

Definición 47 *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 48 *Si $\{v_1, ..., v_n\}$ es una base para el espacio vectorial V, entonces existe una única base $\{v_1^*, ..., v_n^*\}$ del espacio vectorial dual V^* llamado la base dual de $\{v_1, ..., v_n\}$ con la propiedad de que $V_i^* = \delta_{ij}$. Por lo tanto V es isomorfo a V^*.*

Definición 49 *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*

$$N(D) = \{f \in V^* \mid f(v) = 0 \quad \forall v \in D\}. \tag{7.2}$$

Teorema 50 *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 51 *Si $V = L \oplus M$ (suma directa) entonces $V^* = N(L) \oplus N(M)$.*

Teorema 52 *Sean V y W espacios lineales, si $T(\cdot) : V \to W$ es lineal, entonces el adjunto T^* de T es un operador lineal $T^* : W^* \to V^*$ definido por*

$$T^*(w^*)(u) = w^*(Tu). \tag{7.3}$$

Teorema 53 *Si H es un espacio completo con producto interior, entonces $H^* = H$.*

Definición 54 *Si V es un espacio vectorial con producto interior y $T(\cdot) : V \to V$ es una transformación lineal, entonces existe una transformación asociada a T llamada la transformación auto-adjunta T^* definida como*

$$\langle Tu, v\rangle = \langle u, T^*v\rangle . \tag{7.4}$$

Definición 55 *Sea V un espacio vectorial sobre los reales. Se dice que una función $\tau(\cdot,\cdot) : V \times V \to \mathbb{R}$ es una forma bilineal sobre V, si para toda $x, y, z \in V$ y $\alpha, \beta \in \mathbb{R}$ se tiene*

$$\begin{aligned} \tau(\alpha x + \beta y, z) &= \alpha\tau(x, z) + \beta\tau(y, z) \\ \tau(x, \alpha y + \beta z) &= \alpha\tau(x, y) + \beta\tau(x, z). \end{aligned} \tag{7.5}$$

Definición 56 *Si $\tau(\cdot,\cdot)$ es una forma bilineal sobre V, entonces la función $q_\tau(\cdot) : V \to \mathbb{R}$ definida por*

$$q_\tau(x) = \tau(x, x) \quad \forall x \in V \tag{7.6}$$

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 57 *Sea $V \subset \mathbb{R}^n$ un subespacio, $P \in \mathbb{R}^n \times \mathbb{R}^n$*

7.2 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 [8] y [1].

Definición 58 *Una σ-álgebra sobre un conjunto Ω es una familia ξ de subconjuntos de Ω que satisface*

- $\varnothing \in \xi$
- *Si* $\psi_n \in \xi$ *entonces* $\bigcup_{n=1}^{\infty} \psi_n \in \xi$
- *Si* $\psi \in \xi$ *entonces* $\psi^c \in \xi$.

Definición 59 *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 60 *Una medida μ sobre Ω es una función no negativa real valuada cuyo dominio es una σ-algebra ξ sobre Ω que satisface*

- $\mu(\varnothing) = 0$ *y*
- *Si* $\{\psi_n\}$ *es una sucesión de conjuntos ajenos de* ξ *entonces*

$$\mu\left(\bigcup_{n=1}^{\infty} \psi_n\right) = \sum_{n=1}^{\infty} \mu(\psi_n). \tag{7.7}$$

Teorema 61 *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 62 *Una función $f : \Omega \to \mathbb{R}$ es llamada medible si $f^{-1}(U)$ es un conjunto medible para todo abierto U de Ω.*

Definición 63 *Sea $E \subset \Omega$ un conjunto, se dice que el conjunto E tiene medida cero si $\mu(E) = 0$.*

Teorema 64 *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 65 *(Fubini)*
Si $f(x,y)$ es medible en $X \times Y$ entonces

$$\int_{X\times Y} f(x,y)\, d\mu = \int_X \int_Y f(x,y)\, d\beta d\alpha = \int_Y \int_X f(x,y)\, d\alpha d\beta \tag{7.8}$$

en el sentido de que cualquiera de las integrales existe y son iguales.

Teorema 66 *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 2 *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 67 *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.*

7.3 Espacios L^p

Las definiciones y material adicional puede ser consultada en [11], [7] y [1].

Definición 68 *Una función medible $f(\cdot)$ (en el sentido de Lebesgue) es llamada integrable sobre un conjunto medible $\Omega \subset \mathbb{R}^n$ si*

$$\int_{\Omega} |f| \, d\underline{x} < \infty. \tag{7.9}$$

Definición 69 *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*

$$\int_{\Omega} |u(\underline{x})|^p \, d\underline{x} \tag{7.10}$$

es integrable.

Al espacio $L^2(\Omega)$ se le llama cuadrado integrable.

Definición 70 *La norma $L^2(\Omega)$ se define como*

$$\|u\|_{L^2(\Omega)} = \left(\int_{\Omega} |u(\underline{x})|^2 \, d\underline{x} \right)^{\frac{1}{2}} < \infty \tag{7.11}$$

y el producto interior en la norma $L^2(\Omega)$ como

$$\langle u, v \rangle_{L^2(\Omega)} = \int_{\Omega} u(\underline{x}) v(\underline{x}) d\underline{x}. \tag{7.12}$$

Definición 71 *Si $p \to \infty$, 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,*

$$L^\infty(\Omega) = \{u \mid |u(x)| \leq k\} \tag{7.13}$$

definida en casi todo Ω, para algún $k \in \mathbb{R}$.

7.4 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 [11], [7] y [1]

Definición 72 *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 73 *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 $\overline{K}$ la clausura de K. Entonces $\overline{K}$ es llamado el soporte de u. Decimos que u tiene soporte compacto sobre Ω si su soporte $\overline{K}$ es compacto. Al conjunto de funciones continuas con soporte compacto se denota por $C_0(\Omega)$.*

Definición 74 *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 $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$. Denotaremos por $|\alpha|$ la suma $|\alpha| = \alpha_1 + \alpha_2 + ... + \alpha_n$ y por $D^\alpha u$ la derivada parcial*

$$D^\alpha u = \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} ... \partial x_n^{\alpha_n}} \tag{7.14}$$

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

Definición 75 *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 76 *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 77 *Una distribución sobre un dominio $\Omega \subset \mathbb{R}^n$ es toda funcional lineal continua sobre $\mathcal{D}(\Omega)$.*

Definición 78 *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 79 *Un función $f(\cdot)$ es llamada localmente integrable, si para todo subconjunto compacto $K \subset \Omega$ se tiene*

$$\int_K |f(x)|\, dx < \infty. \tag{7.15}$$

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 $F : \mathcal{D}(\Omega) \to \mathbb{R}$ como

$$\langle F, \phi \rangle = \int_\Omega f\phi dx \tag{7.16}$$

con $\phi \in \mathcal{D}(\Omega)$.

Si el soporte de ϕ es $K \subset \Omega$, entonces

$$|\langle F, \phi \rangle| = \left| \int_\Omega f\phi dx \right| = \left| \int_K f\phi dx \right| \leq \sup_{x \in K} |\phi| \int_\Omega |f(x)|\, dx \tag{7.17}$$

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 80 *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

$$\langle (uf), \phi \rangle = \langle f, u\phi \rangle \tag{7.18}$$

para toda $\phi \in \mathcal{D}(\Omega)$. Notemos que la anterior ecuación es una generalización de la identidad

$$\int_\Omega [u(x) f(x)] \phi(x) dx = \int_\Omega f(x) [u(x) \phi(x)] dx \tag{7.19}$$

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 81 *La versión clásica del teorema de Green es dada por la identidad*

$$\int_\Omega u \frac{\partial v}{\partial x_i} d\underline{x} = \int_{\partial\Omega} uvn_i d\underline{s} - \int_\Omega v \frac{\partial u}{\partial x_i} d\underline{x} \tag{7.20}$$

que se satisface para todas las funciones u, v en $C^1(\overline{\Omega})$, 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. (7.20) en una dimensión se obtiene usando la fórmula de integración por partes, quedando como

$$\int_a^b uv'dx = [uv]\,|_a^b - \int_a^b vu'dx, \quad u, v \in C^1[a, b] \tag{7.21}$$

como un caso particular de la Ec. (7.20).

Este resultado es fácilmente generalizable a un resultado usando derivadas parciales de orden m de funciones $u, v \in C^m(\overline{\Omega})$ pero reemplazamos u por $D^\alpha u$ en la Ec. (7.20) y con $|\alpha| = m$, entonces se puede mostrar que:

Teorema 82 *Otra versión del teorema de Green es dado por*

$$\int_\Omega (D^\alpha u) v d\underline{x} = (-1)^{|\alpha|} \int_\Omega u D^\alpha v d\underline{x} + \int_{\partial\Omega} h(u, v) d\underline{s} \tag{7.22}$$

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. (7.22) por ϕ perteneciente a $\mathcal{D}(\Omega)$ y como $\phi = 0$ en la frontera $\partial\Omega$ tenemos

$$\int_{\Omega} \left(D^{\alpha}u\right) \phi d\underline{x} = (-1)^{|\alpha|} \int_{\Omega} uD^{\alpha}\phi d\underline{x} \tag{7.23}$$

ya que u es m-veces continuamente diferenciable, esta genera una distribución denotada por u, tal que

$$\langle u, \phi \rangle = \int_{\Omega} u\phi d\underline{x} \tag{7.24}$$

o, como $D^{\alpha}\phi$ también pertenece a $\mathcal{D}(\Omega)$, entonces

$$\langle u, D^{\alpha}\phi \rangle = \int_{\Omega} uD^{\alpha}\phi d\underline{x} \tag{7.25}$$

además, $D^{\alpha}u$ es continua, así que es posible generar una distribución regular denotada por $D^{\alpha}u$ satisfaciendo

$$\langle D^{\alpha}u, \phi \rangle = \int_{\Omega} \left(D^{\alpha}u\right) \phi d\underline{x} \tag{7.26}$$

entonces la Ec. (7.23) puede reescribirse como

$$\langle D^{\alpha}u, \phi \rangle = (-1)^{|\alpha|} \langle u, D^{\alpha}\phi \rangle , \quad \forall \phi \in D(\Omega). \tag{7.27}$$

Definición 83 *La derivada de cualquier distribución $f(\cdot)$ se define como: La $\alpha-$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 $C^{m}\left(\bar{\Omega}\right)$, entonces la derivada parcial distribucional coincide con la derivada parcial $\alpha-$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

$$\langle u, \phi \rangle = \int_{\Omega} u\phi dx \tag{7.28}$$

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

$$\langle D^{\alpha}u, \phi\rangle = (-1)^{|\alpha|}\langle u, D^{\alpha}\phi\rangle, \quad \forall\phi \in D(\Omega). \tag{7.29}$$

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

$$\langle D^{\alpha}u, \phi\rangle = \int_{\Omega} D^{\alpha}u(x)\,\phi(x)\,d\underline{x} \tag{7.30}$$

y se sigue que la función u y $D^{\alpha}u$ están relacionadas por

$$\int_{\Omega} D^{\alpha}u(x)\,\phi(x)\,d\underline{x} = (-1)^{|m|}\int_{\Omega} u(x)\,D^{\alpha}\phi(x)\,d\underline{x} \tag{7.31}$$

para $|\alpha| \leq m$.

Definición 84 *Llamamos a la función (o más precisamente, a la equivalencia de clases de funciones) $D^{\alpha}u$ obtenida en la Ec. (7.31), la $\alpha-$ésima derivada débil de la función u.*

Notemos que si u pertenece a $C^{m}\left(\bar{\Omega}\right)$, entonces la derivada $D^{\alpha}u$ coincide con la derivada clásica para $|\alpha| \leq m$.

8 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 [22]).

El sistema lineal puede ser expresado en la forma matricial $\underline{\underline{A}}\underline{u} = \underline{f}$, donde la matriz $\underline{\underline{A}}$ -que puede ser real o virtual- es de tamaño $n \times n$ con banda b, pero el número total de datos almacenados en ella es a los más $n * b$ números de doble precisión, en el caso de ser simétrica la matriz, el número de datos almacenados es menor a $(n * b)/2$. Además si el problema que la originó es de coeficientes constantes el número de valores almacenados se reduce drásticamente a sólo el tamaño de la banda b.

En el caso de que el método para la resolución del sistema lineal a usar sea del tipo Factorización LU o Cholesky, la estructura de la matriz cambia, ampliándose el tamaño de la banda de b a $2 * b + 1$ en la factorización, en el caso de usar métodos iterativos tipo CGM o GMRES la matriz se mantiene intacta con una banda b.

Para la resolución del sistema lineal virtual asociada a los métodos de descomposición de dominio, la operación básica que se realiza de manera reiterada, es la multiplicación de una matriz por un vector $\underline{v} = \underline{\underline{C}}\underline{u}$, la cual es necesario realizar de la forma más eficiente posible.

Un factor determinante en la implementación computacional, para que esta resulte eficiente, es la forma de almacenar, recuperar y realizar las operaciones que involucren matrices y vectores, de tal forma que la multiplicación se realice en la menor cantidad de operaciones y que los valores necesarios para realizar dichas operaciones queden en la medida de lo posible contiguos para ser almacenados en el Cache[3] del procesador.

[3]Nótese que la velocidad de acceso a la memoria principal (RAM) es relativamente lenta con respecto al Cache, este generalmente está dividido en sub-Caches L1 -de menor tamaño y el más rápido-, L2 y hasta L3 -el más lento y de mayor tamaño- los cuales son de tamaño muy reducido con respecto a la RAM.

Por ello, cada vez que las unidades funcionales de la Unidad de Aritmética y Lógica requieren un conjunto de datos para implementar una determinada operación en los registros, solicitan los datos primeramente a los Caches, estos consumen diversa cantidad de ciclos de reloj para entregar el dato si lo tienen -pero siempre el tiempo es menor que

8.1 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 todos 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 CPU[4].

[4]Por 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 $\underline{\underline{A}}$ 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 $\underline{\underline{A}}$, 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 los datos de la matriz $\underline{\underline{A}}$ 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-

titarea[5] y multiusuario[6] 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 CPU[7].

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;
}
```

[5] Cuentan 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.

[6] Se 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.

[7] Actualmente existen una gran cantidad de distribuciones de GNU/Linux que vienen muy optimizadas intentando conseguir la mejor desenvoltura 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 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.

8.2 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

tiene la siguiente forma

$$\underline{\underline{A}} = \begin{bmatrix} a_1 & b_1 & & & c_1 & & & & \\ d_2 & a_2 & b_2 & & & c_2 & & & \\ & d_3 & a_3 & b_3 & & & c_3 & & \\ & & d_4 & a_4 & b_4 & & & c_4 & \\ e_5 & & & d_5 & a_5 & b_5 & & & c_5 \\ & e_6 & & & d_6 & a_6 & b_6 & & \\ & & e_7 & & & d_7 & a_7 & b_7 & \\ & & & e_8 & & & d_8 & a_8 & b_8 \\ & & & & e_9 & & & d_9 & a_9 \end{bmatrix} \tag{8.1}$$

la cual puede ser almacenada usando el algoritmo (véase [22]) Compressed Diagonal Storage (CDS), optimizado para ser usado en C++, para su almacenamiento y posterior recuperación. Para este ejemplo en particular, se hará uso de un vector de índices

$$\underline{Ind} = [-5, -1, 0, +1, +5] \tag{8.2}$$

y los datos serán almacenados usando la estructura

$$\underline{\underline{Dat}} = \begin{bmatrix} 0 & 0 & a_1 & b_1 & c_1 \\ 0 & d_2 & a_2 & b_2 & c_2 \\ 0 & d_3 & a_3 & b_3 & c_3 \\ 0 & d_4 & a_4 & b_4 & c_4 \\ e_5 & d_5 & a_5 & b_5 & c_5 \\ e_6 & d_6 & a_6 & b_6 & 0 \\ e_7 & d_7 & a_7 & b_7 & 0 \\ e_8 & d_8 & a_8 & b_8 & 0 \\ e_9 & d_9 & a_9 & 0 & 0 \end{bmatrix} \tag{8.3}$$

de tal forma que la matriz $\underline{\underline{A}}$ puede ser reconstruida de forma eficiente. Para obtener el valor $A_{i,j}$, cálculo $ind = j - i$, si el valor ind está en la lista de índices $\underline{Ind}$ -supóngase en la columna k-, entonces $A_{i,j} = Dat_{ik}$, en otro caso $A_{i,j} = 0$.

Casos Particulares de la Matriz Bandada $\underline{\underline{A}}$ Básicamente dos casos particulares surgen en el tratamiento de ecuaciones diferenciales parciales: El primer caso es cuando el operador diferencial parcial es simétrico y el otro, en el que los coeficientes del operador sean constantes.

Para el primer caso, al ser la matriz simétrica, sólo es necesario almacenar la parte con índices mayores o iguales a cero, de tal forma que se buscará el índice que satisfaga $ind = |j - i|$, reduciendo el tamaño de la banda a $b/2$ en la matriz $\underline{\underline{A}}$.

Para el segundo caso, al tener coeficientes constantes el operador diferencial, los valores de los renglones dentro de cada columna de la matriz son iguales, y sólo es necesario almacenarlos una sola vez, reduciendo drásticamente el tamaño de la matriz de datos.

Implementación de Matrices Bandadas Orientada a Objetos en C++ Una forma de implementar la matriz bandada $\underline{\underline{A}}$ de banda b en C++ es mediante:

```
// Creacion
double **Dat;
Dat = new double*[ren];
for (i = 0; i < ren; i++) Dat[i] = new double[b];
int *Ind;
Ind = new int[b];
// Inicializacion
for (i = 0; i < ren; i++)
      for (j = 0; j < b; j++) Dat[i][j] = 0.0;
for (i = 0; i < b; i++) Ind[i] = 0;
```

8.3 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

$$\underline{\underline{A}} = \begin{bmatrix} a_1 & & & & b_1 & & c_1 \\ & a_2 & & b_2 & & c_2 & \\ & & & a_3 & & b_3 & c_3 \\ a_4 & & b_4 & & & & \\ & & a_5 & & b_5 & & c_5 \\ a_6 & b_6 & c_6 & & & & \\ & & & & & a_7 & b_7 & c_7 \\ & & & a_8 & & b_8 & c_8 & \\ & & & & & a_9 & b_9 & \end{bmatrix} \tag{8.4}$$

la cual puede ser almacenada usando el algoritmo (véase [22]) Jagged Diagonal Storage (JDC), optimizado para ser usado en C++. Para este ejemplo en particular, se hará uso de una matriz de índices

$$\underline{\underline{Ind}} = \begin{bmatrix} 1 & 6 & 9 \\ 2 & 5 & 8 \\ 5 & 8 & 9 \\ 1 & 4 & 0 \\ 3 & 6 & 9 \\ 1 & 2 & 3 \\ 7 & 8 & 9 \\ 4 & 7 & 8 \\ 7 & 8 & 0 \end{bmatrix} \tag{8.5}$$

y los datos serán almacenados usando la estructura

$$\underline{\underline{Dat}} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \\ a_4 & b_4 & 0 \\ a_5 & b_5 & c_5 \\ a_6 & b_6 & c_6 \\ a_7 & b_7 & c_7 \\ a_8 & b_8 & c_8 \\ a_9 & b_9 & 0 \end{bmatrix} \tag{8.6}$$

de tal forma que la matriz $\underline{\underline{A}}$ puede ser reconstruida de forma eficiente. Para obtener el obtener el valor $A_{i,j}$, busco el valor j en la lista de índices $\underline{Ind}$ dentro del renglón i, si lo encuentro en la posición k, entonces $A_{i,j} = Dat_{ik}$, en otro caso $A_{i,j} = 0$.

Casos Particulares de la Matriz Dispersa $\underline{\underline{A}}$ Si la matriz $\underline{\underline{A}}$, que al ser almacenada, se observa que existen a lo más r diferentes renglones con valores distintos de los n con que cuenta la matriz y si $r << n$, entonces es posible sólo guardar los r renglones distintos y llevar un arreglo que contenga la referencia al renglón almacenado.

Implementación de Matrices Dispersas Orientada a Objetos en C++ Una forma de implementar la matriz bandada $\underline{\underline{A}}$ de banda b en C++ es mediante:

```
// Creación
double **Dat;
Dat = new double*[ren];
for (i = 0; i < ren; i++) Dat[i] = new double[b];
int **Ind;
Ind = new int*[ren];
for (i = 0; i < ren; i++) Ind[i] = new int[b];
// Inicialización
for (i = 0; i < ren; i++)
     for (j = 0; j < b; j++) Dat[i][j] = 0.0, Ind[i][j] = -1;
```

8.4 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.

9 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

$$\underline{\underline{A}}\,\underline{u} = \underline{f} \tag{9.1}$$

donde la matriz $\underline{\underline{A}}$ 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 rala.

Los métodos de resolución del sistema algebraico de ecuaciones $\underline{\underline{A}}\,\underline{u} = \underline{f}$ se clasifican en dos grandes grupos (véase [19], [20], [21] y [23]):

- En los métodos directos la solución $\underline{u}$ se obtiene en un número fijo de pasos y sólo están sujetos a los errores de redondeo.
- En los métodos iterativos, se realizan iteraciones para aproximarse a la solución $\underline{u}$ aprovechando las características propias de la matriz $\underline{\underline{A}}$, tratando de usar un menor número de pasos que en un método directo.

Por lo general, es conveniente usar bibliotecas[8] para implementar de forma eficiente a los vectores, matrices -bandadas, dispersas o ralas- 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).

[8] Algunas de las bibliotecas más usadas para resolver sistemas lineales usando matrices bandadas y dispersar son PETCs, HYPRE, ATLAS, LAPACK++, LAPACK, EISPACK, LINPACK, BLAS, entre muchas otras alternativas, tanto para implementaciones secuenciales como paralelas y más recientemente para hacer uso de los procesadores CUDA en las GPU de nVidia.

9.1 Métodos Directos

En los métodos directos (véase [19] y [22]), 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 $\underline{\underline{A}}$ 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.

9.1.1 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 $\underline{\underline{A}}$, 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

$$\sum_{j=1}^{N} a_{ij}^{(0)} x_j = a_{i,n+1}^{(0)}, \quad i = 1, 2, ..., N \tag{9.2}$$

y si $a_{11}^{(0)} \neq 0$ y los pivotes $a_{ii}^{(i-1)}, i = 2, 3, ..., N$ 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)

$$x_i + \sum_{j=i+1}^{N} a_{ij}^{(i)} x_j = a_{i,n+1}^{(i)}, \quad i = 1, 2, ..., N \tag{9.3}$$

donde

$$\begin{aligned} k &= 1,2,...,N;\{j=k+1,...,N\{ \\ a_{kj}^{(k)} &= \frac{a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}; \\ i &= k+1,...,N+1\{ \\ a_{ij}^{(k)} &= a_{ij}^{(k-1)} - a_{kj}^{(k)} a_{ik}^{(k-1)}\}\}\} \end{aligned}$$

y las incógnitas se calculan por sustitución hacia atrás, usando las fórmulas

$$\begin{aligned} x_N &= a_{N,N+1}^{(N)}; \\ i &= N-1, N-2, ..., 1 \\ x_i &= a_{i,N+1}^{(i)} - \sum_{j=i+1}^{N} a_{ij}^{(i)} x_j. \end{aligned} \tag{9.4}$$

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 $\underline{\underline{A}}\underline{u} = \underline{b}$ se transformará en $\underline{u} = \underline{\underline{A}}^{-1}\underline{b}$ obteniendo la solución de $\underline{u}$.

9.1.2 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 $\underline{\underline{A}}\underline{u} = \underline{f}$, 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

el método numérico esquemáticamente como:

$$\begin{array}{l} A_{ii} = 1, \text{ para } i = 1, ..., N \\ \text{Para } J = 1, 2, ..., N \ \{ \\ \quad \text{Para } i = 1, 2, ..., j \\ \quad\quad A_{ij} = A_{ij} - \sum_{k=1}^{i-1} A_{ik} A_{kj} \\ \quad \text{Para } i = j+1, j+2, ..., N \\ \quad\quad A_{ij} = \frac{1}{A_{jj}} \left(A_{ij} - \sum_{k=1}^{j-1} A_{ik} A_{kj} \right) \\ \} \end{array} \tag{9.5}$$

El problema original $\underline{\underline{A}}\,\underline{u} = \underline{f}$ se escribe como $\underline{\underline{L}}\,\underline{\underline{U}}\,\underline{u} = \underline{f}$, donde la búsqueda de la solución $\underline{u}$ se reduce a la solución sucesiva de los sistemas lineales triangulares

$$\underline{\underline{L}}\,\underline{y} = \underline{f} \quad \text{y} \quad \underline{\underline{U}}\,\underline{u} = \underline{y}. \tag{9.6}$$

i,.e.

$$\underline{\underline{L}}\,\underline{y} = \underline{f} \Leftrightarrow \left\{ \begin{array}{c} y_1 = f_1 / l_{11} \\ y_i = \frac{1}{A_{ii}} \left(f_i - \sum_{j=1}^{i-1} A_{ii} y_i \right) \text{ para toda } i = 1, ..., n \end{array} \right. \tag{9.7}$$

y

$$\underline{\underline{U}}\,\underline{u} = \underline{y} \Leftrightarrow \left\{ \begin{array}{c} x_n = y_n / u_{nn} \\ x_i = \frac{1}{A_{ii}} \left(y_i - \sum_{j=i+1}^{n} A_{ij} x_j \right) \text{ para toda } i = n-1, ..., 1 \end{array} \right. \tag{9.8}$$

La descomposición $\underline{\underline{L}}\,\underline{\underline{U}}$ requiere $N^3/3$ operaciones aritméticas para la matriz llena, pero sólo Nb^2 operaciones aritméticas para la matriz con un ancho de banda de b siendo esto más económico computacionalmente.

9.1.3 Factorización Cholesky

Cuando la matriz es simétrica y definida positiva, se obtiene la descomposición $\underline{\underline{L}}\,\underline{\underline{U}}$ de la matriz $\underline{\underline{A}} = \underline{\underline{L}}\,\underline{\underline{D}}\,\underline{\underline{U}} = \underline{\underline{L}}\,\underline{\underline{D}}\,\underline{\underline{L}}^T$ donde $\underline{\underline{D}} = diag(\underline{\underline{U}})$ es la diagonal con entradas positivas.

En el algoritmo de Factorización Cholesky, se toma como datos de entrada del sistema $\underline{\underline{A}}\,\underline{u} = \underline{f}$, a la matriz $\underline{\underline{A}}$, la cual será factorizada en la misma matriz y contendrá a las matrices $\underline{\underline{L}}$ y $\underline{\underline{L}}^T$ producto de la factorización, quedando el método numérico esquemáticamente como:

$$\begin{aligned} &\text{para } i = 1, 2, ..., n \text{ y } j = i+1, ..., n \\ &A_{ii} = \sqrt{\left(A_{ii} - \sum_{k=1}^{i-1} A_{ik}^2\right)} \\ &A_{ji} = \left(A_{ji} - \sum_{k=1}^{i-1} A_{jk} A_{ik}\right) / A_{ii} \end{aligned} \tag{9.9}$$

El problema original $\underline{\underline{A}}\,\underline{u} = \underline{f}$ se escribe como $\underline{\underline{L}}\,\underline{\underline{L}}^T\underline{u} = \underline{b}$, donde la búsqueda de la solución $\underline{u}$ se reduce a la solución sucesiva de los sistemas lineales triangulares

$$\underline{\underline{L}}\,\underline{y} = \underline{f} \quad \text{y} \quad \underline{\underline{L}}^T\underline{u} = \underline{y} \tag{9.10}$$

usando la formulación equivalente dada por las Ec.(9.7) y (9.8) para la descomposición LU.

La mayor ventaja de esta descomposición es que, en el caso en que es aplicable, el costo de cómputo es sustancialmente reducido, ya que requiere de $N^3/6$ multiplicaciones y $N^3/6$ sumas.

9.2 Métodos Iterativos

En los métodos iterativos, se realizan iteraciones para aproximarse a la solución $\underline{u}$ aprovechando las características propias de la matriz $\underline{\underline{A}}$, tratando de usar un menor número de pasos que en un método directo (véase [19] y [22]).

En los métodos iterativos tales como Jacobi, Gauss-Seidel y de Relajación Sucesiva (SOR) en el cual se resuelve el sistema lineal

$$\underline{\underline{A}}\,\underline{u} = \underline{f} \tag{9.11}$$

comienza con una aproximación inicial $\underline{u}^0$ a la solución $\underline{u}$ y genera una sucesión de vectores $\left\{u^k\right\}_{k=1}^{\infty}$ que converge a $\underline{u}$. Los métodos iterativos traen consigo un proceso que convierte el sistema $\underline{\underline{A}}\,\underline{u} = \underline{f}$ en otro equivalente mediante la iteración de punto fijo de la forma $\underline{u} = \underline{\underline{T}}\,\underline{u} + \underline{c}$ para alguna matriz

fija $\underline{\underline{T}}$ y un vector $\underline{c}$. Luego de seleccionar el vector inicial $\underline{u}^0$ la sucesión de los vectores de la solución aproximada se genera calculando

$$\underline{u}^k = \underline{\underline{T}}\underline{u}^{k-1} + \underline{c} \quad \forall k = 1, 2, 3, ... \tag{9.12}$$

La convergencia a la solución la garantiza el siguiente teorema (véase [23]).

Teorema 85 *Si* $\left\|\underline{\underline{T}}\right\| < 1$, *entonces el sistema lineal* $\underline{u} = \underline{\underline{T}}\underline{u} + \underline{c}$ *tiene una solución única* $\underline{u}^*$ *y las iteraciones* $\underline{u}^k$ *definidas por la fórmula* $\underline{u}^k = \underline{\underline{T}}\underline{u}^{k-1} + \underline{c} \quad \forall k = 1, 2, 3, ...$ *convergen hacia la solución exacta* $\underline{u}^*$ *para cualquier aproximación inicial* $\underline{u}^0$.

Nótese que, mientras menor sea la norma de la matriz $\underline{\underline{T}}$, más rápida es la convergencia, en el caso cuando $\left\|\underline{\underline{T}}\right\|$ es menor que uno, pero cercano a uno, la convergencia es lenta y el número de iteraciones necesario para disminuir el error depende significativamente del error inicial. En este caso, es deseable proponer al vector inicial $\underline{u}^0$ de forma tal que sea mínimo el error inicial. Sin embargo, la elección de dicho vector no tiene importancia si la $\left\|\underline{\underline{T}}\right\|$ es pequeña, ya que la convergencia es rápida.

Como es conocido, la velocidad de convergencia de los métodos iterativos dependen de las propiedades espectrales de la matriz de coeficientes del sistema de ecuaciones, cuando el operador diferencial $\mathcal{L}$ de la ecuación del problema a resolver es auto-adjunto se obtiene una matriz simétrica y positivo definida y el número de condicionamiento de la matriz $\underline{\underline{A}}$, es por definición

$$cond(\underline{\underline{A}}) = \frac{\lambda_{\max}}{\lambda_{\min}} \geq 1 \tag{9.13}$$

donde $\lambda_{\max}$ y $\lambda_{\min}$ es el máximo y mínimo de los eigen-valores de la matriz $\underline{\underline{A}}$. Si el número de condicionamiento es cercano a 1 los métodos numéricos al solucionar el problema convergerá en pocas iteraciones, en caso contrario se requerirán muchas iteraciones.

Frecuentemente al usar el método de Elemento Finito, Diferencias Finitas, entre otros, se tiene una velocidad de convergencia de $O\left(\frac{1}{h^2}\right)$ y en el caso de métodos de descomposición de dominio sin precondicionar se tiene una velocidad de convergencia de $O\left(\frac{1}{h}\right)$, donde h es la máxima distancia de separación entre nodos continuos de la partición, es decir, que poseen una pobre velocidad de convergencia cuando $h \to 0$ (véase [2], [18], [19] y [23]).

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:

9.2.1 Jacobi

Si todos los elementos de la diagonal principal de la matriz $\underline{\underline{A}}$ son diferentes de cero $a_{ii} \neq 0$ para $i = 1, 2, ...n$. Podemos dividir la $i-$ésima ecuación del sistema lineal (9.11) por a_{ii} para $i = 1, 2, ...n$, y después trasladamos todas las incógnitas, excepto x_i, a la derecha, se obtiene el sistema equivalente

$$\underline{u} = \underline{\underline{B}}\underline{u} + \underline{d} \tag{9.14}$$

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

$$x_i = \sum_{j=1}^{n} b_{ij} x_j^{(k-1)} + d_i \tag{9.15}$$

donde $x_i^{(0)}$ son arbitrarias $(i = 1, 2, ...n; k = 1, 2, ...)$.

También el método de Jacobi se puede expresar en términos de matrices. Supongamos por un momento que la matriz $\underline{\underline{A}}$ tiene la diagonal unitaria, esto es $diag(\underline{\underline{A}}) = \underline{\underline{I}}$. Si descomponemos $\underline{\underline{A}} = \underline{\underline{I}} - \underline{\underline{B}}$, entonces el sistema dado por la Ecs. (9.11) se puede reescribir como

$$\left(\underline{\underline{I}} - \underline{\underline{B}}\right) \underline{u} = \underline{b}. \tag{9.16}$$

Para la primera iteración asumimos que $\underline{k}=\underline{b}$; entonces la última ecuación se escribe como $\underline{u} = \underline{\underline{B}}\underline{u}+\underline{k}$. 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

$$\underline{u}^{m+1} = \underline{\underline{B}}\underline{u}^m + \underline{k}. \tag{9.17}$$

La aplicación del método a la ecuación de la forma $\underline{\underline{A}}\underline{u} = \underline{b}$, con la matriz $\underline{\underline{A}}$ no cero en los elementos diagonales, se obtiene multiplicando la Ec. (9.11) por $D^{-1} = \left[diag(\underline{\underline{A}})\right]^{-1}$ obteniendo

$$\underline{\underline{B}} = \underline{\underline{I}} - \underline{\underline{D}}^{-1}\underline{\underline{A}}, \quad \underline{k} = \underline{\underline{D}}^{-1}\underline{b}. \tag{9.18}$$

9.2.2 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. (9.17) puede ser escrita como

$$u_i^{m+1} = \sum_{j<i} b_{ij} u_j^{m+1} + \sum_{j>i} b_{ij} u_j^m + k_i. \tag{9.19}$$

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

$$\underline{u}^{m+1} = \underline{\underline{E}}\,\underline{u}^{m+1} + \underline{\underline{F}}\,\underline{u}^m + \underline{k} \tag{9.20}$$

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.

9.2.3 Richardson

Escribiendo el método de Jacobi como

$$\underline{u}^{m+1} - \underline{u}^m = \underline{b} - \underline{\underline{A}}\,\underline{u}^m \tag{9.21}$$

entonces el método Richardson se genera al incorporar la estrategia de sobrerrelajación de la forma siguiente

$$\underline{u}^{m+1} = \underline{u}^m + \omega\left(\underline{b} - \underline{\underline{A}}\,\underline{u}^m\right). \tag{9.22}$$

El método de Richardson se define como

$$\underline{u}^{m+1} = \left(\underline{\underline{I}} - \omega\underline{\underline{A}}\right)\underline{u}^m + \omega\underline{b} \tag{9.23}$$

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.

9.2.4 Relajación Sucesiva

Partiendo del método de Gauss-Seidel y sobrerrelajando este esquema, obtenemos

$$u_i^{m+1} = (1-\omega)\, u_i^m + \omega \left[\sum_{j=1}^{i-1} b_{ij} u_j^{m+1} + \sum_{j=i+1}^{N} b_{ij} u_j^m + k_i \right] \tag{9.24}$$

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 [54] y [22]). Estos métodos minimizan, en la k-ésima iteración alguna medida de error sobre el espacio afín $\underline{x_0} + \mathcal{K}_k$, donde $\underline{x_0}$ es la iteración inicial y $\mathcal{K}_k$ es el k-ésimo subespacio de Krylov

$$\mathcal{K}_k = Generado \left\{ \underline{r_0}, \underline{\underline{A}}\underline{r_0}, ..., \underline{\underline{A}}^{k-1}\underline{r_0} \right\} \text{ para } k \geq 1. \tag{9.25}$$

El residual es $\underline{r} = \underline{b} - \underline{\underline{A}}\underline{x}$, tal $\left\{ \underline{r_k} \right\}_{k \geq 0}$ denota la sucesión de residuales

$$\underline{r_k} = \underline{b} - \underline{\underline{A}}\underline{x_k}. \tag{9.26}$$

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 [25].

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 porqué la construcción de una base ortogonal utilizando el procedimiento de Gramm-Schmidt requiere, al seleccionar cada nuevo elemento de la base, asegurar su ortogonalidad con respecto a cada uno de los vectores construidos previamente. La gran ventaja del método de Gradiente Conjugado radica en que cuando se utiliza este procedimiento, basta con asegurar la ortogonalidad de un nuevo miembro con respecto al último que se ha construido, para que automáticamente esta condición se cumpla con respecto a todos los anteriores.

Definición 86 *Una matriz $\underline{\underline{A}}$ es llamada positiva definida si todos sus eigenvalores tienen parte real positiva o equivalentemente, si $\underline{u}^T \underline{\underline{A}} \underline{u}$ tiene parte real positiva para $\underline{u} \in \mathbb{C} \backslash \{0\}$. Notemos en este caso que*

$$\underline{u}^T \underline{\underline{A}} \underline{u} = \underline{u}^T \frac{\underline{\underline{A}} + \underline{\underline{A}}^T}{2} \underline{u} > 0, \ \ con \ \underline{u} \in \mathbb{R}^n \backslash \{0\}.$$

9.2.5 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 $\mathcal{K}_n \left(\underline{\underline{A}}, \underline{v}^n \right)$ y utilizarla para realizar la búsqueda de la solución en forma lo más eficiente posible.

Tal forma de proceder generalmente no sería aconsejable porqué la construcción de una base ortogonal utilizando el procedimiento de 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

$$\underline{\underline{A}} \underline{u} = \underline{f} \tag{9.27}$$

el vector de búsqueda inicial $\underline{u}^0$ y se calcula $\underline{r}^0 = \underline{f} - \underline{\underline{A}}\underline{u}^0$, $\underline{p}^0 = \underline{r}^0$, quedando el método numérico esquemáticamente como:

$$\begin{array}{l} \alpha^n = \frac{\langle \underline{p}^n, \underline{p}^n \rangle}{\langle \underline{p}^n, \underline{\underline{A}}\underline{p}^n \rangle} \\ \underline{u}^{n+1} = \underline{u}^n + \alpha^n \underline{p}^n \\ \underline{r}^{n+1} = \underline{r}^n - \alpha^n \underline{\underline{A}}\underline{p}^n \\ \text{Prueba de convergencia} \\ \beta^n = \frac{\langle \underline{r}^{n+1}, \underline{r}^{n+1} \rangle}{\langle \underline{r}^n, \underline{r}^n \rangle} \\ \underline{p}^{n+1} = \underline{r}^{n+1} + \beta^n \underline{p}^n \\ n = n+1 \end{array} \tag{9.28}$$

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 $\underline{\underline{A}}\underline{p}^n$, una manera de hacerlo queda esquemáticamente como:

Dado el vector de búsqueda inicial $\underline{u}$, calcula $\underline{r} = \underline{f} - \underline{\underline{A}}\underline{u}$, $\underline{p} = \underline{r}$ y $\mu = \underline{r} \cdot \underline{r}$.

Para $n = 1, 2, ...,$Mientras $(\mu < \varepsilon)$ {
- $\underline{v} = \underline{\underline{A}}\underline{p}$
- $\alpha = \frac{\mu}{\underline{p} \cdot \underline{v}}$
- $\underline{u} = \underline{u} + \alpha \underline{p}$
- $\underline{r} = \underline{r} - \alpha \underline{v}$
- $\mu' = \underline{r} \cdot \underline{r}$
- $\beta = \frac{\mu'}{\mu}$
- $\underline{p} = \underline{r} + \beta \underline{p}$
- $\mu = \mu'$

}

La solución aproximada será $\underline{u}$ y el vector residual será $\underline{r}$.

Si se denota con $\{\lambda_i, V_i\}_{i=1}^N$ las eigen-soluciones de $\underline{\underline{A}}$, i.e. $\underline{\underline{A}}V_i = \lambda_i V_i$, $i = 0, 1, 2, ..., N$. Ya que la matriz $\underline{\underline{A}}$ es simétrica, los eigen-valores son reales y se pueden ordenar $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_N$. Se define el número de condición por $Cond(\underline{\underline{A}}) = \lambda_N / \lambda_1$ y la norma de la energía asociada a $\underline{\underline{A}}$ por $\|\underline{u}\|^2_{\underline{\underline{A}}} = \underline{u} \cdot \underline{\underline{A}}\underline{u}$ entonces

$$\left\| \underline{u} - \underline{u}^k \right\|_{\underline{\underline{A}}} \leq \left\| \underline{u} - \underline{u}^0 \right\|_{\underline{\underline{A}}} \left[\frac{1 - \sqrt{Cond(\underline{\underline{A}})}}{1 + \sqrt{Cond(\underline{\underline{A}})}} \right]^{2k}. \tag{9.29}$$

El siguiente teorema da idea del espectro de convergencia del sistema $\underline{\underline{A}}\underline{u} = \underline{b}$ para el método de Gradiente Conjugado.

Teorema 87 *Sea* $\kappa = cond(\underline{\underline{A}}) = \frac{\lambda_{\max}}{\lambda_{\min}} \geq 1$, *entonces el método de Gradiente Conjugado satisface la* $\underline{\underline{A}}-$*norma del error dado por*

$$\frac{\|e^n\|}{\|e^0\|} \leq \frac{2}{\left[\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^n + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{-n}\right]} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n \tag{9.30}$$

donde $\underline{e}^m = \underline{u} - \underline{u}^m$ *del sistema* $\underline{\underline{A}}\underline{u} = \underline{b}$.

Nótese que para κ grande se tiene que

$$\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \simeq 1 - \frac{2}{\sqrt{\kappa}} \tag{9.31}$$

tal que

$$\|\underline{e}^n\|_{\underline{\underline{A}}} \simeq \left\|\underline{e}^0\right\|_{\underline{\underline{A}}} \exp\left(-2\frac{n}{\sqrt{\kappa}}\right) \tag{9.32}$$

de lo anterior se puede esperar un espectro de convergencia del orden de $O(\sqrt{\kappa})$ iteraciones (véase [23] y [54]).

9.2.6 Gradiente Conjugado Precondicionado

Cuando la matriz $\underline{\underline{A}}$ es simétrica y definida positiva se puede escribir como

$$\lambda_1 \leq \frac{\underline{u}\underline{\underline{A}} \cdot \underline{u}}{\underline{u} \cdot \underline{u}} \leq \lambda_n \tag{9.33}$$

y tomando la matriz $\underline{\underline{C}}^{-1}$ como un precondicionador de $\underline{\underline{A}}$ con la condición de que

$$\lambda_1 \leq \frac{\underline{u}\underline{\underline{C}}^{-1}\underline{\underline{A}} \cdot \underline{u}}{\underline{u} \cdot \underline{u}} \leq \lambda_n \tag{9.34}$$

entonces la Ec. (9.27) se puede escribir como

$$\underline{\underline{C}}^{-1}\underline{\underline{A}}\underline{u} = \underline{\underline{C}}^{-1}b \tag{9.35}$$

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

$$\begin{aligned} \left\langle \underline{u}, \underline{\underline{C}}^{-1}\underline{\underline{A}}\underline{v} \right\rangle &= \underline{u} \cdot \underline{\underline{C}} \left(\underline{\underline{C}}^{-1}\underline{\underline{A}}\underline{v} \right) \\ &= \underline{u} \cdot \underline{\underline{A}}\underline{v} \end{aligned} \tag{9.36}$$

que por hipótesis es simétrica y definida positiva en ese producto interior.

La elección del producto interior $\langle \cdot, \cdot \rangle$ quedará definido como

$$\langle \underline{u}, \underline{v} \rangle = \underline{u} \cdot \underline{\underline{C}}^{-1}\underline{\underline{A}}\underline{v} \tag{9.37}$$

por ello las Ecs. (9.28[1]) y (9.28[3]), se convierten en

$$\alpha^{k+1} = \frac{\underline{r}^k \cdot \underline{r}^k}{\underline{p}^{k+1} \cdot \underline{\underline{C}}^{-1}\underline{p}^{k+1}} \tag{9.38}$$

y

$$\beta^{k+1} = \frac{\underline{p}^k \cdot \underline{\underline{C}}^{-1}\underline{r}^k}{\underline{p}^k \cdot \underline{\underline{A}}\underline{p}^k} \tag{9.39}$$

generando el método de Gradiente Conjugado precondicionado con precondicionador $\underline{\underline{C}}^{-1}$. 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 precondicionador $\underline{\underline{C}}^{-1}$. 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:

$$\begin{aligned} \beta^{k+1} &= \frac{\underline{p}^k \cdot \underline{\underline{C}}^{-1}\underline{r}^k}{\underline{p}^k \cdot \underline{\underline{A}}\underline{p}^k} \\ \underline{p}^{k+1} &= \underline{r}^k - \beta^{k+1}\underline{p}^k \\ \alpha^{k+1} &= \frac{\underline{r}^k \cdot \underline{r}^k}{\underline{p}^{k+1} \cdot \underline{\underline{C}}^{-1}\underline{p}^{k+1}} \\ \underline{u}^{k+1} &= \underline{u}^k + \alpha^{k+1}\underline{p}^{k+1} \\ \underline{r}^{k+1} &= \underline{\underline{C}}^{-1}\underline{r}^k - \alpha^{k+1}\underline{\underline{A}}\underline{p}^{k+1}. \end{aligned} \tag{9.40}$$

Algoritmo Computacional del Método Dado el sistema $\underline{\underline{A}}\underline{u} = \underline{b}$, con la matriz $\underline{\underline{A}}$ 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 precondicionamiento $\underline{\underline{C}}^{-1}$ de dimensión $n \times n$, el algoritmo del método de Gradiente Conjugado Precondicionado queda como:

$\underline{r} = \underline{b} - \underline{\underline{A}}\underline{u}$
$\underline{w} = \underline{\underline{C}}^{-1}\underline{r}$
$\underline{v} = (\underline{\underline{C}}^{-1})^T\underline{w}$
$\alpha = \sum_{j=1}^{n} w_j^2$
$k = 1$
Mientras que $k \leq N$

Si $\|\underline{v}\|_\infty < \varepsilon$ Salir
$\underline{x} = \underline{\underline{A}}\underline{v}$
$t = \dfrac{\alpha}{\sum_{j=1}^{n} v_j x_j}$
$\underline{u} = \underline{u} + t\underline{v}$
$\underline{r} = \underline{r} - t\underline{x}$
$\underline{w} = \underline{\underline{C}}^{-1}\underline{r}$
$\beta = \sum_{j=1}^{n} w_j^2$
Si $\|\underline{r}\|_\infty < \varepsilon$ Salir
$s = \dfrac{\beta}{\alpha}$
$\underline{v} = \left(\underline{\underline{C}}^{-1}\right)^T \underline{w} + s\underline{v}$
$\alpha = \beta$
$k = k + 1$

La salida del método será la solución aproximada $\underline{u} = (u_1, ..., u_n)$ y el residual $\underline{r} = (r_1, ..., r_n)$.

En el caso del método sin precondicionamiento, $\underline{\underline{C}}^{-1}$ 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 $\underline{\underline{A}}$ 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 $\underline{\underline{A}}\underline{u} = \underline{b}$ generada por la discretización de un problema elíptico y cómo extender estos para matrices no simétricas ver [25] y [26].

Teorema 88 *Sean $\underline{\underline{A}}, \underline{\underline{B}}$ y $\underline{\underline{C}}$ tres matrices simétricas y positivas definidas entonces*

$$\kappa\left(\underline{\underline{C}}^{-1}\underline{\underline{A}}\right) \leq \kappa\left(\underline{\underline{C}}^{-1}\underline{\underline{B}}\right)\kappa\left(\underline{\underline{B}}^{-1}\underline{\underline{A}}\right).$$

9.2.7 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

$$\left\{\underline{v}^1, \underline{v}^2, ..., \underline{v}^n\right\} \tag{9.41}$$

para el espacio de Krylov $\mathcal{K}_n\left(\underline{\underline{A}}, \underline{v}^n\right)$. Para hacer $\underline{v}^{n+1}$ ortogonal a $\mathcal{K}_n\left(\underline{\underline{A}}, \underline{v}^n\right)$, es necesario usar todos los vectores previamente construidos $\{\underline{v}^{n+1j}\}_{j=1}^{n}$ -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 $\underline{\underline{V}}_n = [\underline{v}^1, \underline{v}^2, ..., \underline{v}^n]$ la cual denota la matriz conteniendo $\underline{v}^j$ en la j-ésima columna, para $j = 1, 2, ..., n$, y sea $\underline{\underline{H}}_n = [h_{i,j}], 1 \leq i, j \leq n$, donde las entradas de $\underline{\underline{H}}_n$ no especificadas en el algoritmo son cero. Entonces, $\underline{\underline{H}}_n$ es una matriz superior de Hessenberg. i.e. $h_{ij} = 0$ para $j < i - 1$, y

$$\begin{aligned} \underline{\underline{AV}}_n &= \underline{\underline{V}}_n\underline{\underline{H}}_n + h_{n+1,n}\left[0, ..., 0, \underline{v}^{n+1}\right] \\ \underline{\underline{H}}_n &= \underline{\underline{H}}_n^T\underline{\underline{AV}}_n. \end{aligned} \tag{9.42}$$

En el algoritmo del método Residual Mínimo Generalizado, la matriz $\underline{\underline{A}}$ es tomada como no simétrica, y como datos de entrada del sistema

$$\underline{\underline{A}}\underline{u} = \underline{f} \tag{9.43}$$

el vector de búsqueda inicial $\underline{u}^0$ y se calcula $\underline{r}^0 = \underline{f} - \underline{\underline{A}}\underline{u}^0$, $\beta^0 = \|r^0\|$, $\underline{v}^1 = \underline{r}^0/\beta^0$, quedando el método esquemáticamente como:

$$\begin{array}{l}
\text{Para } n = 1, 2, ..., \text{Mientras } \beta^n < \tau\beta^0 \ \{ \\
\quad \underline{w}_0^{n+1} = \underline{\underline{A}}\underline{v}^n \\
\quad \text{Para } l = 1 \text{ hasta } n \ \{ \\
\qquad h_{l,n} = \left\langle \underline{w}_l^{n+1}, \underline{v}^l \right\rangle \\
\qquad \underline{w}_{l+1}^{n+1} = \underline{w}_l^{n+1} - h_{l,n}\underline{v}^l \\
\quad \} \\
\quad h_{n+1,n} = \left\| \underline{w}_{n+1}^{n+1} \right\| \\
\quad \underline{v}^{n+1} = \underline{w}_{n+1}^{n+1}/h_{n+1,n} \\
\quad \text{Calcular } \underline{y}^n \text{ tal que } \beta^n = \left\| \beta^0 \underline{e}_1 - \underline{\underline{\hat{H}}}_n \underline{y}^n \right\| \text{ es mínima} \\
\}
\end{array} \tag{9.44}$$

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

$$\underline{r}^n = \underline{r}^0 - \underline{\underline{AV}}_n y^n = \underline{\underline{V}}_{n+1} \left(\beta^0 \underline{e}_1 - \underline{\underline{\hat{H}}}_n \underline{y}^n \right). \tag{9.45}$$

Teorema 89 *Sea $\underline{u}^k$ la iteración generada después de k iteraciones de GMRES, con residual $\underline{r}^k$. Si la matriz $\underline{\underline{A}}$ es diagonalizable, i.e. $\underline{\underline{A}} = \underline{\underline{V}}\Lambda\underline{\underline{V}}^{-1}$ donde Λ es una matriz diagonal de eigen-valores de $\underline{\underline{A}}$, y $\underline{\underline{V}}$ es la matriz cuyas columnas son los eigen-vectores, entonces*

$$\frac{\left\| r^k \right\|}{\left\| r^0 \right\|} \le \kappa(V) \min_{p_\kappa \in \Pi\kappa, p_k(0)=1} \max_{\lambda_j} |p_\kappa(\lambda_j)| \tag{9.46}$$

donde $\kappa(V) = \frac{\|\underline{\underline{V}}\|}{\|\underline{\underline{V}}^{-1}\|}$ es el número de condicionamiento de $\underline{\underline{V}}$.

9.3 Algunos Ejemplos

Sea el sistema lineal $\underline{\underline{A}}\underline{u} = \underline{f}$ 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 $\underline{\underline{A}}$, tratando de usar un menor número de pasos que en un método directo.

Por lo general, es conveniente usar bibliotecas[9] para implementar de forma eficiente a los vectores, matrices -bandadas, dispersas o ralas- y resolver el sistemas lineal.

[9] Algunas de las bibliotecas más usadas para resolver sistemas lineales usando matrices bandadas y dispersar son PETCs, HYPRE, ATLAS, LAPACK++, LAPACK, EISPACK, LINPACK, BLAS, entre muchas otras alternativas, tanto para implementaciones secuenciales como paralelas y más recientemente para hacer uso de los procesadores CUDA en las GPU de nVidia.

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.

9.3.1 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[10].

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)
```

[10]En 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)
```

9.3.2 Usando C++

GMM++[11] 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;
```

[11] GMM++ [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;
}
```

9.4 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 (Peripherial 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 el 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 lineales[12] y no lineales.

9.5 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 $\underline{\underline{A}}\underline{u} = \underline{f}$ donde la matriz $\underline{\underline{A}}$ es de tamaño $n \times n$ y generalmente bandada, cuyo tamaño de banda es b.

Los métodos de resolución del sistema algebraico de ecuaciones $\underline{\underline{A}}\underline{u} = \underline{f}$ se clasifican en dos grandes grupos (véase [23]): los métodos directos y los métodos iterativos. En los métodos directos la solución $\underline{u}$ se obtiene en un número fijo de pasos y sólo estan sujetos a los errores de redondeo. En los métodos iterativos, se realizan iteraciones para aproximarse a la solución $\underline{u}$ aprovechando las características propias de la matriz $\underline{\underline{A}}$, tratando de usar un menor número de pasos que en un método directo (véase [19], [20], [21] y [23]).

Por lo general, es conveniente usar librerías[13] para implementar de forma

[12] Algunas de las bibliotecas más usadas para resolver sistemas lineales usando matrices bandadas y dispersar son PETCs, HYPRE, ATLAS, LAPACK++, LAPACK, EISPACK, LINPACK, BLAS, entre muchas otras alternativas, tanto para implementaciones secuenciales como paralelas y más recientemente para hacer uso de los procesadores CUDA en las GPU de nVidia.

[13] Algunas de las librerías más usadas para resolver sistemas lineales usando matrices bandadas y dispersar son PETCs, HYPRE, ATLAS, LAPACK++, LAPACK, EISPACK, LINPACK, BLAS, entre muchas otras alternativas, tanto para implementaciones secuenciales como paralelas y más recientemente para hacer uso de los procesadores CUDA en las GPU de nVidia.

eficiente a los vectores, matrices -bandadas y dispersas- y resolver los sistemas lineales locales asociados al método DVS, pero se decidió[14] 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:

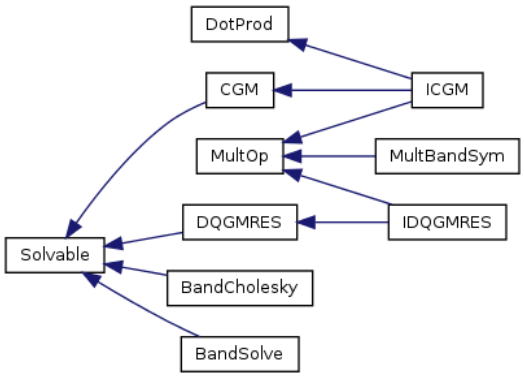


Figura 4: Jerarquía de clases para la resolución de sistemas lineales

Esta jerarquía permite que los métodos DVS le soliciten a la clase abstracta Solvable[15] que resuelva un determinado sistema lineal sin importar el

[14] La variedad de librerías y las diferentes versiones que existen -muchas de ellas, difieren en la forma de programar entre versiones sucesivas- y pueden usarse es grande, pero cada una de ellas requiere de una implementación especifica en el programa y una configuración particular del equipo. Esto restringe al código desarrollado a una plataforma y versión particular de la librería seleccionada.

[15] En general los comportamientos virtuales de las clases abstractas, pueden no ser eficientes si son llamadas una gran cantidad de veces durante la ejecución del programa. Para el caso del esquema DVS, en el cual se usa CGM o GMRES para resolver el sistema

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.

10 Apéndice D: 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 ó $1e + 45$.

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 flotante[16].

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 subdesbordamiento.

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

[16] 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 flotante[17].

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.

[17] ¿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.
- 3.14159265358979323846264338327950288 42 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 norma-lizada, 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 $\pm d.dd...d \times \beta^e$, donde $d.dd...d$ se llama mantisa y tiene p dígitos. De forma más precisa $\pm d_0.d_1d_2...d_{p-1} \times \beta^e$ representa el número

$$\pm (d_0 + d_1\beta^{-1} + ... + d_{p-1}\beta^{-(p-1)})\beta^e, (0 \leq d_i < \beta). \tag{10.1}$$

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. (10.1)), 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 decimales[18].

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.

[18] El 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
```

Ya que el resto es más que la mitad de 10, la mejor aproximación se obtiene redondeándolo

```
>>> q+1
7205759403792794
```

Por lo tanto la mejor aproximación a 1/10 en doble precisión 754 es

$$7205759403792794/2 * *56$$

el dividir tanto el numerador como el denominador reduce la fracción a

$$3602879701896397/2 * *55$$

Notemos que como lo redondeamos, esto es un poquito más grande que 1/10; si no lo hubiéramos redondeado, el cociente hubiese sido un poquito menor que 1/10. ¡Pero no hay caso en que sea exactamente 1/10!

Entonces la computadora nunca «ve» 1/10: lo que ve es la fracción exacta de arriba, la mejor aproximación al flotante doble de 754 que puede obtener

```
>>> 0.1 * 2 ** 55
3602879701896397.0
```

Si multiplicamos esa fracción por 10^{55}, podemos ver el valor hasta los 55 dígitos decimales

```
>>> 3602879701896397 * 10 ** 55 // 2 ** 55
1000000000000000055511151231257827021181583404541015625
```

lo que significa que el valor exacto almacenado en la computadora es igual al valor decimal

$$0.1000000000000000055511151231257827021181583404541015625.$$

en lugar de mostrar el valor decimal completo, muchos lenguajes, redondean el resultado a 17 dígitos significativos.

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 Ulps 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 $|d.d...d - (z/\beta^e)|^{\beta^{p-1}}$ 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 $\overbrace{d.dd...dd}^{p} \times \beta^e$, el error puede ser tan grande como $\overbrace{0.00...00}^{p}\beta' \times \beta^e$, 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 $d.dd...dd \times \beta^e$ 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,

$$\frac{1}{2}\beta^{-p} \leq \frac{1}{2}ulp \leq \frac{\beta}{2}\beta^{-p} \tag{10.2}$$

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. (10.2) 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 $\widetilde{x} = 1.24 \times 10^1$. El error es 0.5 ulps, el error relativo es 0.8ϵ. A continuación, considere el cálculo $8\widetilde{x}$. El valor exacto es $8x = 98.8$, mientras que el valor calculado es $8\widetilde{x} = 9.92 \times 10^1$. 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. (10.2) 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

$$\text{los dígitos contaminados} \approx \log_\beta n. \tag{10.3}$$

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

$x = 2.15 \times 10^{12}$
$y = 0.000000000000000125 \times 10^{12}$
$x - y = 2.149999999999999875 \times 10^{12}$

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

$x = 2.15 \times 10^{12}$
$y = 0.00 \times 10^{12}$
$x - y = 2.15 \times 10^{12}$

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

$x = 1.01 \times 10^1$

$y = 0 - 99 \times 10^1$

$x - y = 0.02 \times 10^1$

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 90 *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. (10.3) 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

$x = 1.010 \times 10^1$

$y = 0.993 \times 10^1$

$x - y = 0.017 \times 10^1$

y la respuesta es exacta. Con un solo dígito de guarda, el error relativo del resultado puede ser mayor que ϵ, como en $110 - 8.59$.

$x = 1.10 \times 10^2$

$y = 0.085 \times 10^2$

$x - y = 1.015 \times 10^2$

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 91 *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 (90)). 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 (91), 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

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \text{ y } x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \tag{10.4}$$

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

$$x_1 = \frac{-2c}{b+\sqrt{b^2-4ac}} \text{ y } x_2 = \frac{-2c}{b-\sqrt{b^2-4ac}} \tag{10.5}$$

Si $b^2 \gg ac$ y $b > 0$, entonces calcular x_1 usando la Ec. (10.4) implicará una cancelación. Por lo tanto, use la Ec. (10.5) para calcular x_1 y (10.4) para x_2. Por otro lado, si $b < 0$, use (10.4) para calcular x_1 y (10.5) 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 (91), 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.

10.1 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. (10.4) o de forma alternativa con el algoritmo que se obtiene mediante la racionalización del numerador Ec. (10.5).

Otro algoritmo que podemos implementar es el método de Newton-Raphson[19] 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)
```

[19] También 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")
    else:
      # Raices complejas
      print("Raices Complejas ...")
if __name__ == '__main__':
    raices(1.0, 4.0, 1.0)
```

y generan la siguiente salida:

```
Polinomio (1.000000) X^2 + (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.

10.2 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

números de punto flotante, es decir, tipos de datos: float (32 bits), double (64 bits) o long double (80 bits).

Problemas de Precisión Como ya vimos en las secciones precedentes, los números de punto flotante se representan en el Hardware de la computadora en fracciones en base 2 (binario). Por ejemplo, la fracción decimal

$$0.125$$

tiene el valor $1/10+2/100+5/1000$, y de la misma manera la fracción binaria

$$0.001$$

tiene el valor $0/2 + 0/4 + 1/8$. Estas dos fracciones tienen valores idénticos, la única diferencia real es que la primera está escrita en notación fraccional en base 10 y la segunda en base 2.

Desafortunadamente, la mayoría de las fracciones decimales no pueden representarse exactamente como fracciones binarias. Como consecuencia, en general los números de punto flotante decimal que usamos en la computadora son sólo aproximados por los números de punto flotante binario que realmente se guardan en la máquina.

El problema es más fácil de entender primero en base 10. Consideremos la fracción 1/3. Podemos aproximarla como una fracción de base 10 como: 0.3 o, mejor: 0.33 o, mejor: 0.333 y así. No importa cuántos dígitos desees escribir, el resultado nunca será exactamente 1/3, pero será una aproximación cada vez mejor de 1/3.

De la misma manera, no importa cuántos dígitos en base 2 quieras usar, el valor decimal 0.1 no puede representarse exactamente como una fracción en base 2. En base 2, 1/10 es la siguiente fracción que se repite infinitamente

$$0.0001100110011001100110011001100110011001100110011...$$

si nos detenemos en cualquier número finito de bits, y tendremos una aproximación. En la mayoría de las máquinas hoy en día, los double se aproximan usando una fracción binaria con el numerador usando los primeros 53 bits con el bit más significativo y el denominador como una potencia de dos. En el caso de 1/10, la fracción binaria es

$$3602879701896397/2^{55}$$

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

$$0.1 + 0.1 + 0.1 \text{ ó } 0.1 * 3 \text{ ó } 0.1 + 0.2^{20}$$

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

$$0.1, 0.10000000000000001 \text{ y}$$

[20] Para 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
```

0.1000000000000000005551115123125782702118158340454101562 5

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: 9.199999999999999928945

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:

0.020000000000000000416333634234433702658861875534057617187 5

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 =3.14159265358979323846264338327 95;
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:

$$a=1234.567, b=45.67834, c=0.0004$$

$(a+b)+c$:

$1234.567+45.67834$ 1280.24534 se redondea a 1280.245
$1280.245+0.0004$ 1280.2454 se redondea a 1280.245

$a+(b+c)$:

$45.67834+0.0004$ 45.67874
$1234.567+45.6787$ 1280.24574 se redondea a 1280.246

Tampoco son necesariamente distributivos. Es decir $(a+b)\times c$ puede no ser lo mismo que $a\times c+b\times c$:

$$1234.567*3.333333=4115.223,\ 1.234567*3.333333=4.115223,$$
$$4115.223+4.115223=4119.338$$

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(nsimplify(4.56 * 100, tolerance=1e-1))
```

que nos dará el 456 esperado.

SymPy es un paquete matemático simbólico para Python, y nsimplify 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(nsimplify(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 nsimplify podría ser ayudarte a recordar fórmulas medio olvidadas. Por ejemplo, quizás estés oxidado con tus identidades trigonométricas, pero recuerdas que cos(p/6) es algo especial.

```
from sympy import *
print(nsimplify(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 nsimplify incluya expresiones con π y e en su búsqueda.

```
from sympy import *
print(nsimplify(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))
```

obteneindo:

$$22/7$$

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 Pruebes por la Igualdad Cuando se usa Punto Flotante no es recomendable escribir código como el siguiente[21]:

```
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 reaiz 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.

[21] Sin 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 >= o <=, por ejemplo para $a + b <= c$ 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 precisión[22] 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 dobles 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";
```

[22] Los 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 $1 + \epsilon! = 1$ 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 $log0 = -\infty$ y $logx$ 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 $sqrt(-2)$, 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

$$200! = 200 * 199 * 198 * ... * 191 * 190!$$

y así

$$200!/(190!10!) = 200 * 199 * 198 * ... * 191/10!$$

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

$$log(200!/(190!10!)) = Log(200!) - log(190!) - log(10!)$$

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

$$log(1 + x) = x + x^2/2! + x^3/3! + ...$$

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 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 $-log(DBL\_EPSILON)$, 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 $LogitInverse(x) - LogitInverse(y)$
- 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.

10.3 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 $46,225$ mm^2, 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.

11 Apéndice E: Integración Numérica

En análisis numérico, la integración numérica constituye una amplia gama de algoritmos para calcular el valor numérico de una integral definida y, por extensión, el término se usa a veces para describir algoritmos numéricos para resolver ecuaciones diferenciales. El término cuadratura numérica (a menudo abreviado a cuadratura) es más o menos sinónimo de integración numérica, especialmente si se aplica a integrales de una dimensión a pesar de que para el caso de dos o más dimensiones (integral múltiple) también se utiliza.

El problema básico considerado por la integración numérica es calcular una solución aproximada a la integral definida:

$$\int_a^b f(x)dx$$

Este problema también puede ser enunciado como un problema de valor inicial para una ecuación diferencial ordinaria, como sigue:

$$y'(x) = f(x), \qquad y(a) = 0$$

Encontrar $y(b)$ es equivalente a calcular la integral. Los métodos desarrollados para ecuaciones diferenciales ordinarias, como el método de Runge-Kutta, pueden ser aplicados al problema reformulado.

Razones para la integración numérica Hay varias razones para llevar a cabo la integración numérica. La principal puede ser la imposibilidad de realizar la integración de forma analítica. Es decir, integrales que requerirían de un gran conocimiento y manejo de matemática avanzada pueden ser resueltas de una manera más sencilla mediante métodos numéricos. Incluso existen funciones integrables pero cuya primitiva no puede ser calculada, siendo la integración numérica de vital importancia. La solución analítica de una integral nos arrojaría una solución exacta, mientras que la solución numérica nos daría una solución aproximada. El error de la aproximación, que depende del método que se utilice y de qué tan fino sea, puede llegar a ser tan pequeño que es posible obtener un resultado idéntico a la solución analítica en las primeras cifras decimales.

Cuadratura de Gauss En análisis numérico un método de cuadratura es una aproximación de una integral definida de una función. Una cuadratura de Gauss n, es una cuadratura que selecciona los puntos de la evaluación de manera óptima y no en una forma igualmente espaciada, construida para dar el resultado de un polinomio de grado $2n-1$ o menos, elegibles para los puntos xi y los coeficientes wi para $i = 1, ..., n$. El dominio de tal cuadratura por regla es de $[-1, 1]$ dada por:

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

la cuadratura dará resultados precisos sólo si $f(x)$ es aproximado por un polinomio dentro del rango $[-1, 1]$. Si la función puede ser escrita como $f(x) = W(x)g(x)$, donde $g(x)$ es un polinomio aproximado y $W(x)$ es conocido.

$$\int_{-1}^{1} f(x)dx = \int_{-1}^{1} W(x)g(x)dx \approx \sum_{i=1}^{n} w_i g(x_i)$$

Fórmula para calcular w_i también conocida como el método de Gauss-Legendre, los coeficientes están dados por

$$w_i = \frac{2}{(1-x_i^2)\left[P_n'(x_i)\right]^2}$$

donde P_n son los polinomios de Legendre en el intervalo $[-1, 1]$.

Polinomios de Legendre las funciones de Legendre son las soluciones de las ecuaciones diferenciales de Legendre:

$$\frac{d}{dx}\left[\left(1-x^2\right)\frac{d}{dx}P_n(x)\right] + n(n+1)P_n(x) = 0$$

la ecuación diferencial de Legendre puede resolverse usando el método de series de potencias. En general la serie de potencias obtenida converge cuando $|x| < 1$ y en el caso particular cuando n sea un entero no negativo las soluciones forman una familia de polinomios ortogonales llamados polinomios de

Legendre. Cada polinomio de Legendre $P_n(x)$ es un polinomio de grado n. Este se puede ser expresado usando la fórmula:

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}\left[\left(x^2-1\right)^n\right]$$

desarrollando esta fórmula, se obtiene la siguiente expresión para los polinomios de Legendre

$$P_n(x) = \frac{1}{2^n}\sum_{k=0}^{n}\binom{n}{k}^2 (x+1)^{n-k}(x-1)^k$$

esta última expresión es útil en caso de querer elaborar un programa que grafique los polinomios de Legendre.

Los primeros polinomios de Legendre se muestran en la siguiente tabla:

n	$P_n(x)$
0	1
1	x
2	$\frac{1}{2}(3x^2-1)$
3	$\frac{1}{2}(5x^3-3x)$
4	$\frac{1}{8}(35x^4-30x^2+3)$

Así, la lista de coeficientes de w_i y puntos x_i para $n = 1, ..., 5$ esta dada en la tabla siguiente:

Número de puntos	Puntos x_i	Pesos w_i
1	0	2
2	$\pm\sqrt{\frac{1}{3}}$	1
3	0	$\frac{8}{9}$
	$\pm\sqrt{\frac{3}{5}}$	$\frac{5}{9}$
4	$\pm\sqrt{\left(3-2\left(\sqrt{\frac{6}{5}}\right)\right)/7}$	$\frac{18+\sqrt{30}}{36}$
	$\pm\sqrt{\left(3+2\left(\sqrt{\frac{6}{5}}\right)\right)/7}$	$\frac{18-\sqrt{30}}{36}$
5	0	$\frac{128}{225}$
	$\pm\sqrt{5-2\left(\sqrt{\frac{10}{7}}\right)}$	$\frac{322+13\sqrt{70}}{900}$
	$\pm\sqrt{5+2\left(\sqrt{\frac{10}{7}}\right)}$	$\frac{322-13\sqrt{70}}{900}$

Cambio de Intervalos

Los cambios de intervalos van de $[-1,1]$ después de aplicar la cuadratura de Gauss:

$$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}x+\frac{a+b}{2}\right)dx$$

Después de aplicar la cuadratura la aproximación es:

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\sum_{i=1}^{n} w_i f\left(\frac{b-a}{2}x_i+\frac{a+b}{2}\right)$$

Ejemplo de Uso La integral

$$\int_1^5 (x^3+2x^2)dx = 238.66667$$

si se resuelve por cuadratura usando $n=2$, entonces tenemos

$$\frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}x+\frac{a+b}{2}\right)dx = \frac{5-1}{2}\int_{-1}^{1} f\left(\frac{5-1}{2}x+\frac{5+1}{2}\right)dx =$$

$$2\int_{-1}^{1} f\left(2x+3\right)dx \approx 2\sum_{i=1}^{2} w_i f\left(2x+3\right) =$$

$$2\left(w_1 f\left(2x_1+3\right)+w_2 f\left(2x_2+3\right)\right) = 238.6667$$

Integrando en Dos o Más Dimensiones es posible hacer integraciones en dos o más dimensiones y sin pérdida de generalidad, para dos dimensiones tenemos:

Los cambios de intervalos van de $[-1,1]\times[-1,1]$ después de aplicar la cuadratura de Gauss:

$$\int_a^b\int_c^d f(x,y)dxdy = \frac{b-a}{2}\frac{d-c}{2}\int_{-1}^{1}\int_{-1}^{1} f\left(\frac{b-a}{2}x+\frac{a+b}{2},\frac{d-c}{2}x+\frac{c+d}{2}\right)dxdy$$

Después de aplicar la cuadratura la aproximación es:

$$\int_a^b \int_c^d f(x,y)dxdy \approx \frac{b-a}{2}\frac{d-c}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j f\left(\frac{b-a}{2}x_i + \frac{a+b}{2}, \frac{d-c}{2}y_j + \frac{c+d}{2}\right)$$

De esta forma se puede extender la integración por cuadratura Gaussiana para cualquier dimensión que se necesite, esto queda plasmado en el siguiente programa en C:

```
#include<stdio.h>
#include<math.h>
// Arreglo para las Xi
static double PUNTOS[][10] = {
 {0.57735026918962576450914878050 2,-0.57735026918962576450914878050 2,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0},
 {0.77459666924148337703585307995 6,0.0,-0.77459666924148337703585307995 6,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0},
 {0.86113631159405257522394648889 3,0.33998104358485626480266575910 3,
-0.33998104358485626480266575910 3,-0.86113631159405257522394648889 3,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0},
 {0.90617984593866399279762687829 9,0.53846931010568309103631442070 0,0.0,
-0.53846931010568309103631442070 0,-0.90617984593866399279762687829 9,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0},
 {0.93246951420315202781230155449 4,0.66120938646626451366139959502 0,
0.23861918608319690863050172168 1,-0.23861918608319690863050172168 1
,-0.66120938646626451366139959502 0,-0.93246951420315202781230155449 4,
 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0},
 {0.94910791234275852452618968404 8,0.74153118559939443986386477328 1,
0.40584515137739716690660641207 7,0.0,-0.40584515137739716690660641207 7,
-0.74153118559939443986386477328 1,-0.94910791234275852452618968404 8,
 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0},
 {0.96028985649753623168356086856 9,0.79666647741362673959155393647 6,
0.52553240991632898581773904918 9,0.18343464249564980493947614236 0,
-0.18343464249564980493947614236 0,-0.52553240991632898581773904918 9,
 -0.79666647741362673959155393647 6,-0.96028985649753623168356086856 9,
0.0,0.0},
 {0.96816023950762608983557620290 4,0.83603110732663579429942978807 0,
```

```
0.61337143270059039730870203934 1,0.32425342340380892903853801464 3,0.0,
-0.32425342340380892903853801464 3,-0.61337143270059039730870203934 1,
-0.83603110732663579429942978807 0,-0.96816023950762608983557620290 4,0.0},
 {0.97390652851717172007796401208 4,0.86506336668898451073209668842 3,
0.67940956829902440623432736511 5,0.43339539412924719079926594316 6,
0.14887433898163121088462600113 0,-0.14887433898163121088462600113 0,
-0.43339539412924719079926594316 6,-0.67940956829902440623432736511 5,
-0.86506336668898451073209668842 3,-0.97390652851717172007796401208 4}
};
// Arreglo para los pesos
static double PESOS[][10] = {
 {1.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0},
 {0.55555555555555555555555555556,0.88888888888888888888888888889,
0.55555555555555555555555555556,0.0,0.0,0.0,0.0,0.0,0.0,0.0},
 {0.34785484513745385737306394922 2,0.65214515486254614262693605077 8,
0.65214515486254614262693605077 8,0.34785484513745385737306394922 2,
0.0,0.0,0.0,0.0,0.0,0.0},
 {0.23692688505618908751426404072 0,0.47862867049936646804129151483 6,
0.56888888888888888888888888889,0.47862867049936646804129151483 6,
0.23692688505618908751426404072 0,0.0,0.0,0.0,0.0,0.0},
 {0.17132449237917034504029614217 3,0.36076157304813860756983351383 8,
0.46791393457269104738987034399 0,0.46791393457269104738987034399 0,
0.36076157304813860756983351383 8,0.17132449237917034504029614217 3,0.0,
 0.0,0.0,0.0},
 {0.12948496616886969327061143267 9,0.27970539148927666790146777142 4,
0.38183005050511894495036977548 9,0.41795918367346938775510204081 6,
0.38183005050511894495036977548 9,0.27970539148927666790146777142 4,
0.12948496616886969327061143267 9,0.0,0.0,0.0},
 {0.10122853629037625915253135431 0,0.22238103445337447054435599442 6,
0.31370664587788728733796220198 7,0.36268378337836198296515044927 7,
0.36268378337836198296515044927 7,0.31370664587788728733796220198 7,
0.22238103445337447054435599442 6,0.10122853629037625915253135431 0
,0.0,0.0},
 {0.08127438836157441197189215811 05,0.18064816069485740405847203124 3,
0.26061069640293546231874286941 9,0.31234707704000284006863040658 4,
0.33023935500125976316452506928 7,0.31234707704000284006863040658 4,
0.26061069640293546231874286941 9,0.18064816069485740405847203124 3,
0.08127438836157441197189215811 05,0.0},
```

```
{0.06667134430868813759356880989 33,0.14945134915058059314577633965 8,
0.21908636251598204399553493422 8,0.26926671930999635509122692156 9,
0.29552422471475287017389299465 1,0.29552422471475287017389299465 1,
0.26926671930999635509122692156 9,0.21908636251598204399553493422 8,
0.14945134915058059314577633965 8,0.06667134430868813759356880989 33}
};
// Funcion Principal ....
int main(void)
{
int k, P_I = 8;
int j, i;
double x = 0.0;

// Integral en una dimension f(x)= x*x - 3x + 7
for (k = 0; k < P_I; k++)
{
x += PESOS[P_I-2][k] * (PUNTOS[P_I-2][k] *
PUNTOS[P_I-2][k] + 3.0 * PUNTOS[P_I-2][k] + 7.0);
}
printf("El resultado es %2.18lf, debe de ser 14.666667\n",x);

// Integral en una dimension f(x)= sin(x)
x = 0.0;
for (k = 0; k < P_I; k++)
{
x += PESOS[P_I-2][k] * (sinl(PUNTOS[P_I-2][k] ));
}
printf("El resultado es %2.18lf, debe de ser 0.0\n",x);

// Integral en dos dimensiones f(x,y)= (x*x*x - 1)(y - 1)(y - 1)
x = 0.0;
for (i = 0; i < P_I; i++)
{
for (j = 0; j < P_I; j++)
{
x += PESOS[P_I-2][i]*PESOS[P_I-2][j]*( (PUNTOS[P_I-2][j]*
PUNTOS[P_I-2][j]*PUNTOS[P_I-2][j]-1.0)*((PUNTOS[P_I-2][i]-1.0)
*(PUNTOS[P_I-2][i]-1.0)) );
```

```
}
}
printf("El resultado es %2.18lf, debe de ser -5.333333\n",x);

// Integral en tres dimensiones f(x,y,z)=(x*x)(y*y - 1)(z*z*z*z - 2)
x = 0.0;
for (i = 0; i < P_I; i++)
{
for (j = 0; j < P_I; j++)
{
for (k = 0; k < P_I; k++)
{
x += PESOS[P_I-2][i]*PESOS[P_I-2][j]*PESOS[P_I-2][k]*
((PUNTOS[P_I-2][k]*PUNTOS[P_I-2][k])*(PUNTOS[P_I-2][j]*
PUNTOS[P_I-2][j]-1.0)*(PUNTOS[P_I-2][i]*PUNTOS[P_I-2][i]*
PUNTOS[P_I-2][i]*PUNTOS[P_I-2][i]-2.0) );
}
}
}

printf("El resultado es %2.18lf, debe de ser 3.2\n",x);
getchar();
}
```

12 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] B. D. Reddy; *Introductory Functional Analysis - With Applications to Boundary Value Problems and Finite Elements*. Springer 1991. 30, 44, 55, 63, 97, 99, 100

[2] B. I. Wohlmuth; *Discretization Methods and Iterative Solvers Based on Domain Decomposition*. Springer, 2003. 5, 121

[3] I. Herrera; *Método de Subestructuración* (Notas de Curso en Preparación). Instituto de Geofísica, (UNAM).

[4] I. Herrera y R. Yates; UnifieD Multipliers-Free Theory of Dual-Primal Domain Decomposition Methods, ...

[5] 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.

[6] M. Diaz; *Desarrollo del Método de Colocación Trefftz-Herrera Aplicación a Problemas de Transporte en las Geociencias*. Tesis Doctoral, Instituto de Geofísica, UNAM, 2001.

[7] P.G. Ciarlet, J. L. Lions; *Handbook of Numerical Analysis, Vol. II*. North-Holland, 1991. 30, 99, 100

[8] W. Rudin; *Principles of Mathematical Analysis*. McGraw-Hill International Editions, 1976. 97

[9] F. Brezzi y M. Fortin; Mixed and Hibrid Finite Element Methods, Springer, 1991. 30

[10] K. Hutter and K Jöhnk, *Continuum Methods of Physical Modeling*, Springer-Verlag Berlin Heidelberg New York, 2004. 4

[11] J. L. Lions and E. Magenes, *Non-Homogeneous Bounduary Value Problems and Applicarions* Vol. I, Springer-Verlag Berlin Heidelberg New York, 1972. 30, 44, 55, 63, 99, 100

[12] A. Toselli, O. Widlund, *Domain Decomposition Methods - Algorithms and Theory*. Springer, 2005.

[13] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*. Clarendon Press Oxford, 1999. 5

[14] A. Quarteroni and A. Valli, *Numerical Approximation of Partial Diferential Equations*. Springer, 1994.

[15] B. Dietrich, *Finite Elements: Theory, Fast Solvers*, and Applications in Solid Mechanics, Cambridge University, 2001.

[16] B. F. Smith, P. E. BjØrstad, W. D. Gropp, *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996. 5

[17] Fuzhen Zhang, *The Schur Complement and its Aplications*, Springer, Numerical Methods and Algorithms, Vol. 4, 2005.

[18] L. F. Pavarino and A. Toselli, *Recent Developments in Domain Decomposition Methods*. Springer, 2003. 121

[19] M.B. Allen III, I. Herrera & G. F. Pinder, *Numerical Modeling in Science And Engineering.* John Wiley & Sons, Inc . 1988. 4, 14, 22, 64, 116, 117, 120, 121, 142

[20] R. L. Burden and J. D. Faires, *Análisis Numérico.* Math Learning, 7 ed. 2004. 116, 142

[21] S. Friedberg, A. Insel, and L. Spence, *Linear Algebra*, 4th Edition, Prentice Hall, Inc. 2003. 96, 116, 142

[22] Y. Saad, *Iterative Methods for Sparse Linear Systems.* SIAM, 2 ed. 2000. 105, 111, 113, 117, 120, 124

[23] Y. Skiba, *Métodos y Esquemas Numéricos, un Análisis Computacional.* UNAM, 2005. 116, 121, 127, 142

[24] M. Diaz, I. Herrera, *Desarrollo de Precondicionadores para los Procedimientos de Descomposición de Dominio.* Unidad Teórica C, Posgrado de Ciencias de la Tierra, 22 pags, 1997.

[25] I. Herrera, *Un Análisis del Método de Gradiente Conjugado.* Comunicaciones Técnicas del Instituto de Geofísica, UNAM, Serie Investigación, No. 7, 1988. 124, 130

[26] G. Herrera, Análisis de Alternativas al Método de Gradiente Conjugado para Matrices no Simétricas. Tesis de Licenciatura, Facultad de Ciencias, UNAM, 1989. 130

[27] W. Gropp, E. Lusk and A. Skjellem, *Using MPI, Portable Parallel Programming Whit the Message Passing Interface.* Scientific and Engineering Computation Series, 2ed, 1999.

[28] I. Foster, *Designing and Building Parallel Programs.* Addison-Wesley Inc., Argonne National Laboratory, and the NSF, 2004.

[29] Jorge L. Ortega-Arjona, *Patterns for Parallel Software Design*, Wiley series in Software Design Patterns, 2010.

[30] DDM Organization, *Proceedings of International Conferences on Domain Decomposition Methods*, 1988-2012.

http://www.ddm.org and http://www.domain-decomposition.com

[31] Toselli, A., and Widlund O. *Domain decomposition methods- Algorithms and theory*, Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2005, 450p. 6

[32] 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.

[33] Mandel J. and Tezaur R. *Convergence of a Substructuring Method with Lagrange Multipliers*, Numer. Math. 73 (1996) 473-487.

[34] Farhat C., Lesoinne M. Le Tallec P., Pierson K. & Rixen D. *FETI-DP a Dual-Primal Unified FETI method, Part 1: A Faster Alternative to the two-level FETI Method*, Int. J. Numer. Methods Engrg. 50 (2001) 1523-1544.

[35] Farhat C., Lesoinne M. and Pierson K. *A Scalable Dual-Primal Domain Decomposition Method*, Numer. Linear Algebra Appl. 7 (2000) 687-714.

[36] Mandel J. and Tezaur R. *On the Convergence of a Dual-Primal Substructu-ring Method*, Numer. Math. 88(2001), pp. 5443-558.

[37] Mandel, J. *Balancing Domain Decomposition*. Comm. Numer. Meth. Engrg., 9:233-241, 1993.

[38] Mandel J. and Brezina M., *Balancing Domain Decomposition for Problems with Large Jumps in Coefficients,* Math. Comput. 65 (1996) 1387-1401.

[39] Dohrmann C., *A Preconditioner for Substructuring Based on Constrained Energy Minimization*. SIAM J. Sci. Comput. 25 (2003) 246-258.

[40] Mandel J. and Dohrmann C., *Convergence of a Balancing Domain Decomposition by Constraints and Energy Minimization*. Numer. Linear Algebra Appl. 10 (2003) 639-659.

[41] Da Conceição, D. T. Jr., *Balancing Domain Decomposition Preconditioners for Non-symetric Problems*, Instituto Nacional de Matemática pura e Aplicada, Agencia Nacional do Petróleo PRH-32, Rio de Janeiro, May. 9, 2006.

[42] J. Li and O. Widlund, *FETI-DP, BDDC and block Cholesky Methods*, Int. J. Numer. Methods Engrg. 66, 250-271, 2005.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] Agustín Alberto Rosas Medina, *El Número de Péclet y su Significación en la Modelación de Transporte Difusivo de Contaminantes,* Tesis para obtener el titulo de Matemático, UNAM, 2005.

[48] Omar Jonathan Mendoza Bernal, Resolución de Ecuaciones Diferenciales Parciales Mediante el Método de Diferencias Finitas y su Paralelización, Tesis para obtener el titulo de Matemático, UNAM, 2016.

[49] 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.

[50] Tezaur R., *Analysis of Lagrange Multipliers Based Domain Decomposition*. P.H. D. Thesis, University of Colorado, Denver, 1998.

[51] Herrera I. and Rubio E., *Unified Theory of Differential Operators Acting on Discontinuos Functions and of Matrices Acting on Discontinuous*

Vectors, 19th International Conference on Domain Decomposition Methods, Zhangjiajie, China 2009. (Oral presentation). Internal report #5, GMMC-UNAM, 2011.

[52] Valeri I. Agoshkov, *Poincaré-Steklov Operators and Domain Decomposition Methods in Finite Dimensional Spaces*. First International Symposium on Domain Decomposition Methods for Partial Differential Equations, pages 73-112, Philadelphia, PA, 1988. SIAM. Paris, France, Junuary 7-9, 1987.

[53] Toselli, A., *FETI Domain Decomposition Methods for Escalar Advection-Diffusion Problems*. Compututational Methods Appl. Mech. Engrg. 190. (2001), 5759-5776.

[54] C.T. Keller, *Iterative Methods for Linear and Nonlinear Equations*, Societe for Industrial and Applied Mathematics, 1995. 124, 127

[55] 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.

[56] Zdengk Dostdl and David Hordk. *Scalability and FETI Based Algorithm for Large Discretized Variational Inequalities*. Math. Comput. Simulation, 61(3-6): 347-357, 2003. MODELLING 2001 (Pilsen).

[57] Charbel Farhat, Michel Lesoinne, and Kendall Pierson. *A Scalable Dual-Primal Domain Decomposition Method*. Numer. Linear Algebra Appl., 7(7-8):687-714, 2000.

[58] 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.

[59] 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.

[60] 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 Solid Mechanics Simulation*. In ACM/IEEE Proceedings of SC02: High Performance Networking and Computing. Gordon Bell Award, pages 1-19, 2002.

[61] 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.

[62] Petter E. BjØrstad and Morten Skogen. *Domain Decomposition Algorithms of Schwarz Type, Designed for Massively Parallel Computers*. In David E. Keyes, Tony F. Chan, Gerard A. Meurant, Jeffrey S. Scroggs, and Robert G. Voigt, editors. Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, pages 362-375, Philadelphia, PA, 1992. SIAM. Norfolk, Virgina, May 6-8, 1991.

[63] Yau Shu Wong and Guangrui Li. *Exact Finite Difference Schemes for Sol-ving Helmholtz Equation at any Wavenumber*. International Journal of Numerical Analysis and Modeling, Series B, Volume 2, Number 1, Pages 91-108, 2011.

[64] Zhangxin Chen, Guanren Huan and Yuanle Ma. *Computational Methods for Multiphase Flow in Porous Media*, SIAM, 2006.

[65] Jos Stam. SIGGRAPH 99, Proceedings of the 26th annual conference on Computer Graphics and Interactive Techniques, Pages 121-128, ACM, 1999.

[66] 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.

[67] 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.

[68] 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.

[69] 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)

[70] 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).

[71] K. Hutter and K. Jöhnk, *Continuum Methods of Physical Modeling.* Springer-Verlag Berlin Heidelber New York 2004. 22

[72] 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.

[73] X. O. Olivella, C. A. de Sacribar, *Mecánica de Medios Continuos para Ingenieros.* Ediciones UPC, 2000. 22

[74] Herrera, I. and Pinder, G. F., *Mathematical Modelling in Science and Engineering: An Axiomatic Approach*, Wiley, 243p., 2012. 4, 18, 22

[75] 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.

[76] Antonio Carrillo-Ledesma, Herrera, I, Luis M. de la Cruz, *Parallel Algorithms for Computational Models of Geophysical Systems*, Geofísica Internacional, en prensa, 2013.

[77] 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. 192(35-36):3799-3830, 2003.

[78] Holger Brunst, Bernd Mohr. *Performance Analysis of Large-Scale OpenMP and Hybrid MPI/OpenMP Applications with Vampir NG.* IWOMP 2005: 5-14.

[79] 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).

[80] Doxygen, *Gererate Documentation from Source Code*.
http://www.stack.nl/∼dimitri/doxygen/

[81] XMPI, *A Run/Debug GUI for MPI*.
http://www.lam-mpi.org/software/xmpi/

[82] VAMPIR, *Performance Optimization*.
http://www.vampir.eu/

[83] Handbook of Floating-Point Arithmetic 2010th Edition, Jean-Michel Muller, Nicolas Brisebarre, Et alii, Birkhäuser, 2010.

[84] What Every Computer Scientist Should Know About Floating-Point Arithmetic, David Goldbergm, ACM Computing Surveys, Vol 23, No 1, March 1991

[85] 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.)

[86] 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)

[87] IEEE, https://www.ieee.org/

[88] Intel, https://www.intel.la

[89] AMD, https://www.amd.com

[90] ARM, https://www.arm.com/

[91] 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