UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERIA
DEPARTAMENTO DE INGENIERIA INFORMATICA

"DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO PARALELO PARA LA SOLUCIÓN DE LA ECUACIÓN DIFERENCIAL DEL TRANSPORTE DE LA LUZ EN MEDIOS DIFUSOS, MEDIANTE ELEMENTOS FINITOS"

OSCAR FRANCISCO ROJAS DIAZ

“DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO PARALELO PARA LA SOLUCIÓN DE LA ECUACIÓN DIFERENCIAL DEL TRANSPORTE DE LA LUZ EN MEDIOS DIFUSOS, MEDIANTE ELEMENTOS FINITOS”

Trabajo de graduación presentado a la Facultad de Ingeniería, en cumplimiento parcial de los requisitos exigidos para optar al grado de Magister en Ingeniería Informática.

UNIVERSIDAD DE SANTIAGO DE CHILE
SANTIAGO - CHILE
2009

"DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO PARALELO PARA LA SOLUCIÓN DE LA ECUACIÓN DIFERENCIAL DEL TRANSPORTE DE LA LUZ EN MEDIOS DIFUSOS, MEDIANTE ELEMENTOS FINITOS"

OSCAR FRANCISCO ROJAS DIAZ

Este trabajo de Graduación bajo la supervisión del profesor guía Sr. Fernando Rannou, del Departamento de Ingeniería Informática y ha sido aprobado por los miembros de la comisión calificadora, de, candidato, Sr ……..y Sr……..

……..

……..

……..

Profesor Guía

……

Director

Tabla de contenido

RESUMEN

En esta tesis se diseñó e implementó una aplicación paralela que resuelve la ecuación de difusión de fotones de luz en un medio turbio, mediante elementos finitos. Se utilizó el método de Galerkin en la formulación variacional de la ecuación de difusión de fotones de luz para un dominio bidimensional con elementos finitos triangulares de tres nodos. El sistema lineal resultante del método de elementos finitos se resolvió utilizando el método LSQR (Least Square QR). Se realizaron pruebas de validación con el algoritmo Monte Carlo, que es el método estándar utilizado para resolver problemas de difusión de partículas. El software se implementó utilizando las librerías estándar de C++, STL y MPI utilizando como paradigma de programación el modelamiento orientado a objetos. Se realizaron pruebas de rendimiento en el clúster del Departamento de Ingeniería Informática de la Universidad de Santiago de Chile, donde se realizó un análisis del uso de memoria, tiempos de cómputo y rendimiento de la aplicación paralela.

ABSTRACT

In this thesis it was designed and it implemented a parallel application that solves the equation of diffusion of photons of light in a half cloudy one, by means of finite elements. The method of Galerkin was used in the formulation variacional of the equation of diffusion of photons the light in domain two-dimensional with finite elements triangular of three nodes. The system lineal resultant of the method of finite elements was solved using the method LSQR (Least Square QR). Were carried out validation tests with the algorithm Monte Carlo that is the standard method used to solve problems of diffusion of particles. The software was implemented using the standard bookstores of C++, STL and MPI using the programming the modeled guided to objects as paradigm the programming. Were carried out yield tests in the cluster the computers of the Department of Computer Engineering of the University of Santiago from Chile, where he/she was carried out an analysis of the use by heart, times of computation and yield of the parallel application.

1. INTRODUCCIÓN

1.1. IMAGEN MOLECULAR

Imagen molecular es toda aquella modalidad de imagen biomédica capaz de proporcionar información que permita detectar procesos celulares a nivel molecular en vivo, y que permite el estudio de dichos procesos de forma remota y no invasiva, sin perturbar el sistema bajo estudio (Massoud, y otros, 2003). La mayoría de las técnicas de imagen molecular se han desarrollado sobre modelos de roedores (ratas y ratones), apoyados en ensayos previos *in vitro*. En imagen molecular, sondas moleculares son enviadas contra dianas biológicas específicas, con el fin de obtener una imagen que permita estudiar procesos celulares y/o moleculares en su medio natural intacto o en el medio característico de un proceso patológico. La finalidad es obtener un mayor conocimiento de las diferentes enfermedades y mejorar así su diagnóstico y tratamiento.

Estas técnicas de imagen son intrínsecamente moleculares, y su uso clínico en la actualidad está ampliamente extendido y aceptado. Las imágenes moleculares tienen sus raíces en la medicina nuclear y en muchos sentidos es una extensión directa de esta disciplina (Massoud, y otros, 2003). La medicina nuclear es una disciplina centrada en el uso de radiofármacos que están formados por un fármaco transportador y un isótopo radiactivo. Estos radiofármacos se aplican dentro del organismo humano por diversas vías (la más utilizada es la vía intravenosa). Una vez que el radiofármaco está dentro del organismo, se distribuye por diversos órganos dependiendo del tipo de radiofármaco empleado. La distribución del radiofármaco es detectado por un aparato detector de radiación llamada cámara gamma y los datos se almacenan digitalmente en un procesador. Luego se puede procesar la información obteniendo imágenes de todo el cuerpo o del órgano en estudio. Estas imágenes, a diferencia de la mayoría de las obtenidas en radiología, son imágenes funcionales, es decir, muestran como están funcionando los órganos explorados.

En imagen molecular la principales variantes son: la imagen plana (gammagrafía), la tomografía computacional por emisión de fotón único (SPECT) y la tomografía por emisión de positrones (PET). De estas tres técnicas, PET es la que mejores características presenta para la investigación biomédica por su mayor sensibilidad, resolución espacial y temporal, y por su carácter cuantitativo.

Un considerable esfuerzo se ha dirigido hacia el desarrollo de distintas técnicas de imagen no invasiva y de alta resolución las cuales permitan obtener información necesaria para la generación de imágenes moleculares. Una de dichas técnicas es "técnicas de imagen óptica" las cuales están basadas en el uso de fotones de luz para la obtención de imágenes. Se utilizan generalmente en biología celular y molecular *in vitro* (microscopios de fluorescencia) o *in vivo* (bioluminiscencia y fluorescencia) o bien en simulaciones utilizando tejidos inorgánicos con propiedades moleculares idénticas a las naturales.

1.2. IMAGEN ÓPTICA, TOMOGRAFÍA ÓPTICA Y BIOLUMISCENCIA

La imagen óptica es probablemente la técnica más extendida actualmente en la investigación biomédica para su uso *in vitro* e in *vivo*, debido a su sencillez y a su bajo costo. Además la imagen óptica también presenta la ventaja ofrecer una alta resolución temporal y una extraordinaria sensibilidad. Sin embargo el traslado de la imagen óptica a la experimentación *in vivo* no está exento de complicaciones, derivadas de la dificultad que supone la detección de la luz emitida en el interior del organismo vivo bajo estudio, dado que el tejido biológico es parcialmente opaco a las longitudes de onda de la luz utilizada.

La tomografía óptica es una modalidad de tomografía relativamente nueva de imágenes en que las imágenes y las propiedades ópticas del medio se calculan sobre la base de las mediciones de luz visible en la superficie del objeto y son una valiosa herramienta de diagnóstico clínico.

En la tomografía se involucran la proyección de datos provenientes de múltiples direcciones y el envío de estos datos para la creación de una reconstrucción tomográfica. Actualmente, las tomografías se obtienen utilizando diferentes fenómenos físicos, tales como rayos X (Tomografía Computarizada - CT), rayos gamma (Tomografía Computarizada por Emisión de Foton Unico - SPECT), aniquilación de electrones y positrones (Tomografía por Emisión de Positrones - PET), campos magnéticos (Imagen de Resonancia Magnética - MRI) y ultrasonido (ultrasonografía). En la figura 1.1 se muestran las tecnologías tomográficas: CT, SPECT, PET y MRI.

Aunque las modalidades de imágenes médicas obtenidas con CT, MRI y PET son las más utilizadas en la actualidad para obtención de información clínica de carácter

funcional, los nuevos sistemas de imágenes son obligatorios. Especialmente por el carácter no invasivo, relativamente baratos, portátiles y son sistemas de formación de imágenes que podrían ser utilizados para diagnósticos ambulatorios y seguimiento continuo de pacientes. Además, las modalidades funcionales de imagen proporcionan información fisiológica y efectos de fármacos con imágenes temporales (Tarvainen, 2006).

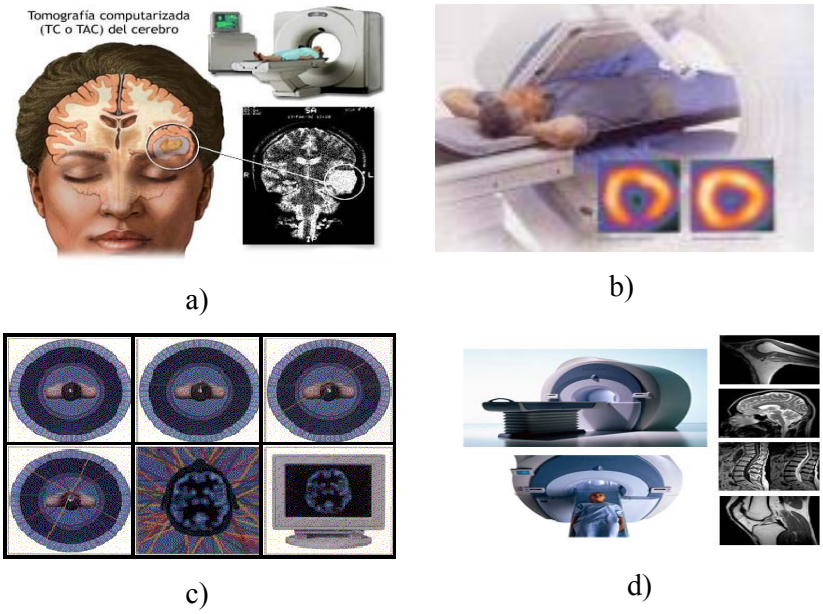


Figura 1.1.- Tecnologías tomográficas: a) Tomografía computarizada, b) tomografía computarizada por emisión de fotón único, c) tomografía por emisión de positrones, d) imagen de resonancia magnética.

La bioluminiscencia es la generación de luz generada en animales que contienen el gen de la luciferasa. Estos genes se encuentran espontáneamente en animales como la luciérnaga o medusa y también pueden ser introducidos mediante técnicas de ingeniería genética en diferentes células de interés (por ejemplo, células tumorales). Las imágenes de bioluminiscencia pueden ser obtenidas con cámaras digitales de alta resolución, las cuales capturan la luz que sale del animal y luego generar imágenes de proyección (imagen 2d). En la figura 1.2 se muestra como es el proceso de captura de la luz que sale por la piel de un animal.

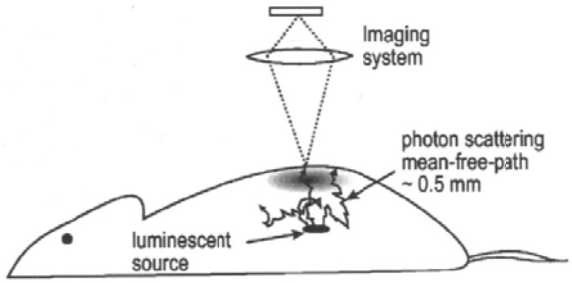


Figura 1.2.-: *Captura de imagen en la superficie del animal.*

Para que se produzca biolumiscencia, la reacción química requiere fundamentalmente de dos elementos: la luciferina y la luciferasa, que cuando se unen en una "foto proteína", puede producir luz cuando un ión particular (calcio) se añade al sistema. Bioluminiscencia no es igual a "fluorescencia" o "fosforescencia", en donde la fluorescencia es inducida cuando la energía de una fuente de luz es absorbida y remitida como otro fotón, a diferencia de la bioluminiscencia, en que la energía para la excitación proviene de una reacción química y no de una fuente de luz. La ventaja más importante de las técnicas de bioluminiscencia es que no presentan actividad de fondo y por lo tanto permiten detectar señales de muy pequeña intensidad. Además no producen daño en los tejidos ya que no utilizan energías ionizantes. La figura 1.3.a muestra una típica imagen de bioluminiscencia de un ratón, capturada con una cámara CCD.

a) b)

Figura 1.3.- *Imágenes de proyección de bioluminiscencia (a) y gammagrafía (b) obtenidas del mismo animal de experimentación 10 días después de la implantación subcutánea en el hombro derecho de células de cáncer de mama.*

Las problemáticas principales para el desarrollo de este tipo de sistemas de detección son la determinación exacta de la profundidad desde donde se emitió la señal, ya que a diferencia de otras técnicas de imagen molecular como PET y SPECT, donde los

fotones gamma viajan en línea recta, en bioluminiscencia los fotones de luz chocan elásticamente miles de veces antes de escapar del animal. Por lo tanto la teoría de reconstrucción tomográfica no se aplica directamente a la bioluminiscencia. En la figura 1.4 se muestra la interacción que los fotones de luz tienen en su ruta al momento de atravesar las moléculas que componen a un tejido orgánico.

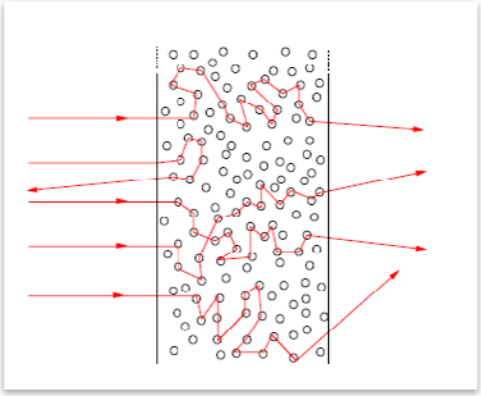


Figura 1.4.- *Interacción que los fotones de luz sufren al atravesar las moléculas que componen la materia orgánica.*

Una primera etapa para la creación tomográfica en bioluminiscencia es entender y modelar cómo los fotones de luz recorren los tejidos antes de escapar del animal y ser detectados por detectores especializados. Es decir, antes de poder estimar la distribución de las fuentes de luz, a partir de la información captada por los detectores, es de vital importancia poder estimar lo que los detectores captarían dado una distribución conocida de fuentes de luz. El problema planteado es tratado principalmente a través de dos modalidades, una física y otra simulada. La física es donde directamente se induce a la emisión de fotones en componentes inorgánicos que tienen propiedades similares a las orgánicas con respecto a las distintas masas y factores de dispersión que tienen los órganos (Ej.: cerebro, riñones, hígados, etc.). A estos componentes se les conoce como Phantom (ver Figura 1.5) para luego utilizarlo en humanos (ver Figura 1.6). La obtención de información es a través de máquinas compuestas por sensores ópticos que captan las emisiones de luz y a partir de esos datos realizan experimentos y estudios que permiten obtener una ubicación aproximada de las fuentes de luz.

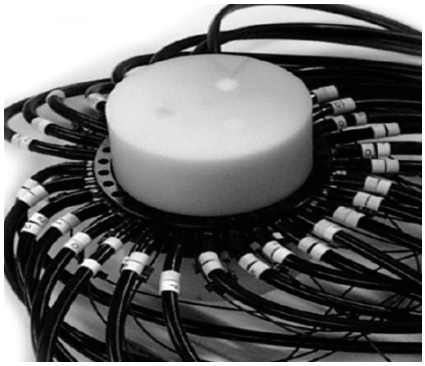


Figura 1.5.- *Fotografía de un Phanton rodeado de sensores ópticos.*

Otra modalidad en la cual se obtienen datos reales, es utilizando ratones de laboratorio, donde el ratón es intervenido genéticamente para que en zonas específicas de su cuerpo se produzca una reacción química en la cual se produzca luz. Esta luz es percibida desde el exterior con sensores ópticos, que captan las intensidades de luz de su piel (Chaudhari, 2006). Generalmente se utilizan ratones sin pelo y con colores de piel claros. La técnica simulada es a través de software donde el problema principal radica en la solución a la ecuación diferencial del transporte de la luz (Arridge, 1999) (Tarvainen, 2006).

Esta solución es abordada por distintos métodos matemáticos, donde destacan principalmente por su adecuación computacional y aceptable aproximación, el método estocástico Monte Carlo y el modelo determinista de Elementos Finitos.

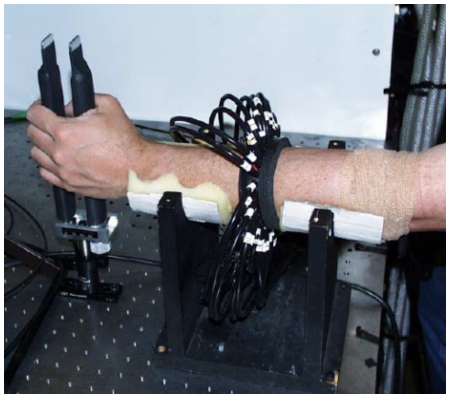


Figura 1.6.- *Fotografía de implementación de maquinas detectoras de fotones en humanos*

1.3. MÉTODOS DE APROXIMACIÓN DE LA DIFUSIÓN DE FOTONES

1.3.1. Método Monte Carlo

El método de Monte Carlo proviene del trabajo realizado en el desarrollo de la bomba atómica durante la segunda guerra mundial. Este trabajo con llevaba la simulación de problemas probabilísticos de hidrodinámica concernientes a la difusión de neutrones en el material de fusión, la cual posee un comportamiento eminentemente aleatorio.

El método Monte Carlo aplicado a la difusión de fotones construye un modelo estocástico, que basándose en las funciones de densidad modela secuencialmente eventos individuales de una variable aleatoria, es decir se siguen todos los eventos o interacciones que sufre cada fotón desde su origen hasta que alcanza una condición terminal (se sale del dominio, disminución de su energía, etc.). Este proceso se repite para todas las partículas creadas en el proceso. Por ejemplo, en el caso de la difusión de fotones a partir de una fuente, el método Monte Carlo utiliza un modelo probabilístico para asignar una dirección al fotón.

Posteriormente utiliza otro modelo probabilístico para estimar una distancia libre media que recorrerá antes de tener una colisión con una partícula específica. Esta última partícula tiene definida características físicas de dispersión, absorción y en algunos casos características energéticas. La energía y dirección de las partículas dispersadas son variables aleatorias que también se calculan en la simulación. Luego se realiza un conteo por unidad de área y se obtiene la densidad de fotones en un lugar específico del modelo geométrico conexo donde se estimaron las trayectorias y ubicaciones antes de la desaparición del fotón.

La implementación computacional del método Monte Carlo aplicado al modelo de difusión de fotones de luz es muy sencilla y ofrece una buena solución, pero el problema radica principalmente en el tiempo necesario para el cálculo, ya que en problemas de difusión de fotones, el dominio puede contener una o más fuentes de fotones las cuales pueden tener densidades del orden de millones de fotones. Además si se desea aproximar la solución del modelo de difusión de fotones con el método Monte Carlo en geometrías más complejas que contengan muchas áreas o volúmenes (geometría no homogénea) con diferentes factores de dispersión se debe identificar en cada instante la ubicación del fotón de luz, lo cual no es necesario de evaluar si se

dispone de una geometría homogénea. Este proceso de seguimiento de la ruta del fotón de luz requiere un costo computacional adicional, ya que se debe estimar si en el recorrido libre medio del fotón de luz, este pasa por algún lugar geométrico en que exista una dispersión distinta a la posición de origen.

Esta estimación requiere la implementación de métodos que identifiquen las distintas áreas o volúmenes por las cuales la ruta del fotón atraviesa, donde es necesario saber el lugar especifico en la ruta del fotón en que se intercepta con el nuevo volumen, para que a partir de ese punto geométrico calcular la nueva dirección y recorrido libre medio que tendrá el fotón en su ruta. En la figura 1.7 se ilustra el proceso de dispersión y recorrido del fotón de luz en los dos tipos de geometrías (homogénea y no homogénea).

Entonces el uso del método de Monte Carlo está limitado al tiempo de cálculo requerido en los procesos de seguimiento de la ruta de los fotones de luz y depende directamente de tipo de geometría que se utilice, las variables físicas del medio y al número de fotones de luz que se desean estimar.

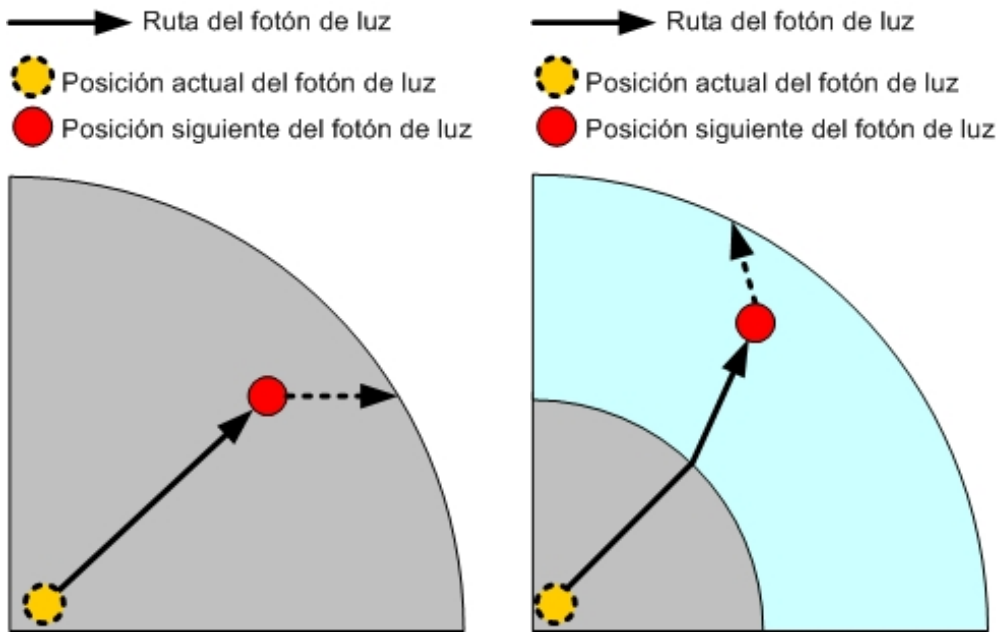


Figura 1.7.- *Ruta de un fotón de luz: a) ruta del fotón de luz en una geometría homogénea, b) ruta del fotón de luz en una geometría no homogénea.*

1.3.2. Método de Elementos Finitos

El método de Elementos Finitos se basa en dividir el dominio del problema y trabajar directamente sobre la ecuación que gobierna el problema, y de esta manera aproximar una solución a dicha ecuación. Esta aproximación se realiza por particiones, para luego unir todas las particiones enlazando los datos de sus fronteras. El problema finalmente queda expresado en un sistema de ecuaciones, el cual puede ser resuelto usando técnicas numéricas disponibles para resolver sistemas de ecuaciones, a diferencia del método Monte Carlo donde el problema es tratado estadísticamente y el seguimiento de las trayectorias se realiza partícula por partícula o por grupos de partículas.

La aproximación de la difusión de fotones de luz con elementos finitos es válida para aproximaciones a escalas de centímetros, a diferencia del método Monte Carlo, donde el problema puede ser modelado en un espacio que puede tender a infinito. A pesar de ello la ventaja principal radica en la posibilidad de que no influye mayormente el número de fuentes ni tampoco el número de fotones de luz a estimar, si no que su complejidad computacional se basa en el orden del sistema de ecuaciones que depende directamente del número de particiones y geometría utilizada en la aproximación.

La otra ventaja sustancial está en que se puede modelar la geometría con cualquier número de subdominios con distintos valores de dispersión sin consecuencias en términos de tiempo de ejecución. Entonces el uso de geometrías homogéneas o no homogéneas en la aproximación del problema no influirá en un mayor costo computacional como sucede con el método Monte Carlo.

1.4. PROBLEMÁTICA

1.4.1 Modelamiento de la difusión de fotones de luz

En el modelamiento de difusión de fotones de luz, el número de fotones a estimar es altísimo, este cálculo a parte de no ser trivial numéricamente es computacionalmente muy costoso. Los dos modelos de aproximación de difusión de fotones de luz presentados en la sección 1.3 difieren en la forma en cómo realizan la aproximación, pero con ambos se obtienen resultados similares. Las diferencias radican principalmente en el tipo de geometrías de los dominios, la complejidad en función de los tiempos de cómputo de cada método y en algunos casos en la calidad de la solución.

La aproximación de la difusión de fotones de luz es gobernada por los factores de absorción y dispersión, los cuales dependen de cómo las moléculas están estructuradas. El proceso de dispersión es tratado probabilísticamente, ya que a pesar de los avances tecnológicos, no es posible obtener la curvatura exacta del exterior de los átomos que componen a las moléculas y los fotones rebotan constantemente al recorrer una masa antes de perder toda su energía.

En el caso de la imagen óptica aplicada al problema de difusión de fotones de luz, las distancias que se estiman en imagen óptica son a escalas de centímetros y en algunos casos milímetros, donde estas distancias son las que recorrerían los fotones si se tratase por ejemplo de detectar cáncer a la piel o de mamas en humanos. Estas distancias al ser pequeñas permiten que ambos modelos sean eficientes principalmente porque los fotones al recorrer estas distancias en tejidos orgánicos no pierden toda su energía y pueden ser captados con los sensores ópticos especializados.

A pesar que ambas soluciones utilizan funciones de densidad probabilísticas al utilizar el factor de dispersión de la masa y establecer una posible nueva ruta del fotón de luz, estos provienen de una fuente de fotones isotrópica lo cual indica que las propiedades del medio son idénticas en todas las direcciones en la fuente, por lo cual existe una distribución uniforme en la emisión de fotones en las fuentes y prácticamente no existen fotones que no se dirijan hacia el exterior del dominio y las variaciones en la direcciones son en su mayoría hacia la misma orientación en que provenían de la fuente.

Esto permite que al estimar un alto el número de fotones, el error posible en las estimaciones globales disminuya, ya que se obtiene una distribución de densidad de fotones de luz global y no necesariamente el recorrido exacto de un fotón. Además como la estimación es en muy pequeñas distancias, el error que se puede producir en la aproximación es mínimo en comparación con el error producido en una masa de gran tamaño, ya sea por el error acumulado en la dispersión en distancias mayores o bien por qué los valores energéticos de los fotones son prácticamente mínimos y no existiría suficiente información para que los sensores ópticos especializados captasen las intensidades de luz en la superficie de objeto o masa en estudio.

1.4.2 Resumen de problemática

Las técnicas de imagen óptica utilizadas para la construcción de imágenes moleculares se basan principalmente en la detección de fotones de luz mediante detectores posicionados fuera del objeto. Imagen molecular facilita el seguimiento y detección de procesos patológicos. La efectividad de los procesos de detección de fotones de luz está directamente relacionada con las propiedades de las fuentes de fotones en el animal. Además del hardware necesario para la detección de luz está y los factores de dispersión y absorción del medio en estudio.

Por tal razón la simulación de los procesos de difusión de la luz en las materias orgánicas juegan un papel importante en el desarrollo de este tipo de tecnologías, ya sea a nivel de calibración de los detectores de fotones como en la reconstrucción de la imágenes tomográficas.

También cabe mencionar que para solucionar el problema de difusión es necesaria la implementación de software capaz de resolver los problemas que presentan las características físico-biológicas en función de las diversas estructuras moleculares que componen la materia biológica.

Además como todo trabajo de desarrollo de software científico éste debe ser portable y de fácil escalamiento a otro tipo de funcionalidades relacionadas con el tema de imagen óptica.

En el trabajo que se presenta a continuación se muestra el desarrollo de una aplicación computacional paralela con programación orientada a objetos que permite realizar una aproximación determinista de la difusión de fotones en una composición molecular simulada en un espacio de dos dimensiones, el cual sea útil para el desarrollo de técnicas de imagen óptica y por consiguiente en la desarrollo de aplicaciones de imagen molecular.

1.5. OBJETIVOS DEL PROYECTO DE TESIS

1.5.1. Objetivo General

Diseñar y construir un método computacional paralelo que resuelva la ecuación de transporte de fotones de luz mediante elementos finitos en medios turbios.

1.5.2. Objetivos Específicos

1. Diseñar, modelar e implementar con programación paralela la solución de la ecuación diferencial del transporte de la luz en medios difusos.
2. Validar la solución con el método Monte Carlo.
3. Evaluar esta solución en el Clúster del Departamento de Ingeniería Informática.
4. Realizar experimentos con la implementación paralela de la ecuación de transporte de la luz.
5. Concluir sobre la utilidad del Software.

1.5.3 Organización del documento

La organización del documento está distribuida de la siguiente manera: En el capítulo 2 se estudia el modelo de transporte de la luz en materia biológica, donde se utiliza la ecuación de trasferencia de la radiación para aproximar la difusión de fotones de luz en medios difusos. En el capítulo 3 se introduce el método de elementos finitos y los conceptos matemáticos más relevantes de este método. En el capítulo 4 se deriva la formulación variacional de la ecuación de difusión de fotones de luz y se la modela mediante elementos finitos. En el capítulo 5 se modela el método de elementos finitos aplicado al problema de difusión de fotones de luz con modelamiento orientado a objetos. En el capítulo 6 de modelan y desarrollan los algoritmos paralelos propuestos y en el capítulo 7 se realiza un análisis de resultados de la implementación paralela, comparando los resultados con el método Monte Carlo y finalmente en el capítulo 8 se discuten las conclusiones generales del trabajo.

2. MODELO DE TRANSPORTE DE LA LUZ EN MATERIA BIOLÓGICA

2.1. INTRODUCCIÓN

La onda de propagación en medios difusos, como propagación de luz en materia biológica se ha descrito a través de dos teorías, la teoría analítica y la teoría de transporte. La teoría analítica es esencialmente con ecuaciones, como las ecuaciones de Maxwell y ecuaciones aproximadas para describir la onda de propagación en medios difusos (Ishimaru, 1978). En la práctica, la aproximación exacta al modelo analítico es de gran dificultad y a veces imposible de obtener. Por lo tanto, la teoría de transporte por lo general es aplicada para describir el transporte de la radiación por un medio que dispersa partículas. En la teoría de transporte se investiga la conservación de la partícula y el fenómeno de onda de las partículas no es tomado en consideración.

En el proceso de difusión, las partículas sufren la dispersión por colisiones elásticas y la energía de las partículas es disminuida por la absorción y aumentada por las fuentes de radiación. La teoría de transporte satisfactoriamente ha sido aplicada en muchas áreas de la ciencia como el transporte de neutrón, la transferencia de partículas cargadas, la visibilidad submarina, la biología marítima, la óptica biomédica y la transferencia de radiación en atmósferas planetarias y estelares (Amaldi, 1959) (Cercignani., 1988) (Dorn, 2000) (Ishimaru, 1978) (Patterson, 1991) (Zweifel, 1967).

La teoría de transporte puede ser modelada y aproximada por métodos estocásticos, como Monte Carlo, métodos analíticos y deterministas que están basados en la descripción del transporte de partícula con ecuaciones diferenciales parciales. En los métodos estocásticos se modelan las interacciones de partículas individuales y se ve como ellas son dispersadas y absorbidas dentro del medio.

Los dos métodos estocásticos que han sido usados en la Tomografía Óptica son el método de Monte Carlo y la teoría de paseo arbitrario (Arridge, y otros, 1997). Monte Carlo es el método estocástico que más a menudo es usado en Tomografía Óptica y por lo general es escogido como el método de referencia al cual otras soluciones son comparadas.

En la aproximación determinista, el transporte de partícula es descrito con ecuaciones diferenciales parciales, que pueden ser solucionadas de forma numérica o analítica (Arridge, y otros, 1997).

La ecuación básica de la teoría de transporte es la ecuación de transporte que es una versión lineal de la ecuación de Boltzmann que puede ser extraída de la teoría cinética de gas (Zweifel, 1967) o simplemente investigando la conservación de partícula dentro de un elemento de pequeño volumen de espacio de fase (Chandrasekhar, 1950) (Dorn, 2000).

En la Tomografía Óptica, un modelo ampliamente aceptado para la propagación de la luz en los tejidos es la ecuación de transferencia radiativa (ETR). Además el problema inverso de la ETR se ha utilizado en la reconstrucción de imágenes en tomografía óptica (Dorn, 2000) (Hielscher, y otros, 1998) (Kim, 2004).

2.2. ECUACIÓN DE TRANSFERENCIA RADIATIVA Y APROXIMACIÓN DE LA DIFUSIÓN

Soluciones analíticas para la ecuación de transporte están disponibles sólo para unas simples geometrías y las soluciones numéricas a menudo conducen a enormes problemas computacionales. Así, en muchos casos, la ecuación de transporte es demasiado compleja para ser utilizada en aplicaciones prácticas, por lo cual se aplican aproximaciones de los modelos.

En medios de gran dispersión, tales como tejidos biológicos, la ecuación de transporte es aproximada por la teoría de difusión (Tarvainen, 2006). A continuación se explica los componentes y términos de la ecuación de transferencia radiativa para luego mostrar cómo se obtiene la ecuación de aproximación de la difusión.

2.2.1 Ecuación de transferencia radiativa

La ecuación de transferencia de la radiación (ETR) en el dominio temporal (2.1) es una ecuación diferencial hiperbólica de varias variables, la cual es una formulación matemática que modela el comportamiento de cómo se propaga la energía en forma de partículas subatómicas en un dominio Ω en el cual existen condiciones impuestas por el medio, específicamente la materia.

La ETR escrita en el dominio temporal es:

$$\frac{1}{c}\frac{\partial}{\partial t}\phi(r,\hat{s},t) + \hat{s}\cdot\nabla\phi(r,\hat{s},t) + (u_s(r)+u_a(r))\phi(r,\hat{s},t) = u_s(r)\int_{s_{n-1}}\Theta(\hat{s}\cdot\hat{s}')\phi(r,\hat{s}',t)d\hat{s}' + q(r,\hat{s},t) \quad (2.1)$$

La ETR es una aproximación de la ecuación de transporte y es obtenida mediante la integración de la ecuación de transporte de la energía y mediante la aproximación de los parámetros de la energía y un promedio de los parámetros. Por lo tanto, se asume en la ETR que la energía de las partículas no cambia en la colisión y que el índice de refracción es constante en el medio (Firbank, y otros, 1996) (Khan, y otros, 2003). En la ecuación (2.1) c es la velocidad de luz en el medio, $u_s(r)$ y $u_a(r)$ son los coeficientes de dispersión y de absorción monocromáticos del medio respectivamente, $\phi(r,\hat{s},t)$ es la radiancia, que es la tasa de partículas que se propagan dentro de un ángulo solido en dirección $\hat{s}$ en un instante t, $\Theta(\hat{s}\cdot\hat{s}')$ es la función de fase de la dispersión y $q(r,\hat{s},t)$ es la fuente de fotones y r es un lugar dentro del dominio Ω. En el dominio $\Omega \subset \Re^n$ ($n = 2$ o 3 indican el dominio físico que se considera isotrópico), la probabilidad de dispersión entre dos direcciones depende sólo del ángulo relativo entre esas direcciones, no en un sentido absoluto. Además si $\partial\Omega$ denota la frontera del dominio y $\hat{s} \in s^{n-1}$ denota un vector unitario en la dirección de interés.

El coeficiente de dispersión se mide en el sistema internacional de medida en m^{-1}, su inverso $l_s = 1/u_s$ representa la distancia media que un fotón recorrerá antes de ser esparcido hacia una dirección. El coeficiente de absorción se mide en m^{-1}, su inverso $l_a = 1/u_a$ representa la distancia promedio que los fotones recorrerán dentro de un medio absorbente antes de ser completamente absorbidos.

La función de fase de la dispersión $\Theta(\hat{s}\cdot\hat{s}')$ describe la probabilidad que un fotón con una dirección inicial $\hat{s}'$ tenga una dirección $\hat{s}$ después de un evento de dispersión. Puede ser considerada como una función de distribución de probabilidad y así es una función no negativa que satisface:

$$\int_{S_{n-1}} \Theta(\hat{s}\cdot\hat{s}')d\hat{s} = \int_{S_{n-1}} \Theta(\hat{s}\cdot\hat{s}')d\hat{s}' = 1 \tag{2.2}$$

En Tomografía Óptica, la función de fase más usada es la función de dispersión de Henyey – Greenstein (2.3) que en el caso tridimensional es de la forma:

$$\Theta(\hat{s}\cdot\hat{s}') = \frac{1}{4\pi}\frac{1-g^2}{(1+g^2-2g\hat{s}\cdot\hat{s}')^{3/2}} \tag{2.3}$$

y en el caso bidimensional es:

$$\Theta(\hat{s}\cdot\hat{s}') = \frac{1}{2\pi}\frac{1-g^2}{(1+g^2-2g\hat{s}\cdot\hat{s}')} \tag{2.4}$$

El parámetro de dispersión g define una probabilidad de densidad asociada a la dispersión, con valores entre -1 $< g <$ 1. Cuando $g = 0$, la probabilidad de dispersión es una distribución uniforme. Cuando $g > 0$ el modelo favorece la dispersión hacia adelante y cuando $g < 0$ favorece hacia atrás. Aunque la función de fase de Henyey - Greenstein es ampliamente la función de fase adoptada en ópticas biomédicas, muestra discrepancias con la teoría de Mie que da una solución exacta de la dispersión de una onda electromagnética plana por una esfera isotrópica y homogénea (Tarvainen, 2006).

La ETR es básicamente una ecuación de equilibrio de cambio en la radiancia. Si consideramos un pequeño elemento en el espacio fase, que es un pequeño volumen dV en torno a una posición r y un pequeño ángulo sólido en todo sentido $\hat{s}$ en el momento t, los componentes de la ecuación en su formulación dominio-tiempo (2.1) puede interpretarse de la siguiente manera.

El primer elemento a la izquierda es la derivada del tiempo de la luminosidad, que es igual a la variación de la luminosidad, es decir, el número de fotones que entran menos el número de fotones que salen. El segundo término representa el flujo de fotones a lo largo de la dirección $\hat{s}$, es decir, la pérdida de fotones a través de las superficies del volumen dV. El tercer término representa la pérdida de fotones debido a la absorción y la pérdida de fotones debido a la dispersión de la dirección ŝ en otras direcciones. En el lado derecho de la ecuación (2.1), el primer término representa la

ganancia de fotones debido a la dispersión de otras direcciones hacia la dirección $\hat{s}$, y el segundo término representa la ganancia de fotones a través de las fuentes de radiación.

2.2.1.1 Condiciones de la Frontera para la ETR

Con el fin de obtener una solución única para la ETR, la distribución de la radiancia en la frontera $\partial\Omega$, es decir, $\phi(r, \hat{s}, t)$ para $\hat{s} \cdot \hat{n} < 0$, donde $\hat{n}$ es la normal unitaria hacia el exterior, debe ser conocida. Se puede aplicar varias condiciones de fronteras a la ETR, y por lo general para resolver el problema de difusión se utiliza la condición de borde que asume que ningún fotón viaje en una dirección interior al borde $\partial\Omega$ del dominio Ω, es decir:

$$\phi(r, \hat{s}, t) = 0, \;\; r \in \partial\Omega, \; \hat{s} \cdot \hat{n} < 0 \tag{2.5}$$

Esta condición de borde, también conocida como la condición de borde de superficie libre y la condición de borde de vacío, implica que una vez que un fotón escapa el dominio Ω no vuelve a entrar.

2.2.2 Aproximación de la Difusión

La propagación de luz en tejidos es normalmente modelada con la aproximación de la difusión (AD). El método más típico para derivar la AD de la ETR es expandir la radiancia, el término de la fuente y la función fase en serie harmónica esférica truncada en el enésimo momento. A esta aproximación se le conoce como aproximación P_N. Una alternativa a la P_N es la aproximación de Boltzmann jerárquico que se acerca a los momentos de radiancia y se usan para formar un juego de ecuaciones acopladas, el cual es una aproximación de la ETR. Además, la AD puede derivarse usando técnicas asintóticas o usando el álgebra de la proyección. Aquí en la derivación de la AD, la aproximación del P_1 es derivada y se obtiene la ecuación de difusión (2.6). La derivación completa de la aproximación P_1 se encuentra en (Arridge, 1999) (Tarvainen, 2006) .

$$-\nabla \cdot k(r)\nabla\Phi(r,t) + u_a(r)\Phi(r,t) \; + \; \frac{1}{c}\frac{\partial\Phi(r,t)}{\partial t} = q_0(r,t) \tag{2.6}$$

En la ecuación 2.6 $k(r) = (3(u_a(r) + u_s'(r)))^{-1}$ y es el coeficiente de difusión, r es una posición en el dominio Ω, donde $u_a(r)$ es el coeficiente de absorción en la posición, $u_s'(r) = u_s(r)(1 - g_1)$ es el coeficiente de dispersión reducido donde g_1 es la función de fase $\int_{S^{n-1}}(\hat{s} \cdot \hat{s}')\Theta(\hat{s} \cdot \hat{s}')d\hat{s}$, c la velocidad de la luz del medio, t es un instante en el tiempo, $q_0(r,t)$ es la densidad de fotones de luz en la fuente ubicada en r y $\Phi(r,t)$ es la densidad de fotones de luz en r.

2.2.2.1 Condiciones de frontera para AD

La aproximación de la difusión no puede cumplir la condición de frontera (2.5). En cambio hay algunas condiciones de frontera que se pueden utilizar en la AD. La más simple es la condición de frontera Dirichlet, condición de frontera que es también referida como la condición de frontera cero, que establece que la densidad de fotones es cero en la frontera del dominio, es decir $\Phi(r) = 0,\ r \in \partial\Omega$. La condición de frontera más utilizada en Tomografía Óptica es la condición de frontera de Robín, y se puede derivar de la siguiente manera.

$$\Gamma(r) = -k(r)\frac{\partial\Phi(r)}{\partial\vec{n}} = \frac{2\gamma_n}{A} \tag{2.7}$$

donde γ_n es una constante que depende de la dimensión y tiene valores $\gamma_2 = 1/\pi$ y $\gamma_3 = 1/4$, donde $A = (1 + R)/(1 - R)$, con $A = 1$ en el caso de no existir reflexión de la superficie y R es el coeficiente de reflexión en la frontera $\partial\Omega$, con $0 \leq R \leq 1$. Por lo tanto, si $R = 0$ no se produce la reflexión en la frontera. La derivación detallada de (2.7) está disponible en el anexo C.

2.2.2.2 Modelos de la Fuente para la aproximación de la difusión

En la construcción de la AD, las fuentes de fotones de luz son normalmente modeladas por dos modelos de aproximación: el modelo de fuente colimado y el modelo de fuente difuso. En este trabajo sólo usaremos el caso de fuente colimada, la cual es modelada como una fuente de punto isotrópico.

$$q_0(r) = \delta(r - r_s) \tag{2.8}$$

donde la posición r_s está localizada a una profundidad $1/u_s'$ de la fuente.

2.2.2.3 Validez de la aproximación de la difusión (AD)

La condición básica para la validez de la AD es que la distribución angular de la radiancia sea casi uniforme. A fin de lograr esto, el medio de dispersión debe ser el factor que domina el problema, por lo tanto $u_a \ll u_s$. La mayoría de los tipos de tejidos tienen mucha dispersión y la AD puede considerarse una buena aproximación para la modelización de la propagación de la luz en tejidos biológicos.

La AD describe la propagación de la luz con buena exactitud en situaciones en las que sus hipótesis son válidas y se ha aplicado con éxito en muchas aplicaciones de Tomografía Óptica. Sin embargo, la condición que declara que la distribución angular de la radiancia debe ser casi uniforme se viola cerca de las fuentes colimadas que se usan en Tomografía Óptica. Además, la condición no se cumple en un medio muy absorbente de baja dispersión, como la del fluido cerebroespinal que rodea al cerebro y los ventrículos del cerebro, por nombrar algunos. Por otra parte, la AD no acoge a condiciones realistas de frontera o discontinuidades.

Cuando se estiman las densidades cerca de las fuentes, las aproximaciones no son válidas o bien tienen un margen de error mayor que en otros lugares del dominio utilizado (Alexandrakis, y otros, 1998) (S. Fantini, 1997). También se ha demostrado que estas limitaciones en la teoría difusión, puede llevar a grandes errores en la imagen reconstruida (Dehghani, y otros, 2000).

A pesar de estas limitancias, que pueden ser mejoradas con métodos híbridos de de aproximación ETR y AD (Tarvainen, 2006), lo presentado en este capítulo es suficiente para abordar el problema de la difusión computacionalmente, y el desarrollo o búsqueda de métodos óptimos en este sentido deberían ser estudiados por especialistas físicos del problema. Gran parte de la información necesaria para un desarrollo más acabado se encuentra disponible en (Arridge, y otros, 1997) (Arridge, 1999) (Tarvainen, 2006).

3 ELEMENTOS FINITOS

El método de los elementos finitos es un método numérico que permite encontrar soluciones aproximadas a ecuaciones diferenciales parciales. Las aplicaciones actuales del método son muy extensas e incluyen sistemas lineales y no lineales, estáticos, dinámicos tales como Mecánica de Sólidos, Teoría de la Elasticidad, Mecánica de Fluidos, Transmisión de Calor y Electromagnetismo. A continuación se menciona los aspectos fundamentales del método de elementos finitos de forma general, y de forma específica la formulación variacional y discretización de un problema con elementos finitos triangulares de tres nodos, para la solución de ecuaciones diferenciales elípticas.

3.1 GENERALIDADES SOBRE EL MÉTODO DE ELEMENTOS FINITOS

El método de elementos finitos se basa en dividir el cuerpo, estructura o dominio (medio continuo) en una serie de subdominios no intersectantes entre sí denominados *elementos finitos* sobre el que están definidas ciertas ecuaciones integrales que caracterizan el comportamiento físico del problema. El conjunto de elementos finitos forma una partición del dominio también denominada discretización. Dentro de cada elemento se distingue una serie de puntos representativos llamados *nodos*. El conjunto de nodos considerando sus relaciones de adyacencia se llama *malla*.

De acuerdo con relaciones de adyacencia o conectividad, en la unión de las aristas (nodos) que componen la malla, se relaciona el valor de un conjunto de variables incógnitas definidas en cada nodo y a esto se les denomina grados de libertad. El conjunto de relaciones entre el valor de una determinada variable entre los nodos se puede escribir en forma de sistema de ecuaciones lineales, donde la matriz de dicho sistema de ecuaciones se llama matriz de rigidez del sistema y el número de ecuaciones de dicho sistema es proporcional al número de nodos. El método de los elementos finitos es muy usado debido a su generalidad y a la facilidad de introducir dominios de cálculo complejos (en dos o tres dimensiones). Además el método es fácilmente adaptable a problemas de difusión de calor, mecánica de fluidos y electromagnetismo por nombrar algunos.

Otro aspecto muy relevante del método es la convergencia, en que si se consideran particiones de elementos finitos sucesivamente más finas, la solución

numérica calculada converge rápidamente hacia la solución exacta del sistema de ecuaciones.

3.1.1 Procedimiento general

En primer lugar se descompone todo el dominio Ω en M subdominios Ω_e o elementos e que tienen forma geométrica, como triángulos, tetraedros, exaedros, prismas, etc., de modo que la unión de todos los elementos aproxime al dominio original o incluso coincida con él.

Supongamos que se descompone el dominio Ω en elementos e con forma de triángulos (Figura 3.1). Dichos elementos tienen unos puntos característicos situados en los vértices, puntos intermedios de las aristas e incluso en el interior denominados nodos. Los elementos adyacentes se conectan por los nodos situados en la arista o vértice común.

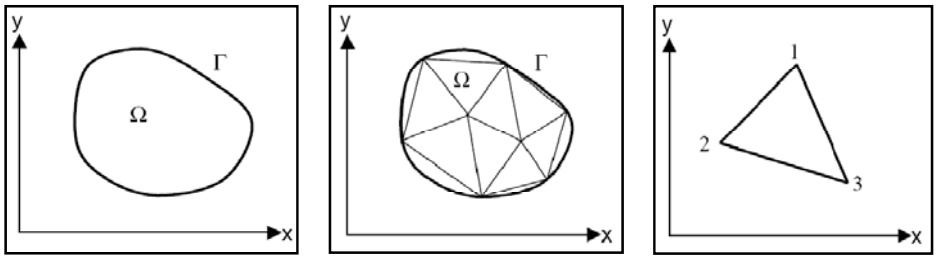


Figura 3.1.- *Método de los elementos finitos: a) Dominio Ω; b) subdivisión del dominio Ω en elementos triangulares; c) elemento triangular de tres nodos.*

De esta manera se tiene un conjunto de nodos para todo el dominio. Los nodos se numeran de 1 a *N*. Esta es la numeración global de todos los nodos del dominio (Figura 3.2.a). Por otra parte los nodos de cada elemento tienen una numeración local (Figura 3.2.b) habiendo para cada elemento una correspondencia entre la numeración local y la global. En cada elemento puede establecerse una relación entre los desplazamientos en los nodos y las acciones aplicadas en ellos.

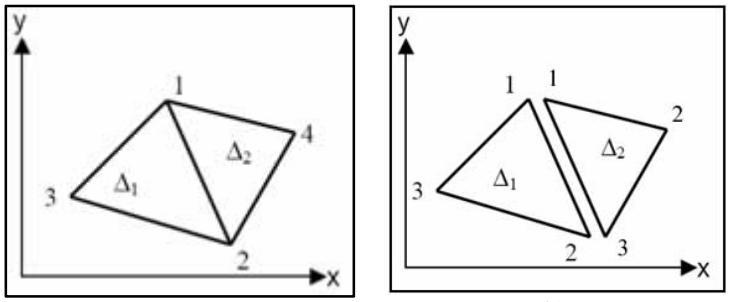


Figura 3.2.- *Método de elementos finitos: a) vértices con notación global; b) notación local de los vértices*

Dicha relación se denomina ecuación de equilibrio local o ecuación de equilibrio del elemento, la cual en forma matricial mediante la matriz de rigidez del elemento k^e, vector de desplazamientos nodales del elemento u^e, y cargas nodales del elemento f^e, puede expresarse como:

$$\begin{bmatrix} k_{11}^e & k_{12}^e & k_{13}^e \\ k_{21}^e & k_{22}^e & k_{23}^e \\ k_{31}^e & k_{32}^e & k_{33}^e \end{bmatrix} \begin{bmatrix} u_1^e \\ u_2^e \\ u_3^e \end{bmatrix} = \begin{bmatrix} f_1^e \\ f_2^e \\ f_3^e \end{bmatrix} \tag{3.1}$$

La ecuación matricial (3.1) se determina mediante técnicas numéricas de interpolación e integración aplicadas a las ecuaciones o sistema de ecuaciones diferenciales del problema expresadas en forma débil. Lo importante ahora es observar que la ecuación de equilibrio relaciona desplazamientos en un conjunto discreto de puntos: los nodos, con acciones aplicadas en dichos nodos, es decir con acciones puntuales. Como contraste, la ecuación o sistema de ecuaciones diferenciales relaciona desplazamientos en todos los puntos del dominio.

El siguiente paso consiste en construir la matriz de rigidez del sistema completo K a partir de las matrices de rigidez de cada elemento $k^e, e = 1,2 \dots, N$. A este proceso se le conoce como proceso de ensamblado de los elementos, y debe considerar la continuidad de la solución y balance de flujos internos, basados en consideraciones físicas del problema. La ecuación global expresada en forma matricial es:

$$K\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix} \tag{3.2}$$

Ahora si por ejemplo establecemos las relaciones de los elementos en la ecuación de equilibrio global utilizando el ejemplo de la Figura 3.2, el sistema de matrices queda como:

$$\begin{bmatrix} k_{11}^1 + k_{11}^2 & k_{12}^1 + k_{13}^2 & k_{13}^1 & k_{14}^2 \\ k_{21}^1 + k_{31}^2 & k_{22}^1 + k_{33}^2 & k_{23}^1 & k_{24}^2 \\ k_{31}^1 & k_{32}^1 & k_{33}^1 & 0 \\ k_{41}^2 & k_{42}^2 & 0 & k_{44}^e \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} f_1^1 + f_1^2 \\ f_2^1 + f_3^2 \\ f_3^1 \\ f_2^2 \end{bmatrix} \tag{3.3}$$

Dentro de cada elemento, se asume que la solución se puede aproximar adecuadamente por una función lineal como:

$$u(x, y) = a + bx + cy \tag{3.4}$$

De esta forma la solución verdadera se reemplaza por una función lineal por tramos, que debe ser continua en todo el dominio. En cada uno de los vértices de un elemento (Figura 3.1.c), la solución puede expresarse en forma matricial como:

$$\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \begin{Bmatrix} a \\ b \\ c \end{Bmatrix} \tag{3.5}$$

Para cada elemento del dominio, los coeficientes a, b y c pueden ser determinados resolviendo el sistema anterior. Entonces, la función (3.4) puede escribirse como:

$$u = [1 \;\; x \;\; y\,] \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix}^{-1} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} \tag{3.6}$$

$$u = \sum_{i=1}^{3} u_i \psi_i(x, y) \tag{3.7}$$

donde la función ψ_i es:

$$\psi_i(x, y) = \frac{1}{2A}[\alpha_i + \beta_i x + \gamma_i y] \tag{3.8}$$

donde

$$\alpha_i = x_j y_k - x_k y_j, \qquad \beta_i = (y_j - y_k), \quad \gamma_i = (x_k - x_j) \tag{3.9}$$

Siendo cíclicos los índices i, j y k. El valor A representa el área del elemento:

$$A = \frac{1}{2}\begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} \tag{3.10}$$

Las funciones ψ_i anteriores tienen la propiedad de valer uno en uno de los vértices y cero en los otros, es decir:

$$\psi_i\,(x_j, y_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{3.11}$$

Hasta ahora sólo hemos mostrado la construcción de la ecuación de equilibrio global sin imponer las condiciones de contorno del dominio discretizado. Para ello debemos comprender algunos conceptos de las formulaciones variacionales los cuales explicaremos en secciones posteriores, pero una vez impuestas las condiciones de contorno, se agrega al sistema de matrices la información sólo en las posiciones donde existen bordes en el dominio, y el sistema global queda como:

$$K \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix} + \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{bmatrix} \tag{3.12}$$

Donde si $q_i \in$ al borde Γ del dominio Ω se realizan los aportes al vector de cargas f en caso contrario el aporte es nulo.

Aunque toda la exposición realizada hasta ahora ha sido muy esquemática se ha pretendido destacar los siguientes aspectos del método de elementos finitos:

- El dominio se divide en subdominios o elementos finitos.
- Cada elemento tiene su ecuación de equilibrio.
- Hay un proceso de ensamblado con el que se determina la ecuación de equilibrio global.
- Se imponen las condiciones de contorno y se determina el sistema reducido que contiene únicamente los desplazamientos incógnitas.
- Con todos los desplazamientos conocidos se calculan las acciones desconocidas o reacciones.
- En cada elemento a partir de su ecuación de equilibrio se puede determinar el estado del elemento: desplazamientos, deformaciones y tensiones en cualquier punto del elemento.

Una vez generada la matriz y el vector elemental que componen el sistema, lo que queda por resolver es un sistema de ecuaciones lineal es del tipo $Ax = b$, donde A es por lo general una matriz simétrica. En la siguiente sección veremos una derivación formal de la formulación variacional, donde se mostrará la base matemática sobre la cual está sustentada la aproximación que se realiza con el método de elementos finitos.

3.2 FORMULACIÓN VARIACIONAL DE ELEMENTOS FINITOS

A continuación algunos conceptos básicos de la formulación variacional del método de elementos finitos que pueden aplicarse a una gran variedad de problemas. En primer lugar describiremos algunos conceptos sobre métodos aproximados de solución para ecuaciones diferenciales, en particular veremos el método de residuos ponderados.

3.2.1 Métodos aproximados de solución

Describiremos los métodos más usuales empleados para la resolución aproximada de ecuaciones diferenciales. En general los métodos empleados son de dos tipos, por un

lado están los métodos basados en principios variacionales, generalmente asociados a la minimización de algún funcional (Ver anexo B), y por otro lado tenemos los métodos del tipo de residuos ponderados que se aplican directamente sobre la ecuación diferencial y sus condiciones de contorno, no precisando de ningún funcional asociado.

3.2.1.1 Métodos de Residuos Ponderados

Con estos métodos es posible obtener soluciones aproximadas de ecuaciones diferenciales que no tienen un funcional asociado. Consideremos una ecuación diferencial definida sobre un dominio Ω como:

$$A(u) - f = 0 \quad \text{en} \quad \Omega \tag{3.13}$$

y estando sujeta a condiciones naturales de contorno sobre la parte Γ_N de su frontera de la forma:

$$M(u) - g = 0 \quad \text{en} \quad \Gamma_N \tag{3.14}$$

siendo $A(u)$, $M(u)$ operadores diferenciales. Algunos ejemplos de los operadores diferenciales $A(u)$ son:

$$A(u) = -\frac{d}{dx}\left(a\frac{du}{dx}\right) + cu \tag{3.15}$$

$$A(u) = -\frac{d^2}{dx^2}\left(b\frac{d^2u}{dx^2}\right) \tag{3.16}$$

$$A(u) = -\left[\frac{\partial}{\partial x}\left(k_x\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y\frac{\partial}{\partial y}\right)\right] \tag{3.17}$$

Si reemplazamos la solución exacta u por una solución aproximada $\tilde{u}$ en la ecuación diferencial (3.13) y en sus condiciones naturales de contorno (3.14), éstas no serán satisfechas exactamente, generando un residuo R_Ω en el dominio y un residuo R_Γ en el contorno, esto es:

$$R_\Omega(\tilde{u}) = A(\tilde{u}) - f \neq 0 \quad \text{en} \quad \Omega \tag{3.18}$$

$$R_\Gamma(\tilde{u}) = M(\tilde{u}) - g \neq 0 \quad \text{en} \quad \Gamma_N \tag{3.19}$$

La idea básica del método de residuos ponderados es imponer la condición:

$$\int_\Omega W R_\Omega(u) d\Omega + \int_{\Gamma_N} \bar{W} R_\Gamma(u) d\Gamma = 0 \tag{3.20}$$

Para cualquier par de funciones arbitrarias W y $\bar{W}$ que sean integrables y no idénticamente nulas. Se puede demostrar que si esto se cumple entonces la función u debe ser la solución exacta de la ecuación diferencial y de sus condiciones de contorno naturales. La solución aproximada $\tilde{u}$ al igual que el método de Rayleigh – Ritz (ver anexo B), es una combinación lineal de funciones de prueba ϕ_i, tal que:

$$\tilde{u} = \sum_{i=1}^{n} c_i \phi_i(x, y) \tag{3.21}$$

donde las funciones de prueba $\phi_i(x, y)$ satisfacen las condiciones esenciales de contorno y las n constantes c_i son coeficientes a determinar imponiendo las n condiciones:

$$\int_\Omega W_i R_\Omega(\tilde{u}) d\Omega + \int_{\Gamma_N} \bar{W}_i R_\Gamma(\tilde{u}) d\Gamma = 0, \quad i = 1,2,\dots,n \tag{3.22}$$

donde las funciones W_i y $\bar{W}_i$ son llamadas *funciones de ponderación*.

Dependiendo del tipo de funciones de ponderación empleadas tenemos diferentes métodos de aproximación. En este trabajo, usaremos funciones de ponderación idénticas a las funciones de prueba. A este método se le llama método de Galerkin.

a) Método de Galerkin: en este método se adoptan las funciones de ponderación iguales a las funciones de prueba resultando:

$$\int_{\Omega} \phi_i R_{\Omega}(\tilde{u}) d\Omega + \int_{\Gamma_N} \phi_i R_{\Gamma}(\tilde{u}) d\Gamma = 0, \quad i = 1,2,\dots,n \tag{3.23}$$

Nótese que las funciones de prueba ϕ_i deben tener derivadas definidas hasta el máximo orden que aparece en el residuo, esto es, del mismo orden que la ecuación diferencial.
Como se puede observar la formulación variacional y sus métodos derivados son una etapa fundamental al momento de resolver una ecuación diferencial con elementos finitos. Además el modelo es adaptable a cualquier tipo de geometría a utilizar ya sea para una, dos o tres dimensiones y a cualquier tipo de elementos finitos que se utilicen (triángulos, cuadriláteros, tetraedros, etc.).

Lo presentado en este capítulo es la base sobre la cual realizaremos la formulación variacional de la ecuación de difusión de fotones de luz. Mayor información sobre el cálculo de las integrales elementales y cálculos geométricos de los elementos triangulares de tres nodos, se encuentra en el anexo A o bien en la bibliografía relacionada con elementos finitos (Reddy, 1993).

4 APROXIMACIÓN DE LA DIFUSIÓN CON ELEMENTOS FINITOS

4.1 INTRODUCCIÓN

La ecuación de aproximación de la difusión de fotones de luz en un medio turbio es:

$$-\nabla \cdot k(r)\nabla\Phi(r) + u_a(\mathrm{r})\Phi(r) + \frac{1}{c}\frac{\partial\Phi(r)}{\partial \mathrm{t}} = q_0(r) \tag{4.1}$$

A continuación se utiliza el método de elementos finitos para resolver la ecuación diferencial (4.1).

4.2 FORMULACIÓN VARIACIONAL DE LA APROXIMACIÓN DE LA DIFUSIÓN.

Para obtener la formulación variacional de la AD, se utiliza la formulación Galerkin (3.2.1.d). Como primer paso se multiplica la ecuación (4.1) por una función de prueba $\Psi(r)$ y se integra sobre el dominio Ω, dando como resultado la siguiente ecuación:

$$\int_\Omega -\nabla \cdot k(r)\nabla\Phi(r)\Psi(r)d\Omega + \int_\Omega u_a(\mathrm{r})\Phi(r)\Psi(r)d\Omega + \int_\Omega \frac{1}{c}\frac{\partial\Phi(r)}{\partial \mathrm{t}}\Psi(r)d\Omega = \int_\Omega q_0(r)\Psi(r)d\Omega \tag{4.2}$$

Utilizando la fórmula de Green que nos permite integrar por partes el primer miembro, podemos escribir la primera integral de la ecuación (4.2) como:

$$\int_\Omega -\nabla \cdot k(r)\nabla\Phi(r)\Psi(r)d\Omega = \int_\Omega k(r)\nabla\Phi(r)\cdot\nabla\Psi(r)d\Omega - \int_{\partial\Omega} k(r)\big(\vec{n}(r)\cdot\nabla\Phi(r)\big)\Psi(r)d(\partial\Omega) \tag{4.3}$$

donde $\vec{n}(r)$ es la normal en un punto $r \in \partial\Omega$. Entonces la ecuación (4.2) queda como:

$$\int_{\Omega} k(r)\nabla\Phi(r)\cdot\nabla\Psi(r)d\Omega \; + \int_{\partial\Omega}\Gamma(r)\Psi(r)d(\partial\Omega) + \int_{\Omega} u_a(\mathrm{r})\Phi(r)\Psi(r)d\Omega + \int_{\Omega}\frac{1}{c}\frac{\partial\Phi(r)}{\partial \mathrm{t}}\Psi(r)d\Omega = \int_{\Omega} q_0(r)\Psi(r)d\Omega \tag{4.4}$$

donde $\Gamma(r) = -k(r)\big(\vec{n}(r)\cdot\nabla\Phi(r)\big) = \frac{2\gamma_n}{A}\Phi(r)$, con $\gamma_n = \frac{1}{\pi}$ para un dominio de dos dimensiones o $\gamma_n = \frac{1}{4}$ en el caso de un dominio de tres dimensiones y A es el factor de reflexión en el borde. La ecuación (4.4) es la formulación variacional de la ecuación (4.1).

4.3 APROXIMACIÓN CON ELEMENTOS FINITOS

4.3.1 Aproximación lineal

Para obtener la aproximación de elementos finitos de la AD, la solución $\Phi(r)$ de la formulación variational (4.4) se aproxima a una base lineal de la forma:

$$\Phi(\mathrm{r}) \approx \Phi^h(\mathrm{r}) = \sum_{\mathrm{j}=1}^{N} \Phi_j\,\psi_j(\mathrm{r}) \tag{4.5}$$

donde $\psi_j(\mathrm{r})$ es la función de base nodal, Φ_j es la densidad de fotones en el punto nodal j y N es el número de puntos nodales.

Ahora, insertando la aproximación (4.5) en la formulación variacional (4.4), dejando la función de prueba $\Psi(r)$ como función $\psi_i(\mathrm{r})$, asumiendo que el problema está formulado en un dominio de dos dimensiones y no hay reflexión en el borde del dominio, la aproximación de elementos finitos de la AD es:

$$\int_{\Omega} k(r)\nabla \sum_{\mathrm{j}=1}^{N} \Phi_j\, \psi_j(\mathrm{r}) \cdot \nabla\psi_i(\mathrm{r})d\Omega \; + \int_{\partial\Omega} \frac{2\gamma_n}{A} \sum_{\mathrm{j}=1}^{N} \Phi_j\, \psi_j(\mathrm{r})\psi_i(\mathrm{r})d(\partial\Omega)$$

$$+ \int_{\Omega} u_a(r) \sum_{\mathrm{j}=1}^{N} \Phi_j\, \psi_j(\mathrm{r})\psi_i(\mathrm{r})d\Omega + \int_{\Omega} \frac{1}{c}\frac{\partial}{\partial \mathrm{t}} \sum_{\mathrm{j}=1}^{N} \Phi_j\, \psi_j(\mathrm{r})\psi_i(\mathrm{r})d\Omega \qquad (4.6)$$

$$= \int_{\Omega} q_0(r)\psi_i(\mathrm{r})d\Omega$$

Sacando la sumatoria fuera de la integral y separando nuestro valor incógnita $\Phi_j(r)$ la ecuación (4.6) queda como:

$$\sum_{\mathrm{j}=1}^{N} \left(\int_{\Omega} k(r)\nabla\psi_j(\mathrm{r}) \cdot \nabla\psi_i(\mathrm{r})d\Omega \right) \Phi_j \; + \sum_{\mathrm{j}=1}^{N} \left(\int_{\partial\Omega} \frac{2\gamma_n}{A} \psi_j(\mathrm{r})\psi_i(\mathrm{r})d(\partial\Omega) \right) \Phi_j$$

$$+ \sum_{\mathrm{j}=1}^{N} \left(\int_{\Omega} u_a(r)\psi_j(\mathrm{r})\psi_i(\mathrm{r})d\Omega \right) \Phi_j \qquad (4.7)$$

$$+ \sum_{\mathrm{j}=1}^{N} \left(\int_{\Omega} \frac{1}{c}\psi_j(\mathrm{r})\psi_i(\mathrm{r})d\Omega \right) \frac{\partial}{\partial \mathrm{t}} \Phi_j = \int_{\Omega} q_0(r)\psi_i(\mathrm{r})d\Omega$$

Esta ecuación puede expresarse en forma matricial de la siguiente forma:

$$k_{ij} = \int_{\Omega} k(r)\nabla\psi_j(\mathrm{r}) \cdot \nabla\psi_i(\mathrm{r})d\Omega \qquad (4.8)$$

$$b_{ij} = \int_{\partial\Omega} \frac{2\gamma_n}{A} \psi_j(\mathrm{r})\psi_i(\mathrm{r})d(\partial\Omega) \qquad (4.9)$$

$$c_{ij} = \int_{\Omega} u_a(r)\psi_j(\mathrm{r})\psi_i(\mathrm{r})d\Omega \qquad (4.10)$$

$$m_{ij} = \int_{\Omega} \frac{1}{c}\psi_j(\mathrm{r})\psi_i(\mathrm{r})d\Omega \qquad (4.11)$$

$$q_i = \int_{\Omega} q_0(r)\psi_i(\mathrm{r})d\Omega \qquad (4.12)$$

Reemplazando en (4.7)

$$\sum_{\mathrm{j}=1}^{N} (k_{ij} + b_{ij} + c_{ij})\Phi_j + \sum_{\mathrm{j}=1}^{N} (m_{ij}) \frac{\partial}{\partial \mathrm{t}} \Phi_j = q_i \qquad (4.13)$$

Finalmente obtenemos:

$$(\mathrm{K}+\mathrm{B}+\mathrm{C})\Phi+\ \mathrm{M}\frac{\partial}{\partial \mathrm{t}}\Phi=\mathrm{Q} \tag{4.14}$$

En donde $\Phi=[\Phi_1\ \Phi_2\ \ldots\Phi_\mathrm{N}]^t$

Ahora se deben calcular las integrales elementales (4.8 a 4.12) y una vez completada la matriz de rigidez y el vector de cargas con los aportes elementales se procede a resolver el sistema de ecuaciones. En la siguiente sección (sección 4.3.2) se realiza un análisis previo de las condiciones del problema y se muestra cómo se obtienen las formulas de los cálculos de cada integral.

4.3.2 Análisis previo

En caso que se desee resolver la ecuación 4.1 en estado estable, es decir la densidad de fotones en cada punto del dominio no cambia, se tiene que:

$$\frac{\partial\Phi(r)}{\partial \mathrm{t}}=0 \tag{4.15}$$

y luego el sistema es:

$$\mathrm{F}\Phi=\mathrm{Q} \tag{4.16}$$

donde $\mathrm{F}=\mathrm{K}+\mathrm{B}+\mathrm{C}$

Si se usa condición de borde del tipo Dirichlet $\Phi_i(r)=0$ tal que $r\in\partial\Omega$, lo cual indica que no hay fotones que se escapan o fluyen fuera del dominio Ω, entonces $\mathrm{B}=0$ y la cuantificación de los valores del borde se realiza en una frontera virtual situada en los nodos que anteceden al borde $\partial\Omega$.

La otra formulación de cálculo del borde es la basada en la condición de tipo Robín (2.26) vista en la sección 2.2.2.1. La cual consiste en asumir que los fotones no fluyen hacia el interior del dominio Ω y se puede obtener valores cuantificables en el

borde del dominio. Sin importar el tipo de geometría utilizado en la partición del dominio, el aporte que recibe la matriz global tiene el mismo procedimiento de llenado, que consiste en calcular las integrales de los elementos locales y realizar el ensamblaje global según la correspondencia nodal global.

4.3.3 Calculo de aportes elementales a las matrices del sistema

El desarrollo del cálculo integral elemental que se presenta a continuación es en base a una geometría de dos dimensiones con elementos finitos triangulares lineales de tres nodos. Cada posición de la matriz del sistema recibe aportes según correspondencia nodal global, es decir supongamos que deseamos calcular todos los aportes a la matriz C, los valores de c_{ij} serán obtenidos a partir de las integrales elementales de todos los elementos e y cada elemento tiene asociada su propia matriz c_{ij}^{e} una vez calculada la integral elemental en todos los nodos del elemento e. Después del cálculo integral se asigna cada valor de c_{ij}^{e} a c_{ij} y este proceso de asignación es similar para cada matriz que compone el sistema de ecuaciones.

Para el cálculo de las integrales elementales se utiliza una formulación integral basada en la integral generalizada (4.17) que deriva de la formulación isoparamétrica de los elementos finitos triangulares de tres nodos. El detalle de la obtención de la integral generalizada (4.17) se encuentra en el Anexo A.

$$\int\int \hat{g}\, dA^{e} = 2A^{e} \int_{0}^{1} \int_{0}^{1-L_j^e} L^{e}\, dL_i^e\, dL_j^e \tag{4.17}$$

donde $\hat{g}$ es una función definida sobre el dominio del elemento e y L^e son funciones de interpolación expresadas en las coordenadas isopraramétricas del elemento e.

A continuación explicaremos cómo se obtienen a partir de la integral generalizada (4.17) las formulas que permiten calcular los aportes elementales $c_{ij}^{e}, m_{ij}^{e}, b_{ij}^{e}, k_{ij}^{e}$ y q_i^e necesarios para generar las matrices C, M, B, K y Q.

4.3.3.1 Calculo de c_{ij}^e y m_{ij}^e

De (4.10), (4.11) y la formulación (4.17) tenemos que $\hat{g}_c = u_a(r)\psi_j^e(r)\psi_i^e(r)$ y $\hat{g}_m = \frac{1}{c}\psi_j^e(r)\psi_i^e(r)$, donde $1/c$ y $u_a(r)$ son constantes, donde c_{ij}^e y m_{ij}^e expresadas en formulación isoparamétrica son equivalentes a:

$$c_{ij}^e = \int\int \hat{g}_c \, dA^e = u_a(r)2A^e \int_0^1 \int_0^{1-L_j^e} (L_j^e L_i^e) dL_i^e dL_j^e \tag{4.16}$$

$$m_{ij}^e = \int\int \hat{g}_m \, dA^e = \frac{2A^e}{c} \int_0^1 \int_0^{1-L_j^e} (L_j^e L_i^e) dL_i^e dL_j^e \tag{4.17}$$

Entonces tenemos dos casos posibles en función de los índices i, j ($i \neq j \; y \; i = j$) de las funciones de interpolación en coordenadas paramétricas. Como $u_a(r)2A^e$ y $2A^e/c$ son constantes, el cálculo integral es idéntico para (4.16) y (4.17) ya que sólo difiere en la constante que multiplica a la integral. A continuación se desarrolla la integral de ambos casos:

Caso 1: $i \neq j$

$$2A^e \int_0^1 \int_0^{1-L_j^e} (L_j^e \, L_i^e) dL_i^e dL_j^e \tag{4.18}$$

Al calcular la primera integral de (4.18) se obtiene:

$$2A^e \int_0^1 (L_j^e) \frac{\left(1 - L_j^e\right)^2}{2} dL_j^e = 2A^e \int_0^1 \frac{L_j^e - 2(L_j^e)^2 + (L_j^e)^3}{2} dL_j^e \tag{4.19}$$

a continuación se calcula (4.19):

$$2A^e \int_0^1 \frac{L_j^e - 2(L_j^e)^2 + (L_j^e)^3}{2} dL_j^e = \frac{A^e}{12} \tag{4.20}$$

Ahora agregando los términos constantes a cada integral éstas quedan como:

$$c_{ij}^{e} = u_a(r)\ \frac{A^e}{12} \tag{4.21}$$

$$m_{ij}^{e} = \frac{1}{c}\ \frac{A^e}{12} \tag{4.22}$$

Caso 2: $i = j$

$$2A^e \int_0^1 \int_0^{1-L_j^e} (L_i^e)^2 dL_i^e dL_j^e \tag{4.23}$$

al calcular la integral (4.23) se obtiene:

$$2A^e \int_0^1 \frac{\left(1 - L_j^e\right)^3}{3} dL_j^e = \frac{A^e}{6} \tag{4.24}$$

Agregando los términos constantes a cada integral éstas quedan como:

$$c_{ij}^{e} = u_a(r)\ \frac{A^e}{6} \tag{4.25}$$

$$m_{ij}^{e} = \frac{1}{c}\ \frac{A^e}{6} \tag{4.26}$$

generalizando el caso 1) y caso 2) para c_{ij} y m_{ij} éstas quedan como:

$$c_{ij}^{e} = u_a(r)\ \frac{A^e}{12}\left(1 + \delta_{ij}\right),\ m_{ij}^{e} = \frac{1}{c}\ \frac{A^e}{12}\left(1 + \delta_{ij}\right), \qquad \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{4.25}$$

Entonces para cualquier formulación similar a (4.18) se puede generalizar como:

$$2A^e \int_0^1 \int_0^{1-L_j^e} (L_j^e\, L_i^e) dL_i^e dL_j^e = \frac{A^e}{12}\left(1 + \delta_{ij}\right) \tag{4.26}$$

donde la función δ_{ij} es la delta de Kronecker.

4.3.3.2 Calculo de b_{ij}^{e}

En el borde, la integral se evalúa entre dos nodos $a,\ b$ que comparten una arista en $\partial\Omega$ y si $h = |b-a|$ es la distancia entre b y a, sabemos que:

$$\hat{\psi}_1(\bar{x}) = 1 - \frac{\bar{x}}{h} \tag{4.27}$$

$$\hat{\psi}_2(\bar{x}) = \frac{\bar{x}}{h} \tag{4.28}$$

donde $\hat{\psi}_1(\bar{x})$ y $\hat{\psi}_2(\bar{x})$ son las funciones de interpolación nodales que comparten una arista en el borde $\partial\Omega$, $\bar{x}$ es un valor en ε o η en coordenadas locales paramétricas (ver anexo A), la cuales dependen de cual contiene a a o b. Entonces las integrales del segmento h para todos los casos de combinación de funciones de interpolación elemental en el borde $\partial\Omega$ quedan como:

Caso 1: Solo existe una función $\hat{\psi}_i(\bar{x})$

$$\int_0^h \hat{\psi}_i(\bar{x})\, d\bar{x} = \frac{|b-a|}{2}, \qquad i = 1,2 \tag{4.29}$$

Caso 2: $i \neq j$

$$\int_0^h \hat{\psi}_i(\bar{x})\, \hat{\psi}_j(\bar{x}) d\bar{x} = \frac{|b-a|}{6}, \qquad i,j = 1,2;\ i \neq j \tag{4.30}$$

Caso 3: $i = j$

$$\int_0^h (\hat{\psi}_i(\bar{x}))^2\, d\bar{x} = \frac{|b-a|}{3}, \qquad i,j = 1,2;\ i \neq j \tag{4.31}$$

generalizando para el caso 2) y caso 3)

$$\int_0^h \hat{\psi}_i(\bar{x})\, \hat{\psi}_j(\bar{x}) d\bar{x} = \frac{|b-a|}{6}\left(1 + \delta_{ij}\right), \qquad \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{4.32}$$

finalmente agregando la constante $2\gamma_n/A$ el cálculo general de (4.9) es:

$$b_{ij}^e = \frac{2\gamma_n}{A}\frac{|b-a|}{6}(1+\delta_{ij}), \qquad \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{4.33}$$

4.3.3.3 Calculo de k_{ij}^e

Se sabe que si $F(x,y)$ es un campo escalar, su vector gradiente es:

$$\nabla F(x,y) = \left[\frac{\partial F(x,y)}{\partial x} \quad \frac{\partial F(x,y)}{\partial y}\right]^t \tag{4.34}$$

Sabemos que $\psi(x,y) = \frac{1}{2A}(\alpha + \beta x + \gamma y)$ entonces sus derivadas con respecto a x e y quedan como:

$$\frac{\partial \psi(x,y)}{\partial x} = \frac{\partial}{\partial x}\left(\frac{1}{2A}(\alpha + \beta x + \gamma y)\right) = \frac{\beta}{2A} \tag{4.35}$$

$$\frac{\partial \psi(x,y)}{\partial y} = \frac{\partial}{\partial y}\left(\frac{1}{2A}(\alpha + \beta x + \gamma y)\right) = \frac{\gamma}{2A} \tag{4.36}$$

Entonces si r es un punto en x, y

$$\nabla\psi_j(\mathrm{r}) \cdot \nabla\psi_i(\mathrm{r}) = \left[\frac{\beta_j}{2A} \quad \frac{\gamma_j}{2A}\right] \cdot \left[\frac{\beta_i}{2A} \quad \frac{\gamma_i}{2A}\right]^{\mathrm{t}} \tag{4.37}$$

$$\nabla\psi_j(\mathrm{r}) \cdot \nabla\psi_i(\mathrm{r}) = \frac{\beta_j}{2A}\frac{\beta_i}{2A} + \frac{\gamma_j}{2A}\frac{\gamma_i}{2A} \tag{4.38}$$

$$\nabla\psi_j(\mathrm{r}) \cdot \nabla\psi_i(\mathrm{r}) = \frac{\beta_j\beta_i + \gamma_j\gamma_i}{4A^2} \tag{4.39}$$

Ahora reemplazando en (4.8) y como el cálculo integral es sobre el dominio Ω del elemento e, el cálculo integral del elemento e queda como:

$$k_{ij}^e = k(r)\frac{\beta_j\beta_i + \gamma_j\gamma_i}{4A^2}\int_{\Omega^e} d\Omega^e \tag{4.40}$$

Como la integral del elemento es el área del mismo, es decir $\int_{\Omega^e} d\Omega^e = A^e$ el cálculo de k_{ij} de un elemento e queda formalmente como:

$$k_{ij}^e = k(r)\frac{\beta_j^e\beta_i^e + \gamma_j^e\gamma_i^e}{4(A^e)^2}A^e = k(r)\frac{\beta_j^e\beta_i^e + \gamma_j^e\gamma_i^e}{4A^e} \tag{4.41}$$

4.3.3.4 Calculo de q_i^e

En el método de elementos finitos tradicionalmente existen dos formulaciones para el cálculo del vector de cargas, una basada en una carga distribuida uniformemente entre los nodos y otra basada en la carga puntual en uno de los nodos. A continuación se muestran los dos tipos de cálculo, donde q_0 es el número de fotones en la fuente, que representa la carga:

$$q_i = \int_\Omega q_0(r)\psi_i(\mathrm{r})d\Omega \tag{4.42}$$

Caso 1: Distribución uniforme

$$2A^e\int_0^1\int_0^{1-L_j^e}(L_i^e)dL_i^e dL_j^e = 2A^e\int_0^1\frac{\left(1-L_j^e\right)^2}{2}dL_j^e = \frac{A^e}{3} \tag{4.43}$$

$$\therefore\ q_i = q_0\ \frac{A^e}{3} \tag{4.44}$$

Caso 2: Distribución Delta de Dirac

El Delta de Dirac es una función generalizada que viene definida por la siguiente integral:

$$\int_{-\infty}^{\infty}\delta(r-r')f(r)dr = f(r') \implies \int_{-\infty}^{\infty}\delta(r)dr = 1 \tag{4.45}$$

ahora si se define como distribución de densidad

$$\int_a^b \delta(r-r')f(r)dr \;=\; \begin{cases} f(r'), & a<r'<b \\ 0, & EOC \end{cases} \tag{4.46}$$

ahora si utilizamos la propiedad 4.47 en q_i

$$q_i = q_0 \int_{\Omega^e} \delta(x-x_0, y-y_0)\psi_i^e(x,y)d\Omega \;=\; q_0\psi_i^e(x_0,y_0) \tag{4.47}$$

$$\Longrightarrow\; \psi_i^e(x_0,y_0) = \begin{cases} \psi_i^e(x_0,y_0), & si\ (x_0,y_0)\ \in\ \Omega^e \\ 0\,, & EOC \end{cases} \tag{4.48}$$

Lo que en la práctica significa que sólo se evalúa en el elemento que contiene a la fuente y 0 en otro caso. El valor de q_0 es el número de fotones en la fuente y se define experimentalmente, se asume que los fotones se dirigen isotrópicamente por el plano de dos dimensiones sobre el cual se realiza la aproximación de difusión de fotones.

El cálculo del término fuente es parte activa de la investigación actual en el problema de difusión, ya que la aproximación de la difusión es débil en valores cercanos a la fuente, lo cual puede ocasionar grandes errores en la reconstrucción de imágenes. Por tal razón esta problemática es abordada principalmente por expertos en física y electromagnetismo y la mayoría de las estimaciones son en base a la intensidad relativa y por consiguiente a la obtención de fluencias relativas de fotones (Arridge, 1999) (Khan, y otros, 2003) (S. Fantini, 1997) (Tarvainen, 2006).

5 MODELAMIENTO ORIENTADO A OBJETOS DE LA APROXIMACIÓN DE LA DIFUSIÓN CON ELEMENTOS FINITOS

5.1 MODELAMIENTO ORIENTADO A OBJETOS DEL MÉTODO DE ELEMENTOS FINITOS

El diseño de un software científico generalmente se modela para un proceso específico. En el problema planteado en esta tesis no existe el concepto de clientes o usuarios y el software es utilizado sólo por una entidad, la cual puede ser un usuario o una máquina. Entonces la propuesta de diseño se basa en los conceptos fundamentales en que se sustenta el modelo de elementos finitos y los tipos de problemas asociados a este método y no en el diseño basado en UML y casos de uso, que es el modelo tradicional de modelamiento en programación orientada a objetos.

La programación orientada a objetos puede ser vista como un modelo análogo de los fenómenos del mundo real. Es una representación de una entidad discreta, tanto del mundo real como del conceptual. Así pues, un objeto puede ser una abstracción, concepto o cosa con los límites bien definidos y con significado en el sistema.

Los conceptos fundamentales del método de elementos finitos se pueden describir utilizando un mapa conceptual del método de elementos finitos el cual se puede observar en la figura 5.1.

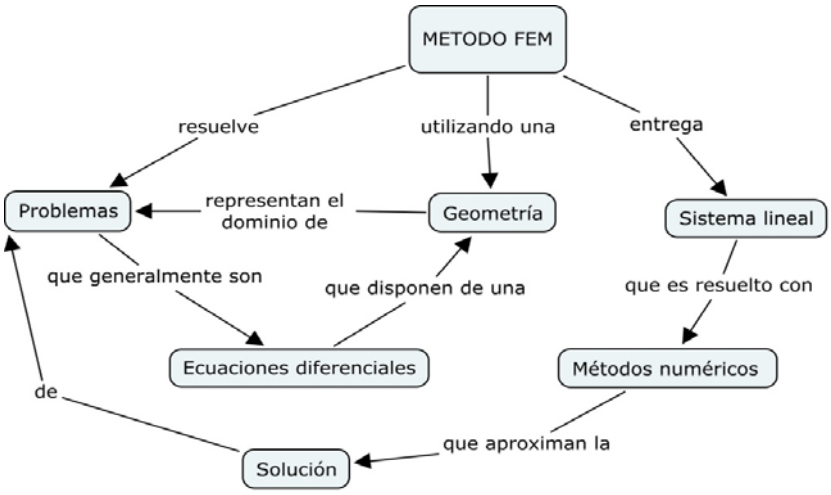


Figura 5.1.- *Mapa conceptual del método de elementos finitos*

En el mapa conceptual del método de elementos finitos (Figura 5.1) existen tres conceptos fundamentales en el método. El primero es que permite solucionar *problemas* que generalmente son *ecuaciones diferenciales*. El segundo es que dispone de una *geometría* que representa al dominio físico de un problema específico. El tercero es que el método entrega un *sistema lineal*, que al ser resuelto con algún método numérico aproxima a la solución del problema específico, donde en nuestro caso el problema específico es la aproximación de difusión de fotones de luz en medios turbios.

A continuación se realiza un modelamiento a partir de los conceptos derivados del mapa conceptual y sus análogos en el modelamiento orientado a objetos del método de elementos finitos aplicado a la solución de ecuaciones diferenciales en general y se utiliza la aproximación de la difusión de fotones luz en medios turbios como modelamiento de un problema específico a resolver.

5.1.1 Definición de clases del método de elementos finitos

Del mapa conceptual (Figura 5.1) podemos decir que una solución de un problema diferencial utilizando el método de elementos finitos se caracteriza por la división geométrica del dominio que representa el dominio físico de un problema real, el cual modelado matemáticamente permite ser tratado con métodos numéricos que entregan un sistema lineal que al ser resuelto da como resultado una aproximación numérica del problema real. Estas abstracciones conceptuales y relaciones se pueden observar en la figura 5.2.

De las abstracciones y relaciones de la figura 5.2 se puede obtener un modelo orientado a objetos que permita la generalización del método de elementos finitos, destacando los conceptos más importantes. Estos conceptos son: el problema especifico y sus características, el tipo de problema diferencial a resolver, la geometría disponible y el sistema lineal, donde la interacción entre todos los conceptos dan como resultado el método de elementos finitos o FEM (siglas en inglés). En la figura 5.3 se muestra las relaciones y clases principales del modelo orientado a objetos del método de elementos finitos.

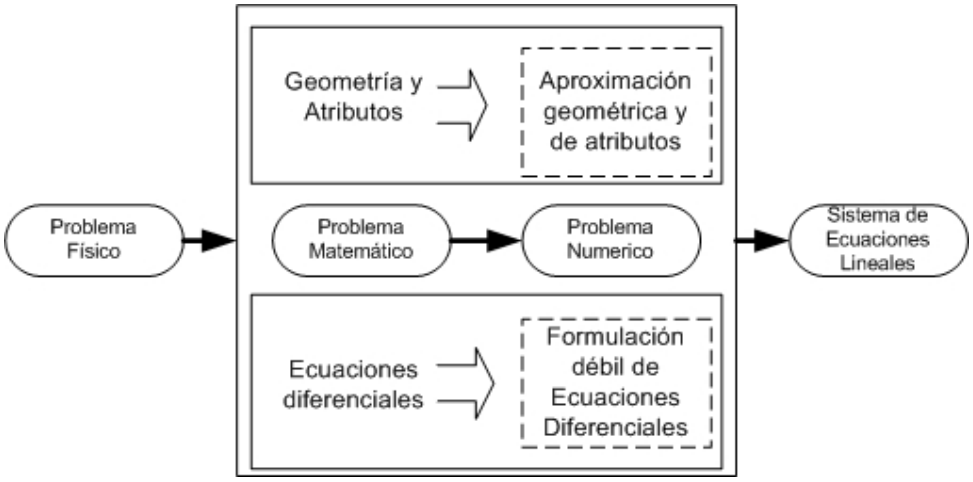


Figura 5.2.- *Abstracciones y relaciones del método de elementos finitos.*

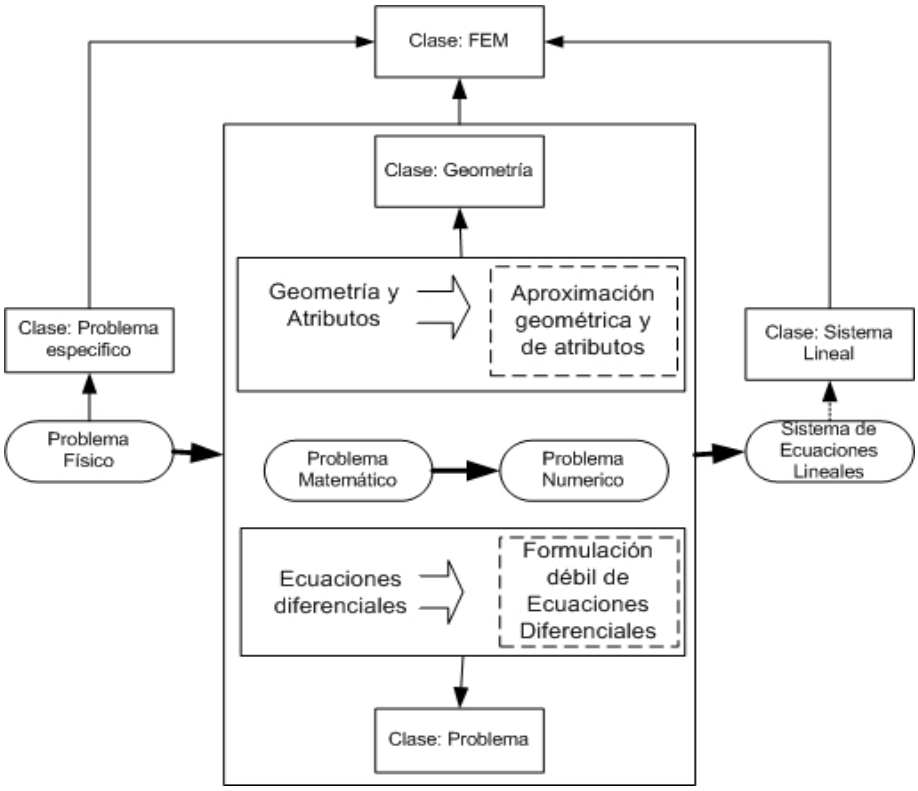


Figura 5.3.- *Relaciones (flechas) y clases principales en el modelamiento orientado a objetos del método de elementos finitos*.

Las relaciones que se observan en la figura 5.3 nos indican la dependencia de las clases. Estas dependencias serán utilizadas como antecedentes para el diseño del diagrama de

clases y diagramas de herencia del modelo orientado a objetos del método de elementos finitos que se propone en este trabajo de tesis.

El diagrama de clases representa el método general de elementos finitos y su funcionamiento conceptual y los diagramas de herencia corresponden a las relaciones de una clase general a una más específica, las cuales heredan propiedades y métodos de las clases generales.

La documentación que se presenta a continuación tiene como objetivo fundamental además de explicar la derivación de las distintas clases del modelo, ser un antecedente útil en el escalamiento del software de elementos finitos que se propone en este trabajo de tesis. Además de modelar el método de elementos finitos con programación orientada a objetos, se construye una clase específica para el problema de difusión de fotones de luz en medios turbios, y se comenta sobre los atributos principales, métodos más relevantes y cuál es su funcionalidad en el modelo propuesto.

5.1.1.1 Modelamiento orientado a objetos del problema diferencial

Definición del tipo de problema

En el método de elementos finitos, las matrices que componen al sistema lineal se construyen a partir de los cálculos integrales elementales (ver 4.3.3) que se obtienen en los nodos de los elementos que representan el dominio. El almacenamiento de los cálculos integrales se realiza a partir de la numeración global de los nodos y este proceso es similar para formulaciones diferenciales elípticas, parabólicas o hiperbólicas. Por lo tanto se puede generalizar la definición de un problema utilizando una clase que determine qué tipo de aproximación se quiere realizar y lo que hará diferente una solución de otra serán las características de la solución a aproximar. Estas características quedan definidas a través de la forma en cómo se realiza la aproximación, donde destacan el tipo de representación del borde del dominio (matricial o vectorial), el tipo de vector de cargas. El modelamiento de los distintos tipos de ecuaciones diferenciales (elípticas, parabólicas o hiperbólicas) queda definido según sus formulación variacional que en el caso de ecuaciones parabólicas e hiperbólicas la variable tiempo es considerada en el modelamiento de la solución.

En este trabajo de tesis sólo se modelo un problema diferencial elíptico, por lo cual el desarrollo de otros tipos de problemas debe ser abordado en posteriores

versiones del software que se propone en este trabajo. A pesar que sólo se implementó el método de elementos finitos para ecuaciones diferenciales elípticas, el sistema está preparado para la incorporación de clases y métodos que implementen otros tipos de formulaciones. Para tal efecto el requerimiento en la definición del problema será, el tipo de ecuación diferencial a resolver y para el modelamiento general del problema se incluye la representación del borde o frontera del problema y el tipo de vector de cargas que se utiliza. Entonces la clase que se utilizara para definir un problema diferencial sólo requerirá estas tres características. La clase que define un problema diferencial la denominaremos "***Problema***". Los atributos de la clase "***Problema***" se muestran en la tabla 5.1.

Tabla 5.1.- *Atributos de la clase "**Problema**".*

Tipo	Nombre	Descripción
int	**tipo_problema**	Tipo de problema: 1-Eliptico, 2-Parabolico, 3 - Hiperbólico
int	**tipo_derivacion**	Tipo derivación en el borde: 1 - Vectorial, 2 – Matricial
int	**tipo_contribucion_carga**	Tipo contribución de vector de cargas: 0 - Nula, 1 - Constante, 2 - Función de distribución de Dirac

Definición del problema específico (difusión de fotones de luz en medios turbios)

El modelamiento del problema específico es en base a las propiedades que caracterizan a la ecuación de difusión de fotones de luz, donde destacan los factores de dispersión y absorción del medio, el número de fuentes de fotones de luz con sus respectivas ubicaciones geométricas y densidades más la función de fase del factor de dispersión reducido.

Además se considera parte del problema el tipo de ecuación a resolver que en nuestro caso es una ecuación diferencial elíptica, el tipo de representación matemática que se utilizara en el borde del dominio ya sea en formato matricial o vectorial y el tipo de contribución al vector de cargas en las fuentes de fotones, ya sea nula, constante o del tipo de distribución de Dirac. Estas últimas características pueden ser heredadas de la clase "***Problema***" definida en el apartado anterior. La aproximación de la ecuación de transporte de fotones de luz utilizando la aproximación de la difusión (**ETRAD**) es una ecuación diferencial hiperbólica, pero en nuestro caso solo se modelara el problema como un sistema en estado estable (sin la variable tiempo), por lo cual la ecuación

diferencial a resolver es del tipo elíptica y su representación matemática en forma general es la ecuación 5.1.

$$-\nabla \cdot (c\nabla u) + au = f \tag{5.1}$$

Para modelar el problema de difusión de fotones de luz se diseña una clase llamada "***ETRAD***" que herede propiedades de la clase "***Problema***". Se implementaran los métodos de carga de información al sistema que contienen las características físicas del problema de difusión, como lo son la absorción, dispersión, función de fase del coeficiente de dispersión reducido, número de fuentes de fotones y sus densidades respectivas. En la tabla 5.2 se muestran los atributos más relevantes de la clase "***ETRAD***". En la tabla 5.3 se muestran los métodos más relevantes y en la figura 5.4 se muestra el diagrama de herencia de la clase "***Problema***".

Tabla 5.2.- *Atributos de la clase "**ETRAD**".*

Tipo	Nombre	Descripción
Vector *	**Us**	Factor de dispersión
Vector *	**Ua**	Factor de absorción
Matriz *	**Fuentes_Fotones**	Densidades y ubicaciones de fuentes de fotones
double	**fFase**	Función de fase
char *	**NombreArchivoAbsorcion**	Nombre del archivo de datos de absorción
char *	**NombreArchivoScattering**	Nombre del archivo de datos de dispersión
char *	**NombreArchivoFuentes**	Nombre del archivo de datos de las fuentes de fotones

Tabla 5.3.- *Métodos de la clase "**ETRAD**".*

Tipo	Nombre	Descripción
double	**Coeficiente_difusion** (**double** UaValor, **double** UsValor, **double** fase)	Retorna el coeficiente de difusión reducido de la ecuación de difusión de la luz, dado absorción, dispersión y función de fase
void	**Asigna_Funcion_Fase** (**double** fase)	Asigna el valor de la función de fase en la difusión
void	**Carga_Absorcion** (**char** *arch)	Carga del archivo que contiene los datos de absorción en los distintos subdominios
void	**Carga_Scattering** (**char** *arch)	Carga el archivo que contiene los datos de

		dispersión en los distintos subdominios
void	**Carga_Fuentes** (**char** *arch)	Carga el archivo que contiene los datos de las fuentes
void	**Set_Longitud_Vector_Ua** (**int** longitud)	Reserva memoria para el vector que almacenara los datos de absorción
void	**Set_Longitud_Vector_Us** (**int** longitud)	Reserva memoria para el vector que almacenara los datos de dispersión
void	**Set_Valor_Ua**(**int** indice, **double** valor)	Asigna el valor de absorción al vector **Ua** en la posición "indice"
void	**Set_Valor_Us**(**int** indice, **double** valor)	Asigna el valor de dispersión al vector **Us** en la posición "indice"
Otros métodos son retornos de datos del objeto "*ETRAD*"		

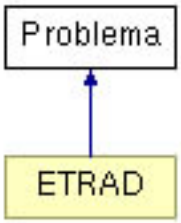


Figura 5.4.- *Diagrama de herencia de la clase "**Problema**".*

5.1.1.2 Modelamiento orientado a objetos de la geometría de elementos finitos

En el modelamiento de una solución con elementos finitos, el concepto de geometría es la base fundamental del método, y la diferencia entre los algoritmos y cálculos que se deben realizar dependen estrictamente del tipo de geometría que se utilice. Pero a pesar de esta variante en el cálculo geométrico, la dimensión de la geometría y la numeración global de los nodos asociadas a los elementos que representan el dominio, tienen el mismo procedimiento de numeración y asociación sin importar el tipo de geometría que se utilice.

Los componentes principales que se utilizan para construir una geometría son los nodos y los elementos finitos. Entonces antes de definir formalmente la geometría, definiremos los nodos y elementos finitos que son los componentes principales que definen una geometría de elementos finitos.

Nodos

Los nodos son puntos geométricos del dominio del problema con una numeración global y los diferentes tipos de nodos dependen de la dimensión del problema. Por tal razón se diseño una clase principal llamada "***Nodo***" que tiene un identificador de nodo y la dimensión del mismo. En las tablas 5.4 y 5.5 se muestran los atributos y métodos de la clase "***Nodo***", respectivamente. Además como existen tres dimensiones posibles en el modelamiento de elementos finitos se diseñaron tres clases de nodos que heredan las propiedades de la clase "***Nodo***" los cuales se definen según la dimensión que se utilice en la geometría, por lo tanto se definieron las subclases "***Nodo_1D***", "***Nodo_2D***" y "***Nodo_3D***". En la figura 5.5 se muestran el diagrama de herencia la clase "***Nodo***".

Tabla 5.4.- *Atributos de la clase "**Nodo**".*

Tipo	Nombre	Descripción
unsigned long long	**Id_nodo**	Identificador del nodo
int	**Dimensión_nodo**	Dimensión del nodo

Tabla 5.5.- *Métodos de la clase "**Nodo**".*

Tipo	Nombre	Descripción
void	**Set_Dimension_Nodo** (**int** dimension)	Asigna dimensión del nodo
void	**Set_Id_Nodo** (**unsigned long long** id)	Asigna el identificador del nodo
int	**Get_Dimension_Nodo**()	Retorna la dimensión del nodo
unsigned long long	**Get_Id_Nodo**()	Retorna el identificador del nodo

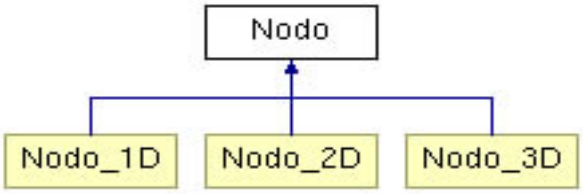


Figura 5.5.- Diagrama de herencia de la clase "***Nodo***"

El modelo de elementos finitos que se implementó es para dos dimensiones y sólo se utiliza la clase "***Nodo_2D***" y los atributos de valores nodales, pertenencias nodales que identifican si el nodo está en el interior del dominio o pertenece al borde del dominio,

sólo se definen para la clase "***Nodo_2D***". En las tablas 5.6 y 5.7 se muestran los atributos y métodos de la clase "***Nodo_2D***" respectivamente.

Tabla 5.6.- *Atributos de la clase "**Nodo_2D**".*

Tipo	Nombre	Descripción
double	**valorX**	Posición nodal en el eje x
double	**valorY**	Posición nodal en el eje y
double	**valorNodal**	Valor del nodo
int	**NV**	Número de elementos que contienen al nodo
int	**pBorde**	Indicador de nodo frontera donde es 1 si es nodo frontera
Vector_Entero *	**vecindadNodo**	Elementos que contienen al nodo
Int	**banderaNodo**	Bandera para el cálculo de fuentes que comparten un mismo nodo.

Tabla 5.7.- *Métodos de la clase "**Nodo_2D**".*

Tipo	Nombre	Descripción
void	**Set_bandera** (**int** valor)	Asigna el valor de la bandera
void	**Set_X**(**double** x)	Asigna un valor en x
void	**Set_Y**(**double** y)	Asigna un valor en y
void	**Set_pBorde**(**int** valor)	Asigna como nodo en el borde
void	**Set_vNodal**(**double** valor)	Asigna el valor del nodo
void	**Set_Orden_Vecindad**(**int** orden)	Asigna el número de elementos que contienen al nodo
void	**Set_Vecindad** (**int** id, **int** valor)	Asigna información de elemento contenedor de nodo
Otros métodos son retornos de datos del objeto "***Nodo_2D***"		

Elementos finitos

Si bien los elementos finitos pueden ser de una, dos o tres dimensiones, todos deben tener un identificador de elemento. Entonces un elemento finito puede ser definido por una clase que contenga estos dos atributos (identificador de elemento y el tipo de elemento). A esta clase se le da el nombre de "***Elemento***". En las tablas 5.8 y 5.9 se muestran los atributos y métodos de la clase "***Elemento***" respectivamente.

Tabla 5.8.- *Atributos de la clase "**Elemento**".*

Tipo	Nombre	Descripción
unsigned long long	**id_elemento**	Identificador del Elemento
Int	**tipo_elemento**	Tipo de elemento: 1 - Lineal, 2 - Triangular, 3 - Cuadrilátero, 4 - Tetraedro

Tabla 5.9.- *Métodos de la clase "**Elemento**".*

Tipo	Nombre	Descripción
void	**Set_Id_Elemento** (**unsigned long long** valor)	Asigna Id del elemento
void	**Set_Id_Tipo**(**int** tipo)	Asigna el tipo de elemento
unsigned long long	**Get_Id_Elemento**()	Retorna el id del elemento
Int	**Get_Tipo_Elemento** ()	Retorna el tipo de elemento

En este trabajo de tesis sólo se utilizan elementos finitos triangulares lineales de tres nodos, pero el modelamiento orientado a objetos que se propone permite la incorporación de librerías que utilicen otros tipos de elementos.

Para la utilización de elementos finitos triangulares se definió una clase llamada "***Elemento_Triangular***" que hereda las propiedades de la clase "***Elemento***" donde los atributos principales son: el número de nodos por elemento, los nodos que componen al elemento, el dominio al cual pertenece, identificador si contiene cargas puntuales (ej. Fuentes de fotones en el problema de difusión), variables físicas del dominio (ej. dispersión y absorción en el problema de difusión), matrices locales (carga, masa, rigidez, gradiente, pertenencia global nodal, coeficientes de permutación del elemento finito) y los vectores locales (carga, centroide y gradiente).

Los atributos la clase "***Elemento_ Triangular***" se muestran en la tabla 5.10. El detalle de los métodos implementados en la clase "***Elemento_ Triangular***" se encuentran en el anexo D.

Tabla 5.10.- *Atributos de la clase "**Elemento_ Triangular**".*

Tipo	Nombre	Descripción
Nodo_2D *	**Nodos_Elemento**	Nodos locales del elemento, se utiliza como un vector de objetos **Nodo_2D[]**
int	**numero_nodos_elemento**	Numero de nodos del elemento
int	**dominio**	Dominio al cual pertenece el elemento

int	**eFuente**	Indica si el elemento contiene a una fuente
double	**area**	Area del elemento
double	**kVariable**	Variable física del elemento asociada al comportamiento físico - geométrico
double	**mVariable**	Variable física del elemento asociado a la físico-masa.
double	**sVariable**	Variable de dispersión del elemento
Vector *	**Carga**	Vector de Cargas del elemento
Matriz *	**Coeficientes_Permutacion**	Coeficientes geométricos de la función de forma del elemento
Matriz *	**MRE**	Matriz Rigidez Elemento
Matriz *	**MMC**	Matriz de Masa Elemento
Matriz_Entera *	**Pertenencia**	Matriz de pertenencia global (numeración nodal global) del elemento.

Para los elementos que pertenecen al borde del dominio se define una clase llamada "***Elemento_Frontera***" que hereda las propiedades de la clase "***Elemento_Triangular***". Esta clase además de tener las propiedades heredadas de la clase "***Elemento_Triangular***" tiene atributos y métodos que caracterizan el comportamiento del elemento finito en el borde, como lo son el cálculo de la matriz o vector de carga en el borde del dominio según sea el tipo de modelamiento utilizado para la aproximación que se definió en la clase "***Problema***" (ver Figura 5.4). En la figura 5.6 se muestra el diagrama de herencia de clases de la clase "***Elemento***".

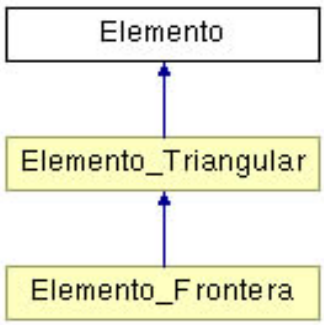


Figura 5.6.- *Diagrama de herencia de la clase "**Elemento**".*

Los atributos de la clase "***Elemento_Frontera***" se muestran en la tabla 5.11 y el detalle de los métodos implementados de la clase "***Elemento_Frontera***" se encuentra en el anexo D.

Tabla 5.11.- *Atributos de la clase "**Elemento_Frontera**".*

Tipo	Nombre	Descripción
Nodo_2D *	**Nodos_Frontera**	Arreglo de objetos **Nodo_2D[]** que contendrán información sobre la frontera del dominio del problema
Vector_Entero *	**Nodos_Borde**	Índices globales de los nodos de la frontera
Matriz *	**MIB**	**Matriz** de borde Integral
Vector *	**VIB**	**Vector** de borde Integral
int	**dominio_frontera**	Dominio al cual pertenece la frontera
double	**kScatering**	Constante de dispersión
double	**Norma**	Distancia, usada para el cálculo integral entre dos puntos
Vector *	**Normal**	Vector normal

5.1.1.3 Geometría de elementos finitos

La clase principal que define la geometría es llamada "***Geometría***" y tiene atributos relacionados con la dimensión que se utiliza (una, dos o tres dimensiones), el número de elementos, número de nodos, numero de nodos en las fronteras de la geometría y el número de puntos de control que se encuentran en la geometría (para el caso de difusión, son el numero de fuentes de fotones). En las tablas 5.12 y 5.13 se muestran los atributos y métodos de la clase "***Geometría***".

Tabla 5.12.- *Atributos de la clase "**Geometría**".*

Tipo	Nombre	Descripción
unsigned long long	**Numero_Nodos**	Numero de nodos
unsigned long long	**Numero_Elementos**	Número de elementos.
int	**DIM**	Dimensión del dominio
unsigned long	**Numero_Fronteras**	Numero de fronteras
int	**Numero_Fuentes**	Numero de fuentes

Tabla 5.13.- *Métodos de la clase "**Geometría**".*

Tipo	Nombre	Descripción
void	**Asigna_Numero_de_Nodos**(**int** num)	Retorna el número de nodos
void	**Asigna_Numero_de_Elementos**(**int** num)	Asigna el número de elementos de la geometría
void	**Asigna_Numero_de_Fronteras**(**int** num)	Asigna el número de fronteras de la geometría
void	**Asigna_Numero_de_Fuentes**(**int** num)	Asigna el número de fuentes puntuales para problemas de difusión
int	**Retorna_Numero_de_Fuentes**(**void**)	Retorna el número de fuentes
unsigned long	**Retorna_Numero_Fronteras**(**void**)	Retorna el número de fronteras
unsigned long long	**Retorna_Numero_Nodos**(**void**)	Retorna el número de nodos
unsigned long long	**Retorna_Numero_Elementos**(**void**)	Retorna el número de elementos

Además es necesario definir una clase en la cual se definan los métodos y atributos asociados a la dimensión del problema, para posteriores aproximaciones a partir de la aproximación inicial del método de elementos finitos e información geométrica del dominio global. Para tal efecto se diseña una clase que hereda propiedades de la clase "***Geometría***" llamada "***Geometria2D***". En la tabla 5.14 se muestran los atributos de la clase "***Geometría2D***". El detalle de los métodos de la clase "***Geometria2D***" se encuentra en el anexo D.

Tabla 5.14.- *Atributos de la clase "**Geometria2D**".*

Tipo	Nombre	Descripción
Matriz_Entera *	**GrillaDominio**	Grilla que almacena datos del dominio de la geometría: 1 si pertenece al dominio del problema y 0 EOC.
Nodo_2D *	**PuntosGrilla**	Puntos a evaluar de la grilla.
Nodo_2D *	**CentroidesGrilla**	Centroides de particiones de la grilla, utilizado en la selección de elementos que están contenidos en las sub grillas particionadas para el proceso paralelo
unsigned long long	**NumeroPuntosGrilla**	Número de puntos a calcular de la grilla.

int	**tipoGeometria2D**	Tipo de geometría: 2 - Triangulo, 3 - Cuadrilátero
int	**nParticiones**	Numero de particiones: utilizado en calculo paralelo
int	**FilaGrilla, ColumnaGrilla**	Numero de filas y columnas de la partición paralela.
int	**ResolucionX, ResolucionY**	Resolución en ejes X e Y de la grilla
double	**minX, minY, maxX, maxY**	Fronteras de la Grilla.
double	**Area_DIferencial**	Area diferencial
int	**tipoGrilla**	Tipo aproximación de la Grilla: 1 – en vértice, 2 - en centro geométrico de 4 vértices que forman un cuadrilátero
Doublé	**distanciaX, distanciaY**	Medidas de distancias en X e Y de la grilla.

Los elementos finitos son la base sobre la cual se construye la geometría que representa el dominio del problema. Todos los cálculos geométricos necesarios para la obtención de las matrices que representan el comportamiento físico – geométrico dependen principalmente del tipo de elemento finito que se utilice. En este trabajo de tesis se utilizaron elementos finitos triangulares lineales de tres nodos y se define una clase llamada "***Geometria2D_Triangulos***" que contiene los atributos y métodos asociados a los cálculos geométricos e integrales necesarios para la construcción del vector de carga, matrices de rigidez y de masa de cada elemento finito triangular.

Además, esta clase es la encargada de cargar la información proporcionada por los archivos que contienen datos de la representación geométrica del dominio de un problema diferencial específico. En la tabla 5.15 se muestran los atributos de la clase "***Geometria2D_Triangulos***" y en la figura 5.7 se muestra el diagrama de herencia de clases de la clase "Geometría". El detalle de los métodos implementados en la clase "***Geometria2D_Triangulos***" se encuentran en el anexo D.

Tabla 5.15.- *Atributos de la clase "**Geometria2D_Triangulos**".*

Tipo	Nombre	Descripción
Nodo_2D *	**Nodos_Geometria**	Arreglo de objetos **Nodo_2D[]** que contiene los nodos de la geometría.
Vector_Entero *	**Nodos_Frontera_Geometria**	Vector que contiene los

		identificadores de los nodos frontera.
Elemento_Triangular *	**Elementos_Geometria**	Arreglo de objetos **Elemento_Triangular[]** que contiene los elementos de la geometría.
Elemento_Frontera *	**Fronteras_Geometria**	Arreglo de objetos **Elemento_Frontera[]** que contiene los elementos que pertenecen a la frontera (borde) de la geometría.
Nodo_2D *	**Fuentes**	Arreglo de Objetos **Nodo_2D[]** que contiene las ubicaciones de las fuentes puntuales
double	**Area_Dominio**	Área del dominio global
unsigned long long	**nPuntosInterior**	Número de puntos internos (puntos de la grilla que pertenecen a la geometría FEM)
unsigned long long	**nFronteraDominio**	Número de puntos frontera.

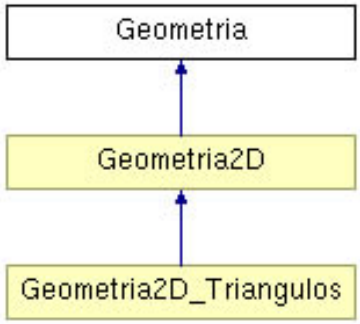


Figura 5.7.- *Diagrama de herencia de la clase "**Geometría**".*

5.1.1.4 Modelamiento orientado a objetos del sistema lineal

Para resolver un sistema lineal $Ax = b$ se puede utilizar una diversidad de métodos numéricos iterativos o directos, donde algunos pueden ser más eficientes que otros dependiendo de las distribuciones y estructuras de los datos en las matrices (dispersas, densas, simétricas, etc.). Pero todos los métodos necesitan la misma información en función de los datos de entrada (matriz A y vector b), la diferencia radica en los

parámetros utilizados por cada método (tolerancia, iteraciones, etc.). Entonces al diseñar una clase que permita la implementación de éstos, ésta debe tener entradas genéricas de los datos y parámetros y sólo se deben implementar los diversos procedimientos numéricos como métodos de una misma clase.

Para tal efecto se diseña una clase llamada "***Sistema_Lineal***" donde se implementaron varios métodos numéricos que resuelven sistemas lineales del tipo $Ax = b$. En la tabla 5.16 se muestran los atributos de la clase "***Sistema_Lineal***". El detalle de los métodos implementan en la clase "***Sistema_Lineal***" se encuentran en el anexo D.

Tabla 5.16.- *Atributos de la clase "**Sistema_Lineal**".*

Tipo	Nombre	Descripción
unsigned long long	**nDesconocido**	Numero de variables desconocidas
int	**NP**	Numero de procesos
unsigned long long	**Iteraciones**	Numero de iteraciones en métodos iterativos
int	**Traza**	Indica si se desea almacenar en archivo la convergencia del error
Vector_Entero *	**Indice_Filas**	Usado en el cálculo paralelo para indicar fila inicial global
Doublé	**tSistema**	Tiempo utilizado en resolver sistema

5.1.1.5 Modelamiento orientado a objetos del método de elementos finitos

El método de elementos finitos requiere una interacción de varios procesos como se puede observar en la figura 5.3. Donde la información proveniente de los procesos de cálculos geométricos, el problema específico y el sistema lineal que se obtiene son en conjunto el método de elementos finitos.

Entonces se diseña una clase principal llamada "***FEM***" en donde se define e implementan los procedimientos generales del método de elementos finitos. En esta clase se realizan las declaraciones y llamadas a los métodos definen la geometría que se va a utilizar, se realizan las llamadas a los métodos que construyen las matrices que componen al sistema de ecuaciones y su posterior resolución con procedimientos numéricos. Además se declaran los procedimientos paralelos utilizados en el software de elementos finitos que se presenta en este trabajo de tesis, los cuales serán explicados en profundidad en el capítulo 6. En la tabla 5.17 se muestran los atributos más

relevantes de la clase "***FEM***". El resto de los atributos y métodos se encuentran en el anexo D.

Tabla 5.17.- *Atributos de la clase "**FEM**".*

Tipo	Nombre	Descripción
Matriz *	**A**	**Matriz** de rigidez del sistema
Vector *	**B**	**Vector** de cargas del sistema
Vector *	**X**	**Vector** incógnita del problema
Vector_Entero *	**FronteraDominioGlobal**	**Vector** Fronteras
Sistema_Lineal *	**Sistema**	Puntero al objeto **Sistema_Lineal**
Rutinas_Paralelas *	**Proceso_Paralelo**	Puntero al objeto **Rutinas_Paralelas**
Geometria2D_Triangulos *	**GeometriaTrabajo**	Puntero al objeto **Geometria2D_Triangulos**
Int	**nProcesadores**	Numero de procesadores a utilizar

Como el método de elementos finitos en su implementación depende directamente de la dimensión del problema se diseña una clase que hereda las propiedades de la clase "***FEM***" llamada "***FEM2D***", donde destacan los métodos de construcción de las matrices globales que componen al sistema de ecuaciones a partir de los cálculos realizados para una geometría de dos dimensiones, la cual está es compuesta en una geometría de elementos finitos triangulares de tres nodos que está implementada en la clase "***Geometria2D_Triangulos***".

Se utiliza esta subclase "***FEM2D***" principalmente por que el método de elementos finitos es genérico en casi todo el proceso, exceptuando los procesos relacionados con los cálculos geométricos e integrales y su implementación puede ser extendida fácilmente a otros tipos de elementos finitos utilizando subclases que definan y utilicen los procedimientos y cálculos necesarios para cada tipo de geometría elemental. En las tablas 5.18 y 5.19 se muestran los atributos y métodos de la clase "***FEM2D***". En la figura 5.8 se muestra el diagrama de herencia de clases de la clase "***FEM***".

Tabla 5.18.- *Atributos de la clase "**FEM2D**".*

Tipo	Nombre	Descripción
Matriz_Entera *	**PerteneciaEL**	Matriz de pertenencia de un elemento
Matriz *	**RigidezEl**	Matriz de rigidez de un elemento

Matriz *	**MasaEl**	Matriz de masa de un elemento
Elemento_Triangular *	**EL**	Puntero a objeto **Elemento_Triangular**
Elemento_Frontera *	**ELF**	Puntero a objeto **Elemento_Frontera**
Vector *	**QEl**	Vector de cargas de un elemento
Vector_Entero *	**PerteneciaELF**	Vector de pertenencia de un elemento frontera
Matriz *	**BordeEl**	Matriz de borde

Tabla 5.19.- Métodos de la clase "***FEM2D***"

Tipo	Nombre	Descripción
void	**Libera_Memoria_Carga**()	Libera carga de memoria
void	**Inicializar_FEM2D**(**int** Tipo_resolucion_AX)	Reserva memoria para el sistema de ecuaciones
void	**Llena_Matriz_B_FEM**()	Método de construcción de la vector de carga global B
void	**Llena_Matriz_A_FEM**(**int** borde, **int** tipo_solucion)	Método de construcción de la matriz global A
void	**Set_Suma_Nodal_B_FEM**(**unsigned long long** i, **double** aNodal)	Suma un aporte nodal al vector de cargas global B del sistema de ecuaciones
void	**Set_Suma_Nodal_A_FEM**(**unsigned long long** i, **unsigned long long** j, **double** aNodal)	Suma un aporte nodal a la matriz global A del sistema de ecuaciones
void	**Set_Reserva_Memoria_A_FEM**(**unsigned long long** nNodo)	Reserva memoria para la matriz global A del sistema de ecuaciones
void	**Set_Reserva_Memoria_B_FEM**(**unsigned long long** nNodos)	Reserva memoria para el vector global B del sistema de ecuaciones
void	**Llena_Matriz_B_FEM_2D_Triangulos**()	Construye el vector de cargas global B del sistema de ecuaciones calculada con elementos finitos triangulares.
void	**Llena_Matriz_A_FEM_2D_Triangulos**(**int** tipo_AX)	Construye la matriz global A del sistema de ecuaciones a partir de las matrices de rigidez y de masa, calculada con elementos finitos triangulares
void	**Aporte_Matriz_A_FEM_Borde_2D_Triangulos**(**int** tipo_AX)	Aporta a la matriz global A del sistema de ecuaciones los cálculos del borde

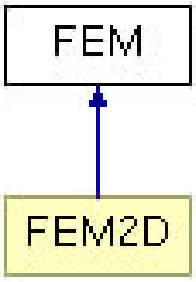


Figura 5.8.- *Diagrama de herencia de la clase "**FEM**".*

5.1.1.6 Modelamiento orientado a objetos de métodos paralelos

La clase en la cual se implementan los métodos paralelos tiene el nombre de "***Rutinas_Paralelas***". Esta clase tiene métodos de almacenamiento, cómputo paralelo y transmisión de datos. Los métodos de almacenamiento están encargados de reservar el uso de memoria de vectores y matrices que contienen información de los nodos, elementos y de matrices del sistema, puntos de una grilla, aproximaciones nodales, etc. que es requerida por los cómputos paralelos de la solución del sistema de ecuaciones y de la aproximación de una grilla de puntos que están contenidos dentro de la geometría de elementos finitos. Se utiliza este almacenamiento y no de arreglo de objetos por la facilidad práctica y eficiente del uso de punteros en la transmisión de datos en los procesos de cómputo paralelo. En la tabla 5.20 se muestran los atributos de la clase "***Rutinas_Paralelas***". El detalle de los métodos implementados en la clase "***Rutinas_Paralelas***" se encuentran en el anexo D.

Tabla 5.20.- *Atributos de la clase "**Rutinas_Paralelas**".*

Tipo	Nombre	Descripción
unsigned long long	**nDesconocido**	Numero de valores desconocidos de la grilla
int	**nProcesadores**	Numero de procesadores a utilizar
Matriz *	**Datos_Elementos**	Matriz que contiene todo los datos de los elementos (id elemento, área, coeficientes de permutación, aproximaciones nodales y listas de nodos)
Vector *	**Datos_Elementos_Vector**	Vector que contiene la información en formato vectorial de **Datos_Elementos**
Vector *	**puntosX**, **puntosY**	Vector que contiene datos de las coordenadas X e Y respectivamente para el cálculo paralelo de la grilla FEM
Vector *	**Densidad_Grilla**	Vector que contiene las densidades obtenidas en el cálculo de la grilla.
Vector *	**nodoX, nodoY**	Vector que contiene datos de los nodos en X e Y respectivamente del dominio FEM
Vector *	**CentroidesGrilla**	Vector que contiene los centroides de la grilla
Matriz_Entera *	**Particion_FC**	Matriz que contiene información sobre la partición paralela de un matriz

Matriz_Entera *	**Particion_Procesadores**	Matriz que contiene información sobre los identificadores de los procesadores
double	**tGrilla**	Tiempo utilizado en comunicación por el cálculo grilla

5.1.1.7 Modelamiento orientado a objetos de datos y manipulación de archivos

Las estructuras de datos más utilizadas en el software de elementos finitos, son los vectores y matrices. Se utilizan vectores en la mayoría de las clases e incluso en la construcción de los objetos que contienen información geométrica como lo son los nodos y elementos, si bien se utilizan arreglos de objetos al definir y construir una geometría basada en nodos y elementos estructuralmente son vectores.

Vectores

Se definen tres tipos de vectores y cada uno asociado a una clase distinta y no se utilizan plantillas principalmente por factores de mantención, definición de tipos de datos y procedimientos de cómputo vectorial asociados a los distintos tipos de vectores. Los tres tipos de clases de vectores son: "***Vector***" que es de tipo double, "***Vector_Entero***" que es de tipo entero sin signo y "***Vector_Disperso***" que es un vector double con almacenamiento disperso. En las tablas 5.21 a 5.23 se muestran los atributos de las tres clases de vectores. El detalle de los métodos de las tres clases de vectores se encuentra en el anexo D.

Tabla 5.21.- *Atributos de la clase "**Vector**".*

Tipo	Nombre	Descripción
unsigned long long	Columna	Número de columnas
char *	Nmb	Nombre del vector
double *	V	Puntero al Vector double

Tabla 5.22.- *Atributos de la clase "**Vector_Entero**".*

Tipo	Nombre	Descripción
unsigned long long	**Columna**	Número de columnas
char *	**Nmb**	Nombre del vector
unsigned long long *	**V**	Puntero al Vector entero sin signo

Tabla 5.23.- *Atributos de la clase "**Vector_Disperso**".*

Tipo	Nombre	Descripción
Vector_Entero *	**vdColumnas**	Vector entero que contiene datos de columnas
Vector *	**vdDato**	Vector double que contiene los datos del vector disperso
unsigned long long	**nzDatos**	Número de elementos del vector no ceros
unsigned long long	**dimOriginal**	Dimensión original del vector

Matrices

En el caso de las matrices de definien tres tipos de clases de matrices: "***Matriz***" de tipo doublé, "***Matriz_Entera***" que de tipo entero positivo y "***Matriz_Dispersa***" que es double con almacenamiento disperso y en las tablas 5.24 a 5.26 se muestran los atributos de la tres clases de matrices. El detalle de los métodos implementados en las tres clases de matrices se encuentra en el anexo D.

Tabla 5.24.- *Atributos de la clase "**Matriz**".*

Tipo	Nombre	Descripción
double **	M	Puntero de 2 dimensiones.
unsigned long long	Col	Número de columnas.
unsigned long long	Fil	Número de filas.
char *	Nmb	Nombre de la matriz.

Tabla 5.25.- *Atributos de la clase "**Matriz_Entera**".*

Tipo	Nombre	Descripción
unsigned long long **	**M**	Puntero bidimensional que almacenara los valores de una matriz.
unsigned long long	**Col**	Número de columnas
unsigned long long	**Fil**	Número de filas.
char *	**Nmb**	Nombre de la matriz.

Tabla 5.26.- *Atributos de la clase "**Matriz_Dispersa**".*

Tipo	Nombre	Descripción
Vector_Entero *	**vColumna**	Vector entero sin signo que contiene los datos de columnas
Vector_Entero *	**vFila**	Vector entero sin signo que contiene los datos de

		filas
Vector_Entero *	**vIndiceFila**	Vector Entero sin signo que contiene los índices de cada nueva fila de la matriz original
Vector *	**vDato**	Vector que contiene los datos de la matriz dispersa
unsigned long long	**nDatos**	Número de elementos de la matriz original que son distinto de cero
unsigned long long	**nColumnasDatos**	Dimensión del vector de datos
unsigned long long	**nIndices**	Dimension del vector de indices

Archivos

Para el manejo de archivos se diseña una clase llamada "***Archivo***" que tiene como objetivo la manipulación y construcción de archivos. En la tabla 5.27 y 5.28 se muestran los atributos y métodos de la clase "***Archivo***".

Tabla 5.27.- *Atributos de la clase "**Archivo**".*

Tipo	Nombre	Descripción
int	**Estado**	Indica el estado de uso del archivo (activo – pasivo)
FILE *	**ARCHIVO_TRABAJO**	Puntero a un archivo
char *	**ARCHIVO**	Nombre del archivo
int	**MODO**	Modo en el cual se está utilizando el archivo (lectura – escritura)
char *	**ACCESO**	Definición de permisos del archivo

5.1.1.8 Modelo orientado a objetos del método de elementos finitos aplicado al problema de difusión

A continuación se muestra en la figura 5.9 el diagrama de clases del modelo orientado a objetos del método de elementos finitos aplicado al problema de difusión de fotones de luz en medios difusos que se propone en este trabajo de tesis. Destacan las clases asociadas a la definición de un problema diferencial específico, las clases en la cual se define un problema, las clases asociadas a la geometría, la clase que define el método numérico a utilizar, la clase en la cual se definen los métodos paralelos utilizados en la optimización de los tiempos de cálculo del método y las clases asociadas a la manipulación de archivos y tipos de datos que se utilizan en el método propuesto. El

diagrama de clases de la figura 5.9 sólo contempla la dependencia de las clases y la herencia de las mismas. Un mayor detalle de los módulos se encuentra en el anexo D o bien en la documentación completa del software que se encuentra en el CD adjunto a este trabajo de tesis.

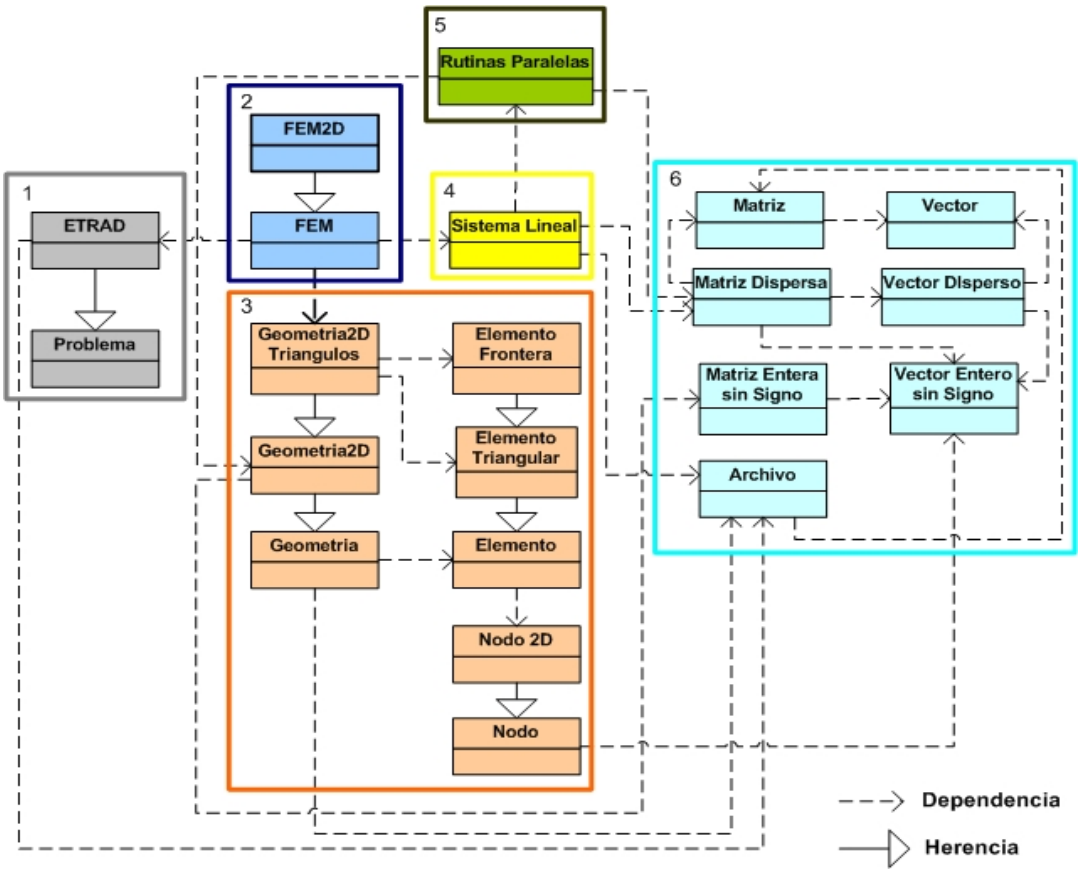


Figura 5.9.- *Diagrama de clases del método FEM – ETRAD: 1) clases asociadas a la definición del problema diferencial, 2) clases del método de elementos finitos donde se define un problema FEM de dos dimensiones asociada a una geometría de triángulos y que entrega un sistema lineal, 3) clases de la geometría de elementos finitos, 4) clase en que se define y resuelven sistemas lineales, 5) clase donde se implementan los algoritmos paralelos, 6) clases para manipulación de archivos y datos (matrices y vectores).*

6 IMPLEMENTACIÓN PARALELA

6.1 INTRODUCCIÓN

El modelo computacional está diseñado bajo el paradigma de programación orientada a objetos, programado en el lenguaje C++ con STL (Standard Library Template) en conjunto con la interfaz de paso de mensajes (MPI). Cabe mencionar que el paradigma de programación orientada a objetos sacrifica algo de eficiencia computacional por requerir mayor manejo de recursos computacionales al momento de la ejecución. Pero en contraste, permite mayor flexibilidad a la hora adaptar los códigos a nuevas especificaciones. Además disminuye notoriamente el tiempo invertido en el mantenimiento y búsqueda de errores dentro del código.

6.1.1 Procesamiento paralelo

El procesamiento paralelo se puede definir, en esencia, como la conjunción de dos o más procesadores que realizan un cómputo, simultaneamente en el tiempo, subtareas correspondientes a un problema más grande.

Dentro del procesamiento paralelo se pueden identificar dos ramas o modelos principales, donde la diferencia entre uno y otro reside en la forma en la que cada procesador accede a su memoria (Chaudhuri, 1992). En el primero, llamado modelo de memoria compartida, o conocido también como sistemas multiprocesador, los procesadores que intervienen en el sistema acceden a una única memoria común a todos. En contraste, en el modelo de memoria distribuida, también denominado como sistemas multicomputador o clúster de computadoras, cada procesador posee su propia memoria local. Por tal motivo, si un procesador desea acceder a la memoria local de otro, se requiere de un intercambio de mensajes entre ambos a través de una red que los comunique.

En el modelo de memoria compartida los procesadores se comunican con la memoria a través del bus o sistema de *switches* (llaves) de alta velocidad. Esto les permite lograr un mejor desempeño en comparación con los sistemas de memoria distribuida.

Otra ventaja que presenta este modelo es su uso más eficiente de la memoria, dado que no hay necesidad de replicación de datos. No obstante, este tipo de arquitecturas presentan dos importantes desventajas: el alto costo actual de este tipo de

hardware y la pobre escalabilidad. En contrapartida, los sistemas distribuidos poseen varias ventajas. En primer lugar, este modelo permite desarrollar software más portable debido a los estándares existentes en los protocolos de comunicación.

Por otra parte, los clúster poseen bajo costo y buena escalabilidad dado que usualmente están integrados por computadoras personales interconectadas por una red Ethernet. Por estas razones, la tendencia actual en procesamiento paralelo apunta hacia el empleo de este tipo de sistemas multicomputador (Buyya, 1999) (Buyya, 2000).

6.1.2 Plataforma de hardware y software de desarrollo

Para el desarrollo del software se utilizo el compilador g++ de GNU y la implementación de LAM-MPI que puede ser obtenido de la página Web http://www.lam-mpi.org. Además se utilizó el software MATLAB V7 entorno GNU-Linux, para la generación de mallas de elementos finitos, validación de métodos numéricos y generación de gráficos de difusión.

Las pruebas de ejecución y rendimiento del programa se realizaron en un clúster no homogéneo que está montado en el departamento de Ingeniería Informática de la Universidad de Santiago de Chile. Las características del sistema operativo, compilador y hardware se muestran a continuación:

- Clúster no homogéneo compuesto por 2 nodos de 4 núcleos (Intel Xeon E5405, 2.00 Ghz) y 8 nodos de doble núcleo (AMD Athlon(tm) 64 X2 Dual Core Processor 4200, 2.200 Ghz). Sistema operativo Scientific Linux con el compilador g++ de GNU y LAM-MPI

6.2 ANÁLISIS DE LA SOLUCIÓN PARALELA

En la etapa de diseño de la implementación paralela, se realizo un análisis a partir de versiones seriales del software, donde se estudio principalmente el tiempo de ejecución de los distintos procesos involucrados. El análisis de dividió en dos secciones una enfocada al proceso de elementos finitos y otra al cálculo de una grilla de densidades a partir de la primera aproximación de elementos finitos.

6.2.1 Análisis de la aproximación con elementos finitos

La aproximación con elementos finitos se puede dividir en dos etapas fundamentales. La primera es el cálculo de las geometrías y cálculo de las matrices que componen al sistema de ecuaciones y la segunda corresponde a la resolución del sistema de ecuaciones. En lo que concierne a la solución del sistema de ecuaciones se analizaron tres métodos iterativos Jacobi, Gauss-Seidel y LSQR y dos métodos directos, la factorización LU y la factorización de Cholezky. Para mayor información de los métodos numéricos se puede recurrir a bibliografía disponible (Constanzo, 2006) (Paige, 1982) (Jaramillo, y otros, 2006) (Higham, 2008).

En el método de elementos finitos la fase de resolución del sistema de ecuaciones consume la mayor parte del tiempo de cálculo. La carga de datos, cálculos geométricos asociados al llenado de las matrices que componen el sistema de ecuaciones no influyen mayormente en el tiempo de cómputo de la aproximación (ver figura 6.1). Además, a medida que se aumenta el número de nodos, la solución del sistema de ecuaciones consume cerca del 99% del tiempo total de la aproximación con elementos finitos (ver figura 6.2).

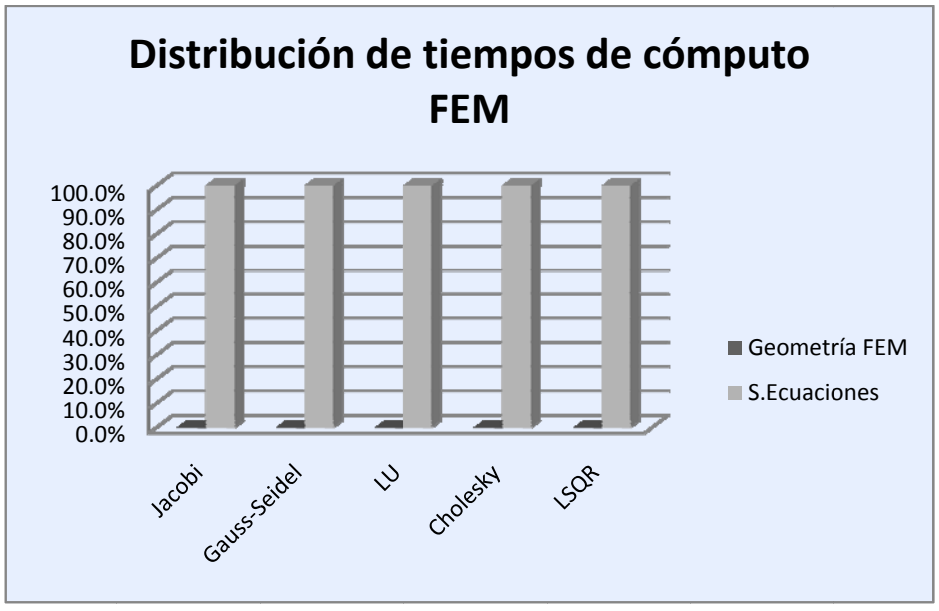


Figura 6.1. *Distribución del consumo de tiempo requerido por el método de elementos finitos en una geometría de 16641 nodos, con distintos métodos de resolución para el sistema de ecuaciones.*

Como se puede observar en el grafico 6.1, en una aplicación de elementos finitos, la paralelización de la carga de datos y cálculos geométricos no tiene sentido y el esfuerzo algorítmico debe ser dirigido a la optimización del proceso de solución del sistema de ecuaciones. Respecto al consumo de memoria, el mayor consumo es debido a las matrices utilizadas en el sistema de ecuaciones, por lo tanto la optimización del cálculo será enfocado al uso optimizado de memoria y al cálculo paralelo de la solución del sistema de ecuaciones.

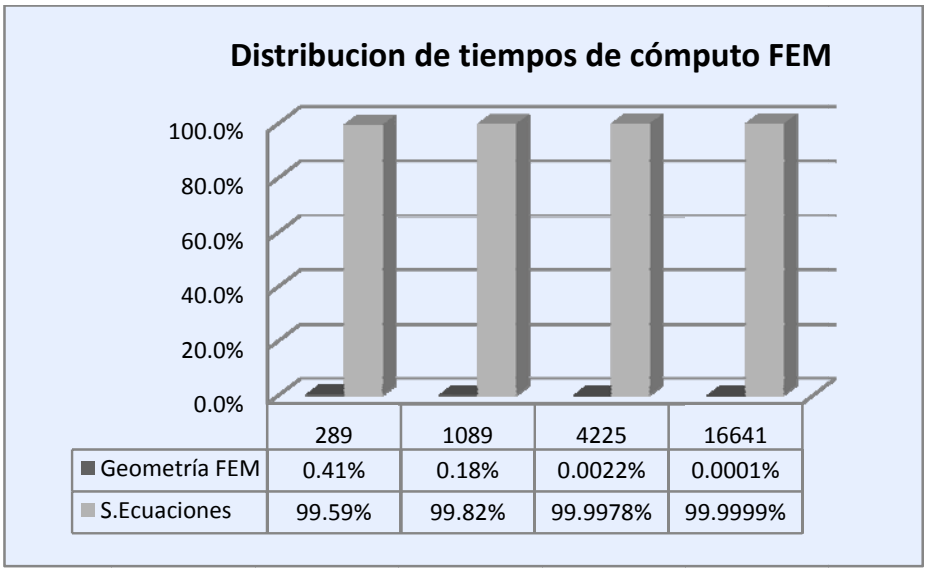


Figura 6.2.- *Distribución promedio (Jacobi, Gauss-Seidel, LU, Cholesky y LSQR) de los tiempos de cálculo en el método FEM, con distintas dimensiones nodales de 289, 1089, 4225 y 16641 nodos respectivamente.*

6.2.2 Análisis del cálculo de la grilla de densidades

Una de las ventajas que tiene el método de elementos finitos es que una vez obtenida una aproximación de en los nodos de la geometría, se puede estimar cualquier valor que pertenezca al dominio de mediante la ecuación 6.1. Esto implica que si ya tenemos una solución del problema pero deseamos por ejemplo estimar valores que pertenecen a , no es necesario realizar un cálculo FEM con nodos y resolver un sistema de ecuaciones de nodos, lo cual requería un uso excesivo de recursos. Para realizar este tipo de aproximación en un elemento finito triangular lineal

de tres nodos, se debe tener a disposición la información geométrica del elemento que contiene a un punto específico, siendo este punto especifico la aproximación que deseamos realizar.

$$u(p) = \sum_{j=1}^{3} \Phi_j \, \psi_j^e(p), \qquad p \in \Omega^e \tag{6.1}$$

Entonces el esfuerzo computacional del proceso de cálculo de una grilla a partir de una aproximación de elementos finitos está en la búsqueda del elemento contenedor de un punto y éste será el proceso que optimizaremos en el cálculo de la grilla. Cabe destacar que el uso de una grilla es relevante si se desean imágenes de mayor resolución en las aproximaciones, o conocer densidades en puntos específicos, como por ejemplo en el borde o cercanos a la fuente.

6.2.3 Análisis de métodos de resolución de sistemas de ecuaciones lineales

A continuación se realizará el análisis de los métodos numéricos utilizados para resolver el sistema de ecuaciones, donde se estudiará la convergencia de los métodos, aplicabilidad de ellos y cuales se pueden adaptar sin mayor dificultad a una solución paralela. Además se tomará en consideración el evitar un consumo excesivo de memoria en el proceso paralelo. Es importante mencionar que el análisis realizado está enfocado a la solución del problema de difusión donde se obtienen matrices cuadradas definidas positivas en el proceso de aproximación con elementos finitos.

6.2.3.1 Análisis de convergencia de los métodos numéricos

En los métodos directos se conoce de antemano el número de operaciones que se realizan para obtener la solución y permiten realizar la factorización directa en lugar de aproximaciones cada vez mejores, como en el caso de los métodos iterativos. En los métodos directos no tiene importancia el criterio de convergencia como sí lo tiene en los métodos iterativos. Entonces el análisis de convergencia se enfoco a los tres métodos iterativos mencionados en la sección 6.2.1 (*Jacobi, Gauss-Seidel y LSQR*). Los tres métodos iterativos convergen a la solución del sistema de ecuaciones. Si bien los tres métodos convergen a una solución, estos difieren en como controlan la convergencia y a

la solución a la cual convergen. Los métodos Jacobi y Gauss-Seidel convergen a $A^{-1}b$ sólo si la matriz del sistema de ecuaciones es estrictamente diagonal dominante. Ambos utilizan un factor de tolerancia de parada, que es obtenido en cada iteración y en función de la variación en la solución que se está aproximando, cuando esta variación es menor o igual a la tolerancia se detiene el proceso de aproximación.

El método LSQR converge a la solución de $min\|Ax - b\|_2$, donde el criterio de parada del ciclo iterativo se usa por lo general un error dado por $\|Ax - b\|_2$. La desventaja del cálculo del error en el método LSQR es que se debe realizar una operación $matriz \times vector$ adicional en cada iteración lo cual aumenta el tiempo de cálculo en caso de utilizar como criterio de parada una tolerancia de error especifica en la solución. Por tal razón y por el costo computacional que ésto conlleva conviene fijar el número de iteraciones como criterio de parada, ya que se asegura la solución exacta en un número de iteraciones igual que la dimensión del sistema. En la figura 6.3 se muestra la convergencia de los tres métodos iterativos para un sistema de 1089 incógnitas.

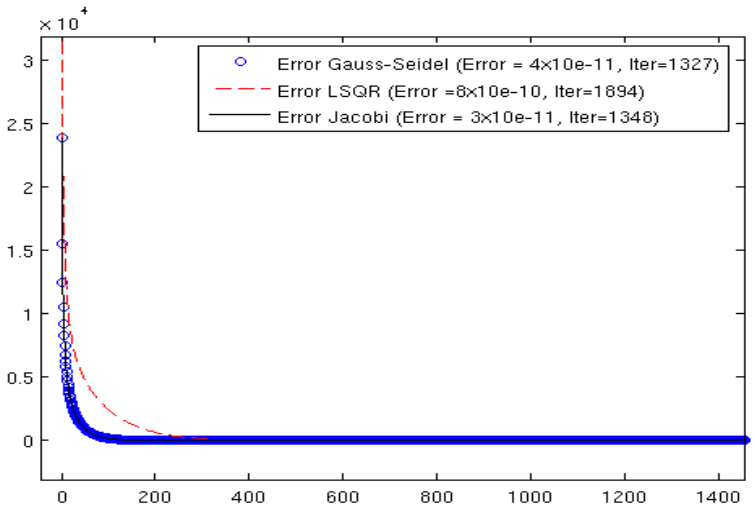
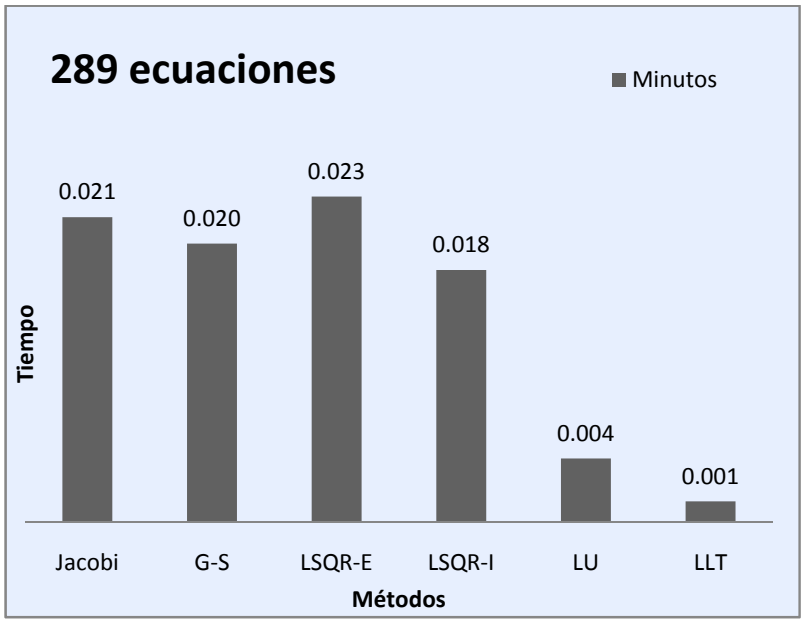


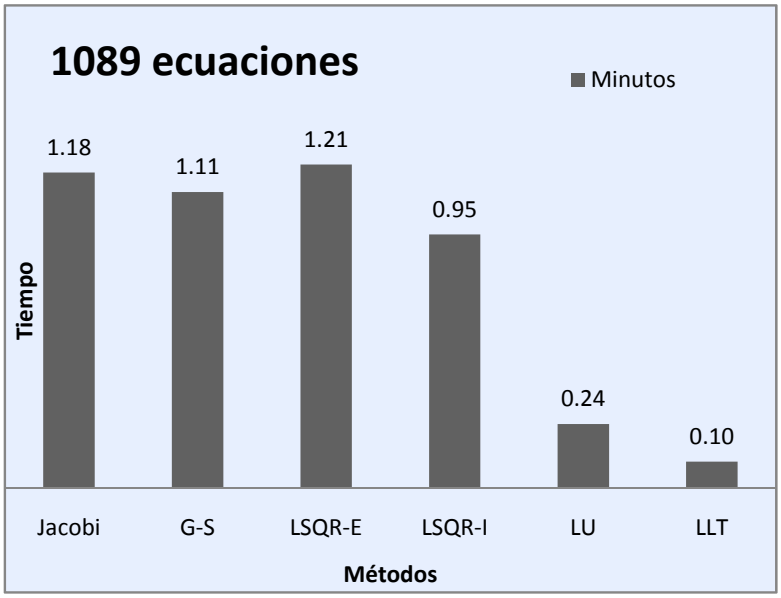
Figura 6.3.- *Convergencia de los métodos Jacobi, Gauss-Seidel y LSQR, donde se muestra el error obtenido utilizando* $\|\boldsymbol{Ax} - \boldsymbol{b}\|_2$ *como medida de error y el número de iteración donde se obtiene el mínimo error.*

6.2.3.2 Tiempos de cómputo de los métodos numéricos

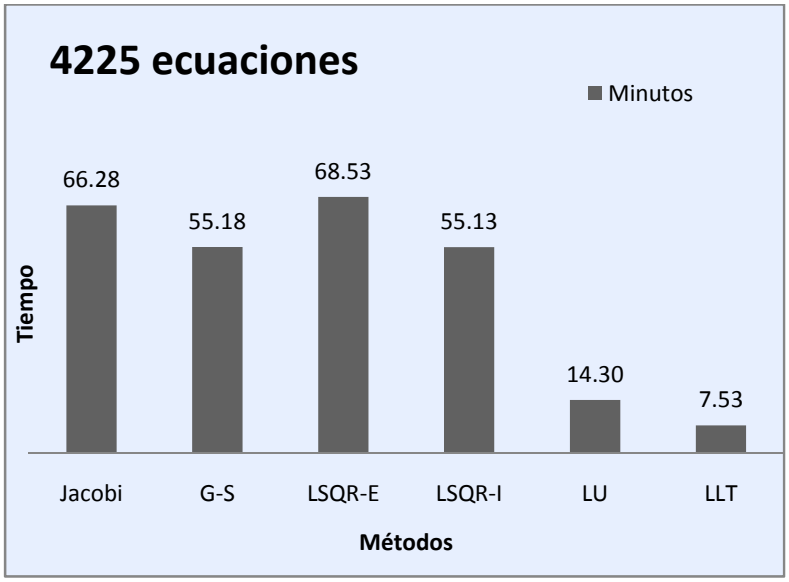
Para el análisis de los tiempos se utilizó el hardware indicado en la sección 6.1.a, donde se realizaron pruebas con sistemas de 289, 1089, 4225 y 16641 ecuaciones. La figura 6.4 muestra los tiempos de cómputo de las versiones seriales de los métodos numéricos analizados.



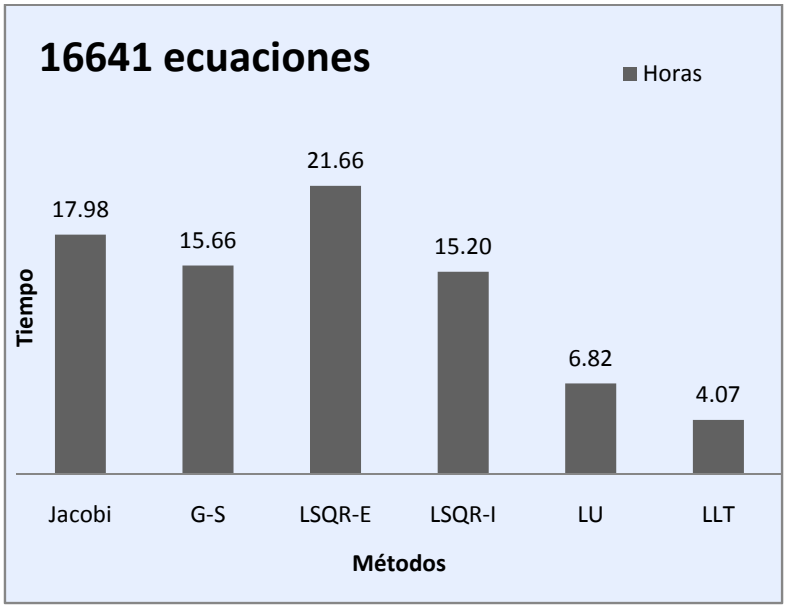
a)



b)



c)



d)

Figura 6.4.- *Tiempos de computo requerido por cada método numérico (Jacobi, Gauss-Seidel "G-S", LSQR con parada en base a error "LSQR-E", LSQR con parada en número de iteraciones igual a la dimensión del sistema "LSQR-I", LU y Cholesky "LLT") que se utilizaron para resolver los distintos sistemas de ecuaciones. a) Ecuaciones = 289, unidad de tiempo = segundos; b) Ecuaciones = 1089, unidad de tiempo = minutos; c) Ecuaciones = 4225, unidad de tiempo = minutos; d) 16641 ecuaciones, unidad de tiempo = horas.*

Cabe destacar que los tiempos obtenidos en los métodos iterativos Jacobi y Gauss-Seidel corresponden a cálculos realizados con una tolerancia de 1×10^{-10}. Si se define como parada una tolerancia mayor como por ejemplo 1×10^{-5} los tiempos de computo son similares al método LU, pero como un objetivo primordial al resolver un sistema de ecuaciones es la precisión de la solución, se consideró la tolerancia 1×10^{-10} en los métodos mencionados, ya que con este valor se pueden evaluar soluciones más homogéneas y los métodos LSQR, LU y LLT tienen un error cercano al 1×10^{-10} el cual solo es obtenido por el método Jacobi y Gauss-Seidel si se utiliza una tolerancia de 1×10^{-10}.

De todos los métodos es fácil deducir que el más rápido es el método Cholesky y el más lento es el método LSQR que utiliza como criterio de parada una tolerancia de error. De los métodos iterativos los más rápidos son el método Gauss – Seidel y el método LSQR con criterio de parada en base a iteraciones, donde el número de iteraciones que se definió fue el número de ecuaciones del sistema.

6.2.3.3 Consumo de memoria de los métodos numéricos

El uso principal de memoria de los métodos numéricos se centra principalmente en la reserva de memoria necesaria para almacenar las matrices y cómo estos métodos utilizan la información de las matrices. Por lo cual en problemas de gran escala la reserva de memoria pasa a ser una de las problemáticas principales al momento de diseñar una solución computacional si no se disponen de arquitecturas de hardware capaces de soportar el volumen de información utilizado para resolver el problema.

Entonces el uso de memoria dependerá exclusivamente de cómo se almacena y usa la información de las matrices. Las matrices a nivel de distribución y volumen de datos se pueden definir como densas o dispersas. Las matrices densas son las que tienen muy pocos valores no nulos en la matriz, y las matrices dispersas son las que por lo general tienen más valores nulos que no nulos.

Las matrices utilizadas en el método de elementos finitos se caracterizan por ser simétricas y a medida que se aumenta el número de nodos las matrices pasan a ser altamente dispersas, por lo cual esta característica debe ser aprovechada en el modelamiento de un método numérico de solución de sistemas de ecuaciones. Por lo cual esta característica de matriz altamente dispersa será tomada en consideración

cuando se decida que método numérico será utilizado y de esta manera solo reservar la memoria necesaria para almacenar la matriz.

Ahora si nos referimos a la memoria utilizada en los métodos directos e iterativos, en el caso de los métodos directos, si bien inicialmente se puede tener almacenada la matriz en un formato disperso, después de la descomposición de la matriz original, pasa a ser una matriz con más elementos no nulos y en muchos de los casos altamente densa, por lo tanto el consumo de memoria después de los procesos de factorización son muy similares a la de la matriz original.

6.2.3.4 Elección de método numérico

Para la elección del método numérico se tomó en consideración el consumo de memoria requerida por las matrices que componen al sistema de ecuaciones del método FEM, la optimización de los cálculos y la comunicación en los procesos paralelos de los cálculos matriciales.

Los métodos directos, basados en la descomposición de la matriz A, son técnicas que se caracterizan por su robustez, puesto que se puede predecir el tiempo (cantidad de operaciones en punto-flotante) y el almacenamiento que requieren. Sin embargo, para la solución de problemas a gran escala los métodos directos llegan a ser poco o nada prácticos, incluso sobre plataformas en paralelo.

Para calcular la descomposición es necesario almacenar N^2 entradas y realizar $O(N^3)$ operaciones en punto-flotante y por lo general la matriz queda con más elementos no nulos que la matriz original y además dependiendo del condicionamiento de la matriz se puede propagar el error a medida que se aumenta la dimensión de la matriz (Jaramillo, y otros, 2006).

En los procesos paralelos el problema principal al utilizar métodos directos es la optimización de los procesos de comunicación y el volumen de información utilizada. Entonces si el objetivo es optimizar el uso de memoria las técnicas iterativas son más idóneas para su implementación en paralelo.

Si nos referimos a las técnicas iterativas analizadas (Jacobi, Gauss-Seidel y LSQR (*Least-Square QR*)), en los tres métodos se puede utilizar un almacenamiento disperso de la matriz A, la diferencia radica en cómo utilizan la información y que datos son prescindibles en cada proceso.

En el caso de los métodos Jacobi y Gauss-Seidel, la matriz principal del sistema ecuaciones se puede almacenar de forma dispersa. Jacobi tiene una implementación paralela trivial ya que todos los elementos de la matriz se pueden paralelizar concurrentemente si se utiliza la matriz completa o bien en formato disperso pero con algún procedimiento que identifique las filas vecinas de cada partición, en cambio Gauss – Seidel necesita los valores de la iteración anterior lo cual no permite una actualización paralela de los elementos de forma eficiente. Pero aún mas importante es que la convergencia de los dos métodos es solo asegurada si la matriz es diagonalmente dominante.

El método LSQR es equivalente al método de gradiente conjugado GC, sobre la base del proceso de "bidiagonalización de Lanczos" y el procedimiento de Golub y Kahan, pero es numéricamente mejor que GC y soporta sistemas no simétricos del tipo $Ax = b$ realizando transformaciones de la matriz a una ecuación normal $A^tAx = A^tb$, donde A^tA es una matriz simétrica definida positiva. Además se asegura la solución exacta en un número de iteraciones igual que al rango de la matriz del sistema. LSQR converge a $min\|Ax = b\|_2$ usando la descomposición QR de la matriz A (Paige, 1982).

La paralelización del método LSQR se enfoca a las multiplicaciones matriz-vector implícitas en el proceso de bidiagonalización del método y estos procesos paralelos si bien requieren un mayor costo a nivel de comunicación ya que en cada iteración existe una actualización de los cálculos, el resto del algoritmo es rápido tratado de forma lineal.

Por las características de estabilidad en la convergencia, la viabilidad de utilizar un almacenamiento disperso y principalmente por su elegancia a nivel matemático, el método numérico seleccionado a paralelizar será el método LSQR y si bien es el más lento en su versión en base a parada con una tolerancia de error, el método asegura convergencia cuando el número de iteraciones igual a la dimensión del sistema de ecuaciones. Además el método puede ser escalable a otro tipo de problemas matriciales, donde los métodos Jacobi y Gauss – Seidel no aseguran convergencia y se restringen a ciertos tipo de matrices.

6.3 OPTIMIZACIÓN DEL CONSUMO DE MEMORIA Y MODELADO DE MÓDULOS PARALELOS

A continuación se mostrará cómo se optimizó el uso de memoria en el algoritmo de cómputo de las matrices construidas por el método de elementos finitos. Además se modelarán los módulos paralelos del algoritmo LSQR y cálculo de la grilla de densidades.

6.3.1 Optimización del uso de memoria

La matriz de rigidez $A = K + C + B$, que es construida a partir de los aportes elementales $k_{ij,}^{e}$ c_{ij}^{e} y b_{ij}^{e} obtenidos del cálculo integral de los elementos e forman una matriz dispersa que a medida que se aumenta el número de nodos, el número de valores nulos en la matriz aumenta considerablemente, por lo cual si queremos optimizar el uso de memoria debemos utilizar un método de llenado y almacenamiento que sólo utilice la memoria necesaria para almacenar valores no nulos. Además, si en este proceso ya tenemos un almacenamiento de la matriz dispersa óptimo en función del uso de memoria, la matriz de entrada A al término del proceso de elementos finitos de llenado de matrices, no será necesaria una conversión de una matriz $N \times N$ a formato disperso que es necesaria para la solución del sistema de ecuaciones con el método LSQR.

El problema de llenado disperso de la matriz de rigidez A está en que debemos saber a priori la memoria necesaria de A y si bien esto se puede obtener en el proceso de carga de datos, es necesaria la reserva de memoria de un tamaño fijo y esto no se sabe hasta el término de la carga de información. Por lo cual la utilización de memoria dinámica a medida que se almacena la información es requerida en la implementación.

Para la implementación de este modulo se utilizo una matriz asociativa de tipo <map> que es una plantilla de la librería STL de C++. Las plantillas tipo <map> son mapas que están diseñados para ser especialmente eficientes en el acceso a sus elementos a través de sus claves y se pueden acceder directamente a los elementos con el $Map[clave]$, donde la búsqueda de un elemento es de orden $O(log\ n)$ y la inserción de un nuevo elemento $O(log\ n)$, además permite el uso memoria dinámica y esta aumenta a medida que agregan nuevos valores asociados a una llave distinta a las almacenadas.

En la implementación se utilizó una plantilla que tiene como clave la posición ij de la matriz A en formato vector $i * (\text{numero nodos}) + j$, y a medida que se retorna la posición global de un aporte nodal de los elementos, se van sumando estos aportes directamente al mapeo en $MapA[i * (\text{numero nodos}) + j]$ y si no existe se agrega un nuevo elemento con la clave $i * (\text{numero nodos}) + j$ con el aporte nodal correspondiente, además se almacenan en dos mapeos secundarios $MapFila$ y $MapColumna$ los cuales almacenan la fila y columna de pertenencia de la matriz A_{ij}.

Como la matriz A está compuesta por la suma $K + C + B$, en cada proceso de llenado se suman los aportes en las mismas ubicaciones $i * (\text{numero nodos}) + j$ de $MapA$. En la figura 6.5 se muestra el algoritmo de llenado la matriz en formato <map>.

La otra utilidad significativa del uso de este tipo de almacenamiento, está en el proceso de reducción del sistema de ecuaciones, donde para resolver el sistema reducido en caso de utilizar la condición de borde de tipo Dirichlet $\partial\Omega = 0$, en donde se asume que los nodos que están en el borde valen 0 se deben eliminar las filas y columnas i, j en que los nodos $i, j \in \partial\Omega$, además de eliminar del vector de cargas los términos q_i, este proceso se realiza simplemente eliminando de la memoria los elementos del mapeo $MapA[clave]$ que tengan la clave asociada a los mapeos $MapFila[clave] = i \; o \; j$ y $MapColumna[clave] = i \; o \; j$, con lo cual solo queda almacenada la matriz dispersa sin las filas y columnas que contienen información del borde.

Ciclo 1: *Mientras no se evalúen todos los elementos* e**:**
Retornar Matrices y almacenar temporalmente p_{ij}^e**,** k_{ij}^e **,** m_{ij}^e **y** b_{ij}^e
/***** p_{ij}^e *es la numeración nodal global de los nodos del elemento e */*
Ciclo 2: Recorrer p_{ij}^e / $i, j = 1,2,3$
$I = p_{ji}^e, J = p_{ij}^e$
$MapA[I * (\text{numero nodos}) + J]$ += $k_{ij}^e + m_{ij}^e + b_{ij}^e$
$MapFila[I * (\text{numero nodos}) + J] = i$
$MapColumna[I * (\text{numero nodos}) + J] = j$
Fin Ciclo 2
Fin Ciclo 1

Figura 6.5.- *Algoritmo de llenado de la matriz de rigidez A, además de las filas y columnas donde se realizan aportes a la matriz A utilizando la plantilla <Map>.*

6.3.2 Paralelización del método LSQR (Least-Square QR)

En la implementación paralela del método LSQR se paralelizó el proceso de cómputo de productos matriz × vector en formato disperso, el cual se usa en cuatro secciones del algoritmo: la primera es al inicio del algoritmo las siguientes se computan en el proceso iterativo del algoritmo LSQR donde solo realizan dos multiplicaciones por ciclo en caso de parada en número de iteraciones y tres en multiplicaciones por ciclo en caso de parada utilizando una tolerancia de error donde se calcula $\|Ax = b\|_2$. El método LSQR en su versión serial se puede observar en la figura 6.6 y en las figuras 6.7 y 6.8 están los Algoritmos de las versiones paralelas del método LSQR.

En las versiones paralelas el mayor costo de cómputo y comunicación es en el proceso de bidiagonalización, por lo tanto en los algoritmos paralelos (Figuras 6.7 y 6.8) nos enfocaremos principalmente en ese proceso.

Paso 1: Inicializar (Resolver sistema Ax=b)

De las relaciones $\beta_1 u_1 = b, \quad \alpha_1 v_1 = A^T u_1$

Calcular $\beta_1 = \|b\|, \; u_1 = b/\beta_1, \; \alpha_1 = \|(A^T u_1)\|, \; v_i = (A^T u_1)/\alpha_1$

Asignar $w_1 = v_1, \; x_0 = 0, \quad \bar{\phi}_1 = \beta_1, \quad \bar{\rho}_1 = \alpha_1$

Ciclo 1: (i = 1,2,3, ... repetir pasos $2 - 5$ hasta criterio de parada)

Paso 2: Bidiagonalización

a) /* $\beta_{i+1} u_{i+1} = A v_i - \alpha_i u_i$ */

Calcular $\beta_{i+1} = \|(A v_i - \alpha_i u_i)\|, \quad u_{i+1} = (A v_i - \alpha_i u_i)/\beta_{i+1}$

b) /* $\alpha_{i+1} v_{i+1} = A^T u_{i+1} - \beta_{i+1} v_i$ */

Calcular $\alpha_{i+1} = \|(A^T u_{i+1} - \beta_{i+1} v_i)\|, \quad v_{i+1} = (A^T u_{i+1} - \beta_{i+1} v_i)/\alpha_{i+1}$

Paso 3: Construir y aplicar la siguiente transformación ortogonal

a) $\rho_i = \sqrt{\bar{\rho}_i^{\,2} + \beta_{i+1}^2}$

b) $c_i = \bar{\rho}_i / \rho_i$

c) $s_i = \beta_{i+1} / \rho_i$

d) $\theta_{i+1} = s_i \alpha_{i+1}$

e) $\bar{\rho}_{i+1} = -c_i \alpha_{i+1}$

f) $\phi_1 = c_i \bar{\phi}_i$

g) $\bar{\phi}_{i+1} = s_i \bar{\phi}_i$

Paso 4: Actualizar x, w

a) $x_i = x_{i+1} + (\phi_i/\rho_i)w_i$

b) $w_{i+1} = v_{i+1} + (\bar{\phi}_{i+1}/\rho_i)w_i$

Paso 5: Test de convergencia (solo en caso de utilizar como parada un error)

Fin Ciclo 1

Paso 6: Exportar solución

Figura 6.6.- *Algoritmo serial del método LSQR* (Paige, 1982)

6.3.2.1 Algoritmo paralelo con MPI_Gatherv y MPI_Bcast

En la propuesta algorítmica presentada en la figura 6.7 después de cada proceso de multiplicación se debe recolectar la información calculada por todos los procesadores. Para tal efecto se utilizó la instrucción MPI_Gatherv(), la cual realiza una recolección de datos en un procesador especifico. Esto es útil en dos aspectos: no es necesario esperar un orden de entrada de los procesos y se puede almacenar en y desde cualquier posición a tamaños variables de los vectores, lo cual es necesario cuando el número de datos de las particiones de un vector no es igual en todos ellos como ocurre por lo general en un vector que representa una sección de una matriz dispersa.

Después de cada producto matriz × vector se realiza la recolección en el proceso root. Este proceso normaliza los vectores y como paso siguiente se realiza un broadcast, para que todos los procesos tengan los vectores u_{i+1} y v_{i+1} necesarios para la siguiente iteración. Los pasos 3 y 4 del algoritmo LSQR los realiza el proceso root. Cabe destacar que hay una instrucción en MPI que realiza esta dos operaciones colectivas MPI_Allgatherv() pero los tiempos en comunicación obtenidos fueron similares con ambas instrucciones por lo cual el uso de uno u otro es irrelevante en el estudio del algoritmo visto en la figura 6.7.

La desventaja principal de esta propuesta es que se realizan 4 comunicaciones colectivas en el ciclo iterativo y a medida que se aumenta el número de procesadores si bien disminuye el tamaño de los vectores el número de mensajes aumenta. La ventaja de esta propuesta paralela radica en su fácil implementación, principalmente porque la multiplicación paralela de una matriz por vector es trivial, ya que se realizan multiplicaciones entre secciones de filas de la matriz por un vector y se almacena el resultado en un vector que será enviado al procesador root en el proceso de recolección y de esta forma construir el vector resultado global de la multiplicación matriz por

vector. En el algoritmo de la figura 6.7 el vector de salida de la multiplicación es igual al número de filas de la partición y este tamaño es fijo en todas las multiplicaciones particionadas del método LSQR. En la figura 6.8 se muestra un esquema general del ciclo iterativo del algoritmo LSQR paralelo utilizando la instrucción MPI_Gatherv() y MPI_Bcast().

Paso 1: Inicializar (Resolver sistema Ax=b)

El proceso root realiza: $\beta_1 = \|b\|,\ u_1 = b/\beta_1$

Producto paralelo de $v_1 = A^T u_1$ *y* ***Gatherv*** *en* v_1

El proceso root realiza todo el proceso de normalización de v_1

El proceso root realiza $w_1 = v_1,\ x_0 = 0,\ \bar{\phi}_1 = \beta_1,\ \bar{\rho}_1 = \alpha_1$

Broadcast *de* v_1

Ciclo 1: ($i = 1,2,3,\ldots$ repetir pasos $2-5$ hasta criterio de parada)

Paso 2: Bidiagonalización

a) /* $\beta_{i+1}u_{i+1} = Av_i - \alpha_i u_i$ */

Producto paralelo de $u_{i+1} = Av_i$ *y* ***Gatherv*** *en* u_{i+1}

El proceso root realiza la normalización de u_{i+1}

Broadcast *de* u_{i+1}

b) /* $\alpha_{i+1}v_{i+1} = A^T u_{i+1} - \beta_{i+1}v_i$ */

Producto paralelo de $v_{i+1} = A^T u_{i+1}$ *y* ***Gatherv*** *en* v_{i+1}

El proceso root realiza todo el proceso de la normalización de v_{i+1}

Broadcast *de* v_{i+1}

Paso 3: Construir y aplicar la siguiente transformación ortogonal

El proceso root realiza este paso

Paso 4: Actualizar x, w

El proceso root solo realiza este paso

Paso 5: Test de convergencia (solo en caso de utilizar como parada un error)

Producto paralelo y ***Gatherv*** *al proceso root en caso de parada mensaje a todos los procesadores que finalicen ciclo iterativo*

Fin Ciclo 1

Paso 6: *Exportar solución del proceso root*

Figura 6.7.- *Algoritmo paralelo del método LSQR utilizando recolección de vectores y difusión de los vectores recolectados.*

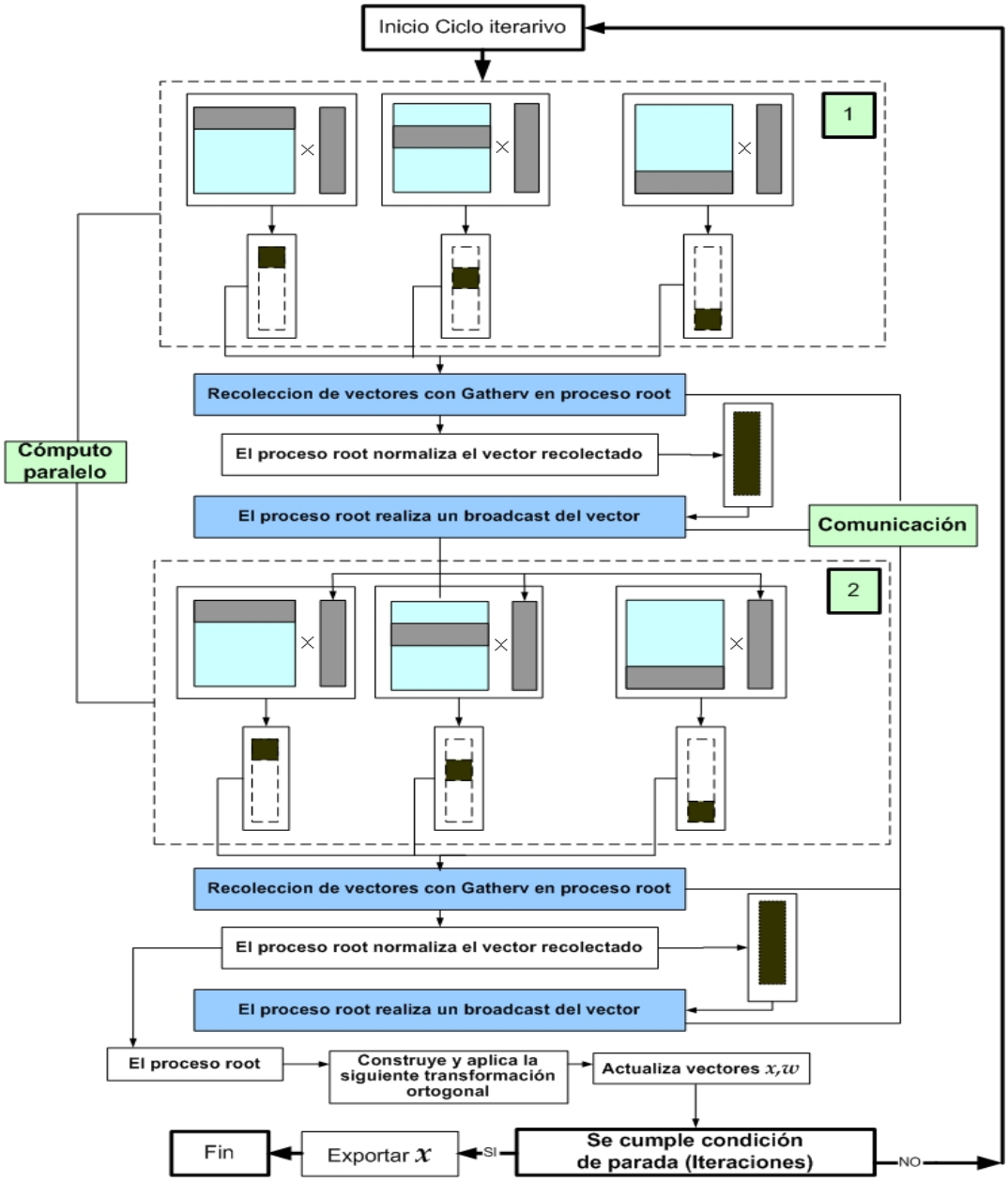


Figura 6.8.- *Esquema de funcionamiento del ciclo iterativo del algoritmo paralelo utilizando 4 comunicaciones colectivas (2 Gatherv y 2 Broadcast) y donde 1) y 2) son los productos de matriz - vector en paralelo.*

6.3.2.2 Algoritmo paralelo con MPI_Allreduce

En el algoritmo con MPI_Allreduce se utilizan dos dimensiones distintas para los vectores x que se calculan en el proceso iterativo, la primera es del tamaño del número de filas de la partición de la matriz A que coincide con la dimensión original de x,

entonces $\dim(x_p) = (\dim(x)/procesos) + resto$ donde solo se suma $resto$ en el último proceso p. La segunda es de tamaño igual al número de ecuaciones del sistema $\dim(x_p) = \dim(x)$. Además se utilizan tres tipos de multiplicaciones matriz × vector en el proceso iterativo. La primera es una multiplicación de la matriz particionada en $\dim(A_p) = ((\dim(x)/procesos) + resto) \times \dim(x)$ por un vector de dimensión $\dim(x)$ y el vector resultado es de dimensión $(\dim(x)/procesos) + resto)$ (ver Figura 6.9.a). La segunda multiplicación es de una matriz de dimensión $\dim(A_p) = \dim(x) \times ((\dim(x)/procesos) + resto)$ ya que se utiliza la matriz traspuesta y se multiplica por un vector de dimensión $(\dim(x)/procesos) + resto)$ y el vector resultado es de dimensión $\dim(x)$ (ver Figura 6.9.b). La tercera es idéntica a la primera pero los valores de salida se almacenan en un vector de dimensión $\dim(x)$ donde las posiciones que no han sido calculadas tienen valor 0 (ver Figura 6.9.c). El detalle del algoritmo paralelo con MPI_Allreduce se puede ver en la figura 6.10 y en la figura 6.11 se muestra un esquema general del ciclo iterativo del algoritmo LSQR paralelo utilizando la instrucción MPI_Allreduce.

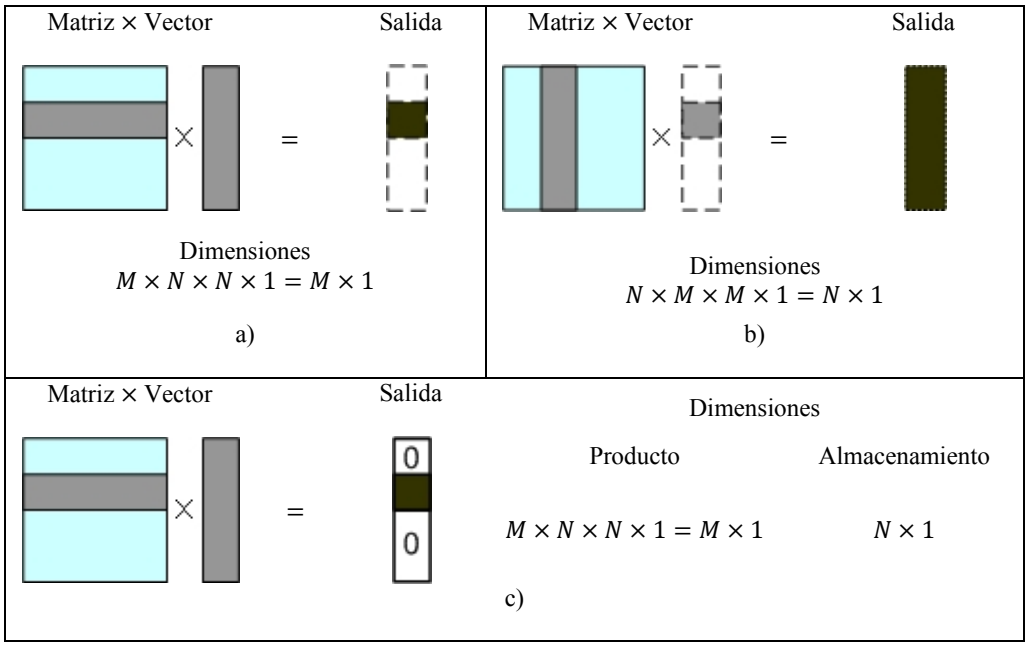


Figura 6.9.- *Tipos de productos de secciones de una matriz A de dimensión $N \times N$ por un vector: a) multiplicación de una sección de la matriz de dimensión $M \times N$ por un vector de dimensión $N \times 1$, b) multiplicación de una sección de la matriz de dimensión $N \times M$ por un vector de dimensión $M \times 1$, c) multiplicación de una sección de la matriz de dimensión $M \times N$ por un vector de dimensión $N \times 1$ donde el vector resultado se almacena en una sección de un vector de dimensión $N \times 1$ en que el resto de los valores es 0.*

Paso 1: *Inicializar (Resolver sistema Ax=b)*

*/*Obs:* $\dim(x) = \text{número de ecuaciones} */$

Todos los procesos realizan: $\beta_1 = \|b\|,\ u_1 = b/\beta_1$

Cada proceso genera un vector U_1 *que es una sección de* u_1

Producto paralelo de $v_1 = A^T u_1$ *donde* v_1 *es de dimensión* $\dim(x)$

/ Obs:* v_1 *solo es llenada en posiciones de la partición el resto vale 0 */*

Allreduce(*SUM,* $\dim(x), v_1$***)***

/ Obs: En Allreduce(SUM,* $\dim(x), v_1$*) todos suman los vectores de todos los procesos y obtienen* v_1 *de dimensión* $\dim(x)$ **/*

Todos los procesos realizan los pasos para la normalización de v_1

Todos los procesos realizan $w_1 = v_1,\ x_0 = 0,\ \bar{\phi}_1 = \beta_1,\ \bar{\rho}_1 = \alpha_1$

Ciclo 1: ($i = 1,2,3, \ldots repetir\ pasos\ 2-5\ hasta\ criterio\ de\ parada$)

Paso 2: *Bidiagonalización*

a) /* $\beta_{i+1}u_{i+1} = Av_i - \alpha_i u_i$ */*

Producto paralelo de $U_{i+1} = Av_i$

Cada proceso asigna su partición $U_{i+1} = U_{i+1} - \alpha_i U_i$

Cada proceso calcula el producto punto PP de U_{i+1}

Allreduce(*SUM,NP,PP****)*** */* NP=Numero de procesos*/*

/ Obs: En* ***Allreduce****(SUM,* NP, PP*) todos suman los valores de PP de todos los procesos y obtienen el producto punto del vector* u_{i+1} *de dimensión* $\dim(x)$ **/*

Cada proceso calcula la raíz de SUM y se obtiene β_{i+1}

Cada proceso normaliza su sección U_{i+1}

b) /* $\alpha_{i+1}v_{i+1} = A^T u_{i+1} - \beta_{i+1}v_i$ */*

Cada proceso calcula $v_{i+1} = A^T U_{i+1}$ *donde* v_{i+1} *es de dimensión* $\dim(x)$

*/*Obs:* A^T *es explícitamente la traspuesta de la partición de A */*

Allreduce(*SUM,* $\dim(x)$*,* v_{i+1}***)***

Cada proceso realiza todos los pasos para la normalización de v_{i+1}

Paso 3: *Construir y aplicar la siguiente transformación ortogonal*

Todos los procesos realizan este paso

Paso 4: *Actualizar* x, w

Todos los procesos realizan este paso

Paso 5: *Test de convergencia (solo en caso de utilizar como parada un error)*

Producto paralelo y ***Allreduce*** *al proceso root y en caso de parada se envía mensaje a todos los procesos que finalicen ciclo iterativo*

Fin Ciclo 1

Paso 6: *Exportar solución*

Figura 6.10.- *Algoritmo paralelo utilizando MPI_Allreduce.*

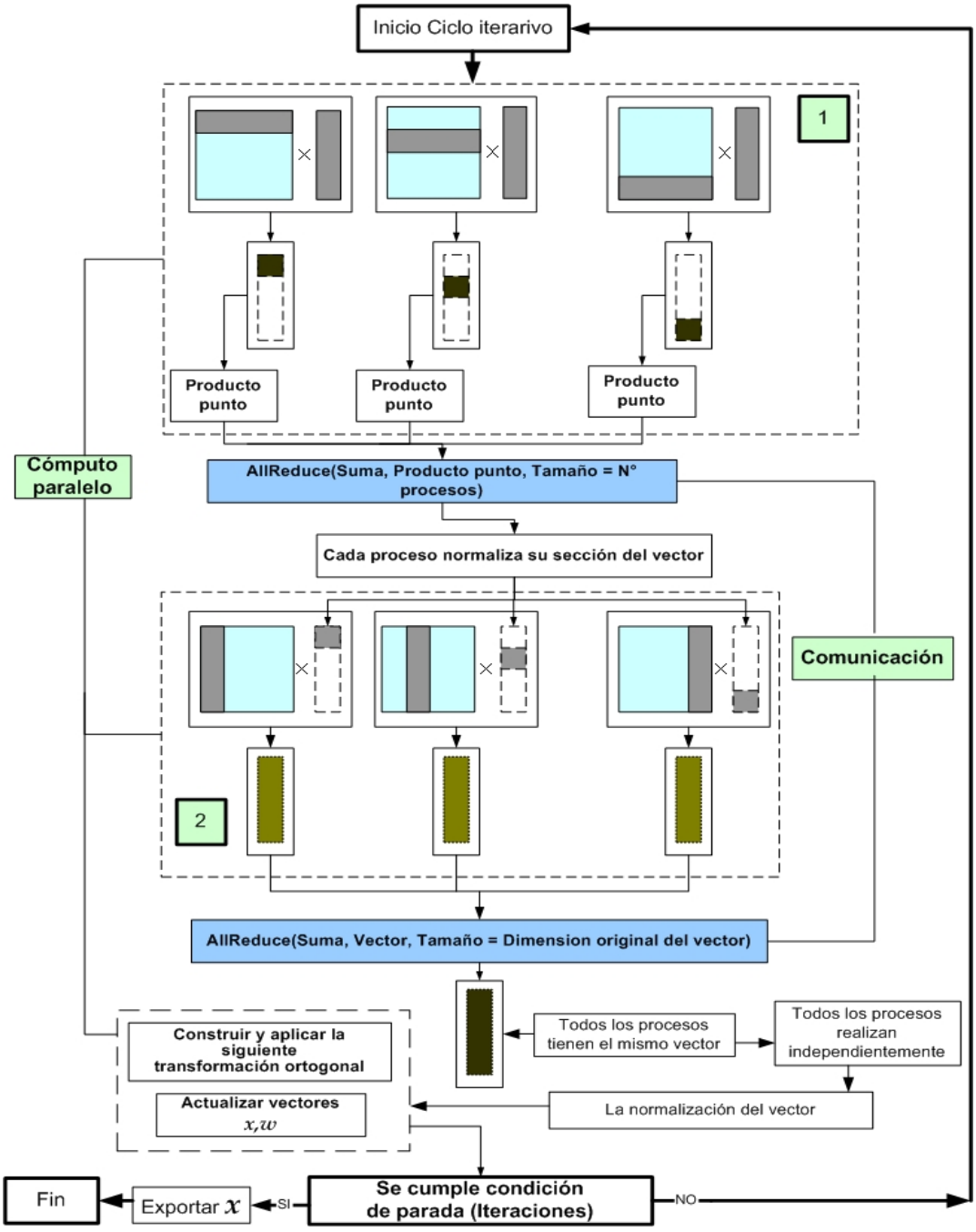


Figura 6.11.- *Esquema de funcionamiento del ciclo iterativo del algoritmo paralelo utilizando 2 comunicaciones colectivas AllReduce: 1) es el cómputo paralelo de* Av_i *y el posterior cálculo del producto punto de cada subvector resultante para luego poder normalizar cada sección de* U_{i+1} *necesario en el siguiente producto del algoritmo LSQR, 2) es el cómputo paralelo de* $A^T U_{i+1}$ *en que una vez sumado todos los vectores se obtiene* v_{i+1}.

El algoritmo de la figura 6.10 en el proceso iterativo con condición de parada en número de iteraciones tiene 2 comunicaciones. La primera comunicación es utilizada para el cálculo de la norma en paralelo, donde el tamaño de los mensajes es de largo 1 y todos los procesadores reciben la información de todos. La segunda comunicación es cuando todos reciben los vectores de dimensión $\dim(x)$ de los otros procesos y se suman, con lo cual se obtiene el producto de la matriz traspuesta por un vector.

Cuando en el algoritmo de la figura 6.10 se comenta que de forma explícita se realiza el producto con la matriz dispersa, es porque en el proceso del producto de la matriz particionada por un vector se intercambian las filas por columnas para que se obtenga la dimensión original en el vector a calcular y no un vector de salida particionado. Cabe destacar que si bien existe una multiplicación de una matriz traspuesta por un vector al inicio del algoritmo, no es necesario realizar un transposición de la matriz, ya que la matriz que se utiliza este problema y para la cual se diseño el método utiliza matrices simétricas por lo cual no es necesario multiplicar por una matriz traspuesta si lo que se desea es obtener un vector resultado particionado.

6.3.3 Paralelización del cálculo de una grilla de densidades

La paralelización del cálculo de una grilla G de I puntos que pertenecen al dominio Ω^{G}, consiste en distribuir las búsquedas de los elementos e que contienen a un punto p en su dominio Ω^{e} y realizar la aproximación lineal utilizando la formulación 6.2.

$$u(p) = \sum_{\mathrm{j}=1}^{3} \Phi_j^e \, \psi_j^e(p), \qquad p \in \Omega^{\mathrm{e}} \in \Omega^{\mathrm{E}} \tag{6.2}$$

Donde $\Omega^{\mathrm{e}} \in \Omega^{\mathrm{E}} \in \Omega^{\mathrm{G}}$, Ω^{e} es el dominio del elemento, Ω^{E} es el dominio global compuesto por la unión de todos los dominios Ω_l^e $(l = 1,2, \dots M;\ M$ es el número de elementos), Ω^{G} es el dominio de la grilla. Φ_j^e la densidad nodal j del elemento e, $\psi_j^e(p)$ es la función de base nodal j del elemento e evaluada en el punto $p \in \Omega^{\mathrm{G}}$ y $u(p)$ es la aproximación que se obtiene al evaluar el punto p, y sólo es calculada si y solo si el punto $p \in \Omega^{\mathrm{E}}$ en caso contrario su valor es 0. En la figura 6.12 se muestra una grilla cartesiana que es el dominio Ω^G de la cual se obtienen los I puntos de la grilla G, un

esquema de la distribución de los distintos dominios y un esquema de los puntos I de la grilla G sobre el dominio de elementos finitos Ω^E.

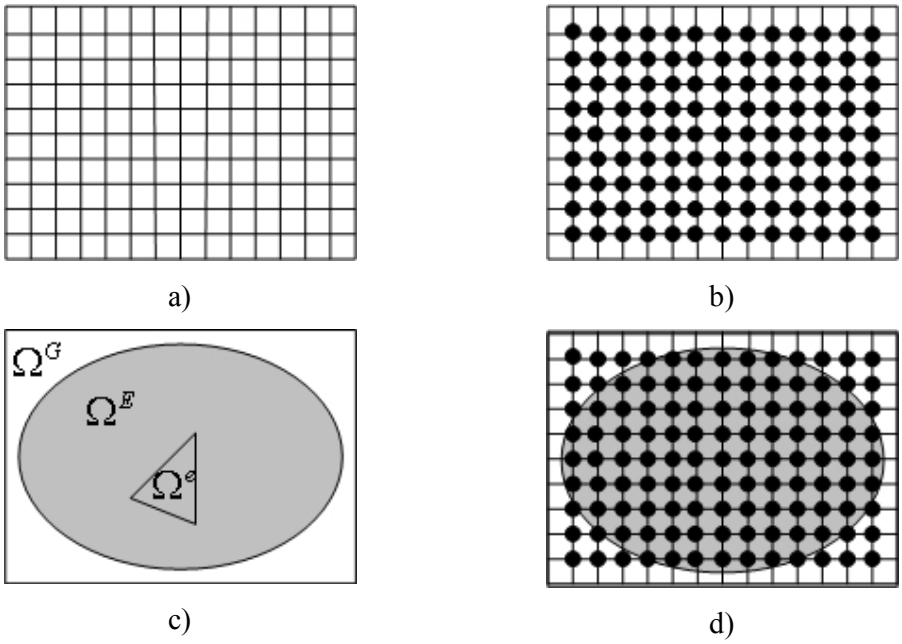


Figura 6.12.- *Esquemas de la grilla y dominios de la geometría: a) dominio de la grilla Ω^G, b) grilla G de I puntos $\in \Omega^G$, c) esquema de distribución de dominios, d) dominio Ω^E contenido en la grilla $G \in \Omega^G$.*

6.3.3.1 Distribución de búsquedas

Para encontrar el elemento e que contiene a un punto p, la búsqueda se distribuye particionando los dominios de Ω^G y Ω^E. La partición de Ω^G tiene como objetivo distribuir las búsquedas de los puntos p_i $(i = 1,2 \dots I)$ en P procesadores, donde I es el número total de puntos de la grilla G. La partición de Ω^E se realiza con el objetivo de que las búsquedas solo se realicen en los elementos e que están contenidos en los subdominios $\Omega_k^g \in \Omega^G$ $(k = 1,2 \dots P)$, donde P es el numero de procesadores.

A cada partición Ω_k^g de Ω^G se le calcula el centroide CT_k (ver figura 6.13) el cual es usado como un punto de referencia para obtener la pertenencia de los elementos e, a una partición k. La asignación de pertenencias k es solo en base a los puntos nodales q $(q = 1,2, \dots N)$ donde N es el numero de nodos, en donde en un proceso anterior al

cálculo de las aproximaciones $u(p)$ se le asigna la pertenencia k a cada punto nodal del dominio Ω^E. La asignación de la pertenencia k a un punto nodal, se realiza buscando cual es el centroide CT_k que tiene la distancia mínima al nodo q.

Esto implica que los elementos e también tendrán una dependencia a k procesador y solo la información de los elementos que contienen nodos que pertenecen al dominio Ω_k^g es utilizada por cada proceso k en la búsqueda de un elemento contenedor de un punto $p_i \in \Omega_k^g$. En el caso de que elementos estén contenidos en más de un dominio Ω_k^g como puede suceder en elementos que se encuentren en las fronteras de los dominios Ω_k^g, simplemente esta información está disponible en todos los procesos que comparten uno o más dominios en común.

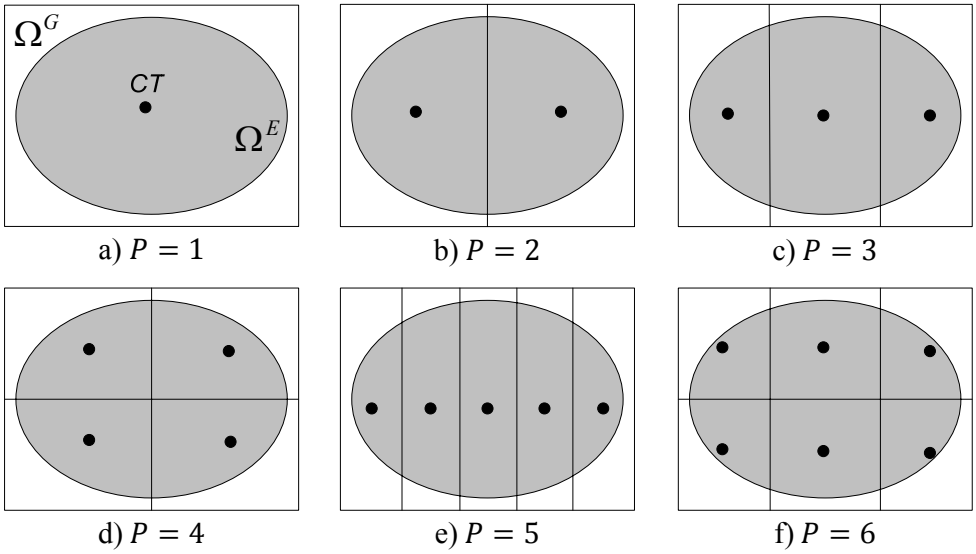


Figura 6.13.- *División del dominio de la grilla Ω^G según numero de procesos P, donde a cada dominio se le calcula su centro geométrico CT.*

6.3.3.2 Determinación de pertenencia de un punto en un triangulo

Para determinar si un punto p está dentro de un elemento finito triangular de tres nodos, se utiliza un procedimiento basado en el concepto de orientación del elemento finito triangular y la orientación de los tres sub triángulos formados por el punto p y dos de los nodos que formar al elemento finito triangular. La orientación de cada triángulo se determina de acuerdo a la dirección del movimiento cuando se visitan los vértices en un

orden específico, utilizando la numeración local de los nodos del elemento finito triangular. Dado un triángulo T compuesto por los vértices (V_1, V_2, V_3) y un punto p del plano, el punto p está en el interior de T si la orientación de los triángulos $T_1(V_1, V_2, p)$, $T_2(V_2, V_3, p)$ y $T_3(V_3, V_1, p)$ es la misma que la orientación del triángulo $T(V_1, V_2, V_3)$. En la figura 6.14 se utiliza el concepto de orientación de triángulos para determinación de pertenencia de un punto a un triángulo.

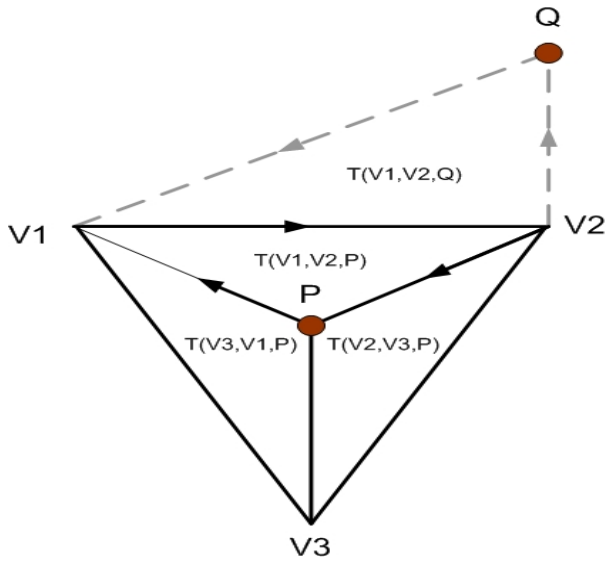


Figura 6.14.- Determinación de pertenencia de un punto a un triángulo: *como se puede observar si un punto Q no está en el triangulo, el sentido de alguno de los sub triángulos formados por el punto y dos vértices es distinto al formado por el triangulo $T(V_1, V_2, V_3)$. En caso contrario punto está contenido en el triangulo, como ocurre con el punto P.*

Para el cálculo de la orientación de un triángulo T se puede utilizar la fórmula (5.3).

$$(V_{(1,x)} - V_{(3,x)})(V_{(2,y)} - V_{(3,y)}) - (V_{(1,y)} - V_{(3,y)})(V_{(2,x)} - V_{(3,x)}) \qquad (6.3)$$

Si el resultado de la ecuación (6.3) es ≥ 0, la orientación del triángulo será positiva $(+)$, en caso contrario, la orientación del triángulo será negativa $(-)$. Como pasó final se verifica si las orientaciones de los tres triángulos que tienen como vértice el punto p tienen el mismo signo y si esto es verdadero el punto está dentro del triángulo. En caso

contrario el punto está situado fuera del triángulo. La complejidad del algoritmo es de orden $O(1)$.

6.3.3.3 Algoritmo general del cálculo paralelo de la grilla de densidades

La información de los elementos y nodos está disponible en todos los procesos al igual que los datos de áreas, coeficientes de área y numeración global de los nodos de los elementos. Estos cálculos, junto con los valores nodales obtenidos en la aproximación inicial de elementos finitos está disponible en todos los procesos. Esta información no está distribuida paralelamente, ya que el consumo de memoria requerida no es alto y un proceso de reordenamiento y creación de sub vectores si bien puede ser relativamente rápido, es un cálculo extra que no es prescindible de optimizar.

El algoritmo paralelo del cálculo de la grilla (Figura 6.15) realiza lo siguiente: cada proceso k genera una grilla Ω_k^g. Además cada proceso calcula en forma independiente las dependencias de los nodos a los centroides son enviados en un proceso anterior, por lo cual cada proceso tiene la información de las pertenencias.

Después que todos los procesadores tienen la información necesaria para realizar la aproximación de $u(p)$ en sus respectivos dominios, comienza el ciclo de cálculo de las aproximaciones. Para cada punto se busca sólo en los elementos e que pertenecen al dominio Ω_k^E donde se utiliza la regla lógica "s*i cualquiera de los nodos del elemento tiene asignada un pertenencia k, donde k es el identificador de procesador*". Si esta regla es verdadera es porque el elemento tiene uno o más nodos en el dominio Ω_k^E y existe la posibilidad de que el punto p_i esté contenido en el elemento e.

Luego se procede a la verificación de que si el punto está contenido en el elemento ($e \ni p_i$), para lo cual se utiliza el procedimiento visto en la sección ***6.3.3.2***. Si el punto es interior al elemento e se realiza la aproximación nodal, en caso contrario verifica el siguiente elemento, si no hay ningún elemento que contenga al punto, entonces el punto está fuera del dominio y su valor es 0. Luego se almacena la aproximación y se continúa el ciclo hasta evaluar todos los puntos de la grilla, para finalmente se exporta la solución a un archivo.

En la figura 6.15 se muestra en pseudocódigo el algoritmo general del proceso de cálculo de la grilla de densidades a partir de una aproximación inicial de elementos finitos triangulares de tres nodos.

Paso 1: Cada procesador k genera una grilla con dominio Ω_k^g a partir de los limites globales, calcula su centroide CT_k y asigna la información de pertenecías de los nodos a cada dominio Ω_k^g.

Paso 2: El procesador central codifica la información de los elementos e y se la envía a todos los procesadores.

Paso 3: Proceso de búsquedas de $e \ni p_i$ y aproximaciones de $u(p_i)$

Mientras no se aproxime u en todos los puntos $p_i \in \Omega_k^g$

Paso 3.1: Buscar en Ω_k^E el elemento e tal que $p_i \in \Omega^e$

Si éxito en Paso 3.1

Calcular $u(p_i) = \sum_{j=1}^{3} \Phi_j^e \psi_j^e(p_i)\ tal\ que\ e \ni p_i$

Si no

$u(p_i) = 0, \quad (\ \Omega_k^E \not\ni p_i\ )$

Paso 3.2: Almacenar $u(p_i)$ en un vector

Fin Mientras

Fin Paso 3

Paso 4: Todos los procesadores envían la información calculada al procesador central.

Paso 6: El procesador central recolecta la información de todos los procesadores y construye el vector de aproximación de todos los puntos de la grilla.

Paso 6: Se almacena en un archivo la aproximación de la grilla.

Figura 6.15.- *Algoritmo general del cálculo de la aproximación de densidades de una grilla de puntos. Obs: en lógica $\ni$ significa contenga.*

7 ANÁLISIS DE RESULTADOS

7.1 INTRODUCCIÓN

El análisis de resultados se divide en tres secciones. En la primera se realiza una validación de los resultados, comparándolo con el método de Monte Carlo. En la segunda sección se realiza un análisis de la implementación paralela en función de los tiempos de cálculo, Speedup, cómputo vs comunicación y mediciones estándar de rendimiento de la solución paralela. En la última sección se realiza un análisis de la aproximación de difusión de fotones en uno y varios dominios.

7.2 VALIDACIÓN DE RESULTADOS

7.2.1 Comparación con método Monte Carlo

El método Monte Carlo utilizado para la validación de la aproximación con elementos finitos. Está basado en el de (Jacques, y otros, 1998). La base teórica de la difusión de fotones con el método Monte Carlo se encuentra en (Jacques, 1998) (Wang, 1998). Algunas modificaciones, realizadas al software original se relacionan con la función de fase, al cambio energético que los fotones sufren en su camino, y a la función de dispersión, para que fueran similares a la derivación de la ecuación de transferencia radiactiva ETR. Cabe recordar que el método Monte Carlo utilizado, estima una tasa de fluencia relativa de fotones a partir del aporte energético que los fotones van dejando en su camino cuando transfiere parte de su energía al momento de ocurrir un evento de absorción en la ruta de cada fotón.

La función de fase que se utiliza es la función de Henyey-Greenstein (2.5) asumiendo una ruta isotrópica ($g = 0$), por lo cual la función de fase para dos dimensiones es $1/2\pi$ y para tres dimensiones $1/4\pi$. Esta función se encarga de ubicar la partícula con su nuevo estado energético en una posición nueva del dominio. El método Monte Carlo utilizado aproxima la tasa relativa de fotones $F/P\ cm^{-2}$, donde F es la tasa de fluencia Wcm^{-2}, donde cada fotón tiene como unidad de potencia $P = 1W$. Para obtener la tasa de fluencia relativa se divide por el producto del número de fotones que se calcularon y el área infitesimal en la sección del radio particionado. Luego para obtener la unidad de medida cm^{-2} se multiplica el resultado por u_a.

La aproximación realizada con elementos finitos da como resultado el número de fotones en zonas donde se ubicaron los nodos de los elementos, pero es necesario utilizar un factor de escala para efectos de normalización de las distribuciones en base a la resolución, además de calcular la unidad de área de cada aproximación nodal, ya que al utilizar valores experimentales en el término fuente no incluimos la unidad de medida en la resolución del problema. La variable de escala está asociada al número de nodos, y la unidad de área se puede aproximar dividiendo el área global por el número de nodos. Esta aproximación de unidad de área es solo válida si se utiliza una geometría de triángulos con áreas similares entre ellas, en otras palabras una triangulación uniforme. Esta problemática es solucionada cuando se calcula la grilla de puntos donde todas las estimaciones son equidistantes entre si y la unidad de área es constante en la aproximación. En esta aproximación se asume que los fotones tienen una energía constante por lo cual no se asigna un valor energético al fotón.

Como estamos utilizando valores experimentales en la fuente de fotones, los valores de tasa de fluencia relativa difieren con los obtenidos en el método Monte Carlo. Entonces para realizar la validación utilizaremos las tasas de fluencias relativas normalizadas de ambos métodos. Para la validación se utilizaron valores de absorción y coeficiente de dispersión reducido obtenidos de los trabajos de (Sterenborg, y otros, 1989) y (Zee, 1993). En la Tabla 7.1 se muestran los valores utilizados:

Tabla 7.1.- *Propiedades ópticas en varios tipos de tejidos*.

Tipo de tejido	**Tipo**	$u_a\ cm^{-1}$	$u_s'\ cm^{-1}$	**Referencia**
Materia gris adulto	In vivo	0.18	4.8	(Bevilacqua, y otros, 1999)
Materia blanca adulto	In vivo	0.13	9.8	(Bevilacqua, y otros, 1999)
Cráneo adulto	In vivo	0.22	9.1	(Bevilacqua, y otros, 1999)
Materia gris adulto	In vitro	0.4	19	(Zee, 1993)
Materia blanca adulto	In vitro	0.2	80	(Zee, 1993)

A continuación se muestran los resultados obtenidos y un análisis de la información que se puede obtener de los gráficos. Las pruebas fueron realizadas utilizando como función de fase $1/2\pi$. Para la validación se utilizo una circunferencia de radio 3 cm como geometría de difusión, con la fuente de fotones centrada en el origen de la

circunferencia, donde se realizaron 350 muestras de densidades de fotones desde el centro de la circunferencia al límite de la circunferencia. La figura 7.1 muestra la distribución de densidades relativas normalizadas obtenidas con el método Monte Carlo (MC) y el método de Elementos Finitos (FEM).

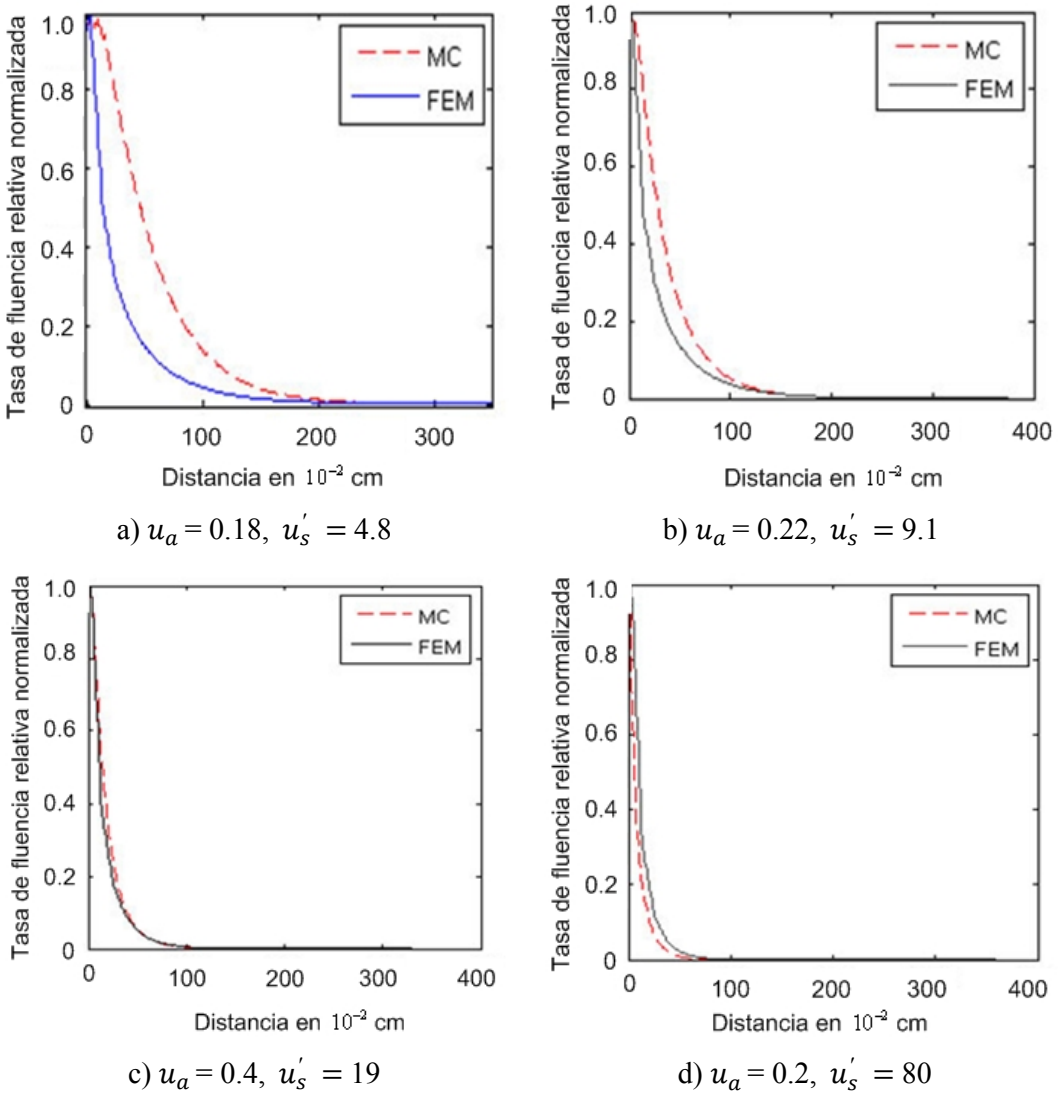


Figura 7.1.- *Distribucion de densidades de fotones normalizada: a) materia gris adulto in vivo, b) cráneo adulto in vivo, c) materia gris adulto in vitro, d) materia blanca adulto in vitro. Obs: en los gráficos la distancia esta en unidad de medida* $10^{-2}cm$.

Como se puede observar en la figura 7.1 el método no aproxima correctamente con todos los valores de absorción y coeficiente de dispersión reducido. Esto se debe a que

la aproximación de la difusión utilizando la derivación de la ETR, es solo valida cuando $u_s' \gg u_a$ (Arridge, 1999). En la figura 7.1 a medida que aumenta la dispersión las curvas de densidades relativas normalizadas tienden a ser similares con ambos métodos, con lo cual se puede sustentar que la aproximación realizada en este trabajo utilizando elementos finitos como método de aproximación de la difusión de fotones es correcta, mientras se cumplan las condiciones de alta dispersión en la materia que se desea obtener la difusión.

7.3 TIEMPOS DE CÓMPUTO, MEDIDAS DE RENDIMIENTO Y CONSUMO DE MEMORIA

Para medir el desempeño de los algoritmos paralelos se utilizó la métrica "speedup relativo" S_A, que la razón entre el tiempo de ejecución del algoritmo usando un procesador T_1 dividido por el tiempo de ejecución del algoritmo paralelo en k procesadores T_k (7.1). Además se realizara un análisis de los tiempos de cómputo versus los tiempos de comunicación. Para el análisis de los tiempos de cálculo, se utilizo dominio circular de radio 3 cm, con geometrías de 1089, 4225, 16641 y 66049 nodos, y con 2048, 8192, 32768 y 131072, elementos respectivamente. Las pruebas paralelas fueron realizadas en 4 nodos con dos procesadores cada uno. Para la resolución de sistema de ecuaciones utilizando el método LSQR se utilizó como criterio de parada, el número de iteraciones igual a la dimensión de nodal de la geometría.

$$S_A = \frac{T_1}{T_N} \tag{7.1}$$

7.3.1 Tiempos de cómputo de versiones seriales

Se realizo un análisis previo de las dos versiones seriales, donde las diferencias entre las dos versiones radican en el tipo de almacenamiento de la matriz (almacenamiento normal y disperso) que compone al sistema de ecuaciones y la forma como se realiza la multiplicación matriz por vector (multiplicación normal y dispersa) requerida por el algoritmo LSQR. Al comparar los tiempos de cómputo de las dos versiones seriales, la versión que utiliza procedimientos de almacenamiento y cálculo disperso es considerablemente menor (Figura 7.2). Es por ello que la versión serial dispersa será la

base de los algoritmos paralelos implementados, los cuales fueron explicados en las secciones 6.3.2.

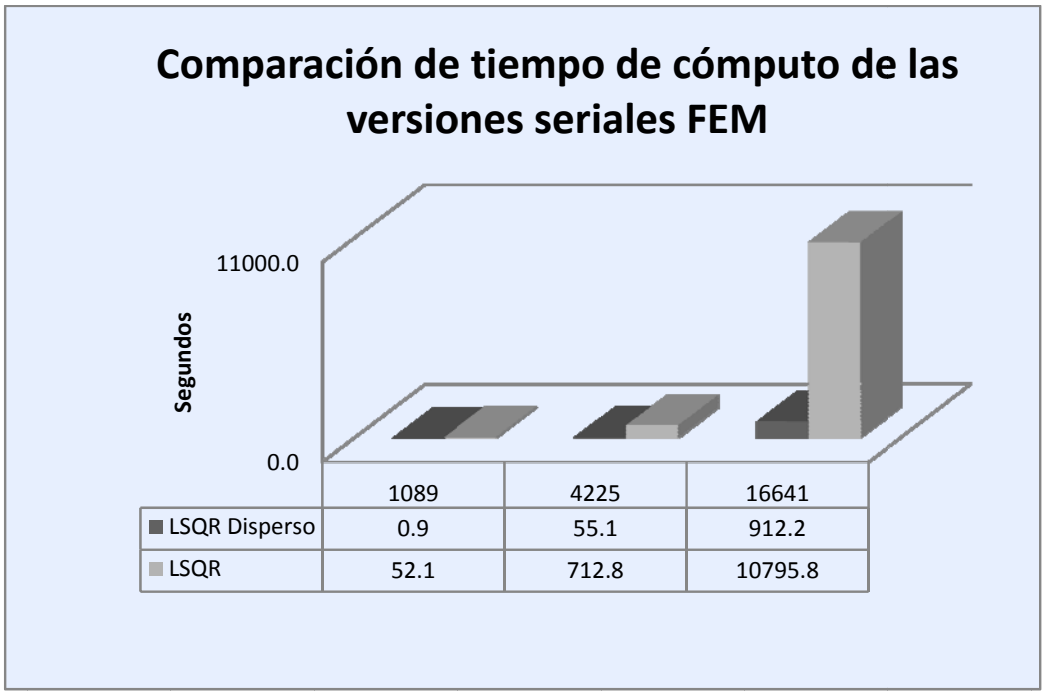


Figura 7.2.- *Grafico comparativo de los tiempos de cálculo obtenido por las dos versiones seriales (un procesador sin procesos de comunicación) del método de aproximación de elementos finitos, en ambos procedimientos se utilizó como criterio de parada el número de iteraciones igual al número de nodos.*

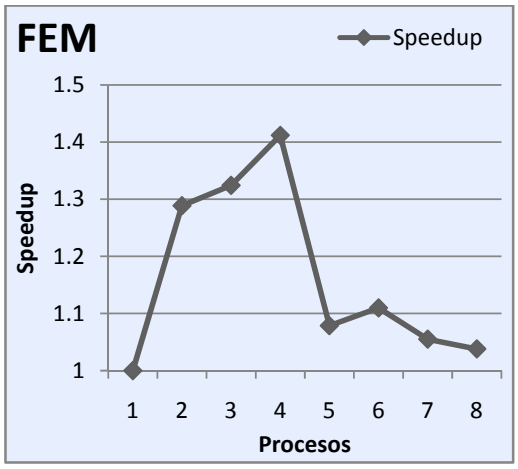
7.3.2 Medidas de rendimiento

A continuación de mostraran una serie de gráficos de rendimiento utilizando la métrica Speedup, los cuales están organizados de la siguiente manera: a) rendimiento de la aproximación de elementos finitos (FEM), específicamente a la resolución paralela del sistema de ecuaciones con el método LSQR que es la que consume casi la totalidad del tiempo del método, b) distribución de tiempos de computo versus comunicación en la aproximación del método de elementos finitos, b) rendimiento del cálculo paralelo de la grilla de densidades (Grilla) y c) rendimiento global que incluye los dos procedimientos paralelos implementados (FEM + Grilla).

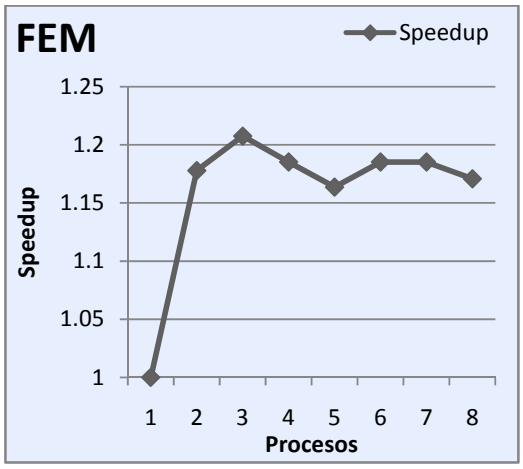
7.3.2.1 Speedup del proceso FEM

En la figura 7.3 se presentan los Speedup obtenidos por los algoritmos propuestos (dos versiones de comunicación) que se enfocan principalmente a resolver en paralelo el sistema de *N* ecuaciones con el método LSQR.

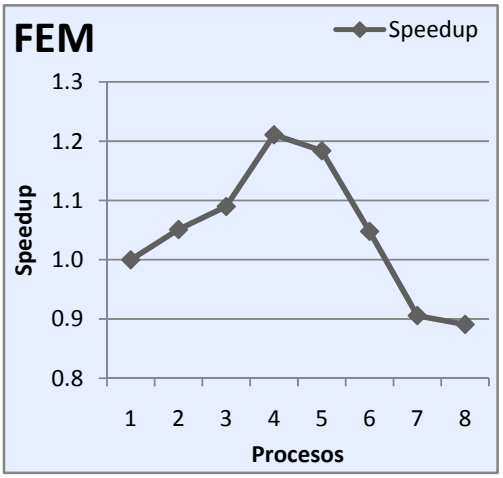
Con Allreduce **Con Gather + Broadcast**



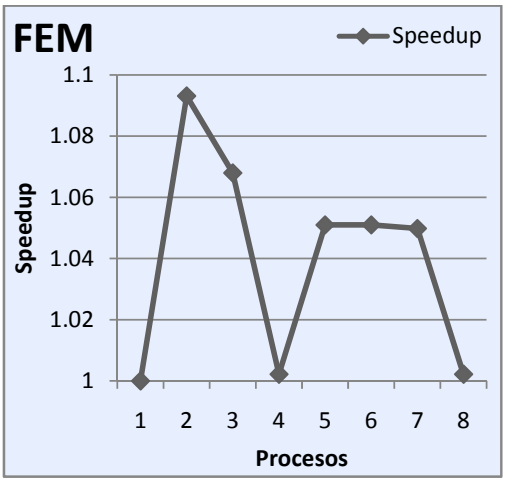
a.1) N=1089



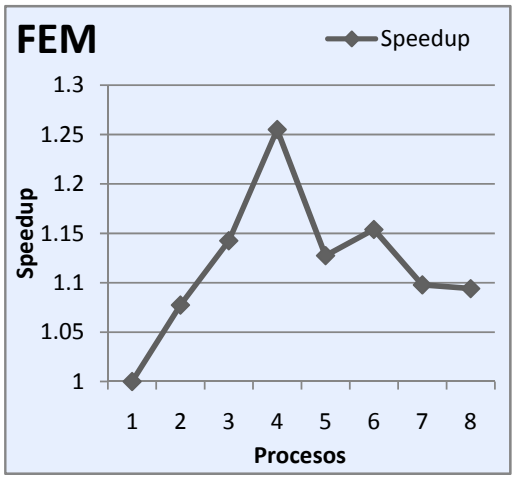
b.1) N=1089



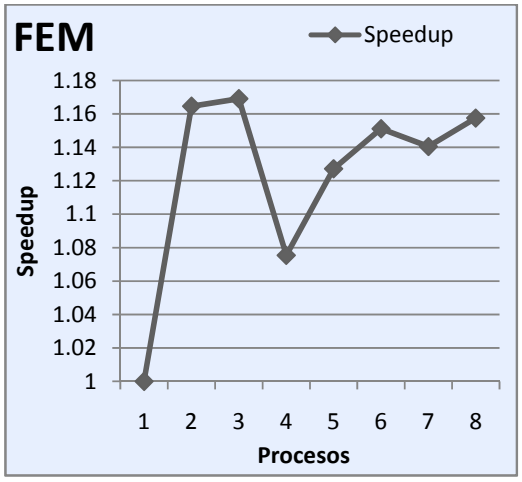
a.2) N= 4225



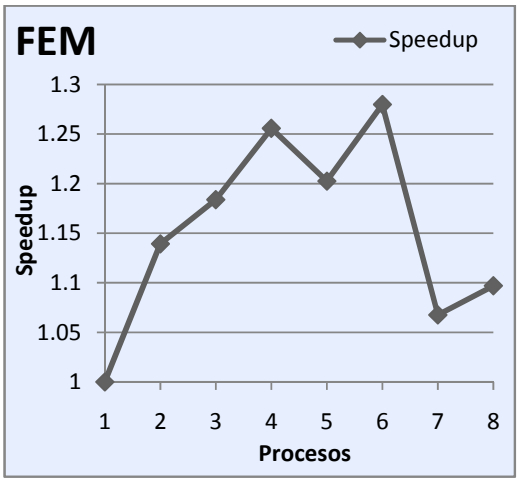
b.2) N= 4225



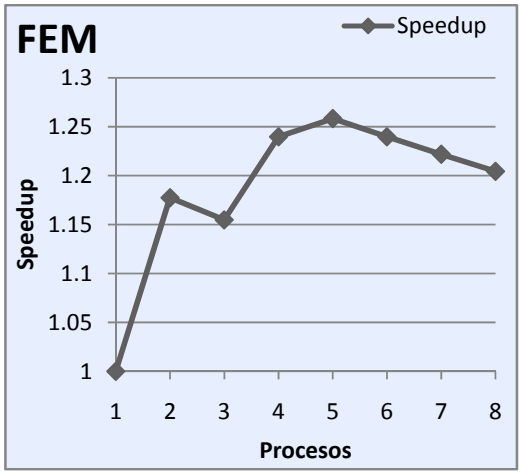
a.3) N =16641



b.3) N=16641



a.4) N= 66641



b.4) N=66641

Figura 7.3.- *Speedup de los dos algoritmos paralelos propuestos para la aproximación FEM utilizando el método LSQR para resolver el sistema de ecuaciones: a) utilizando Allreduce en los procesos de comunicación, b) utilizando Gather + Broadcast en los procesos de comuniacion.*

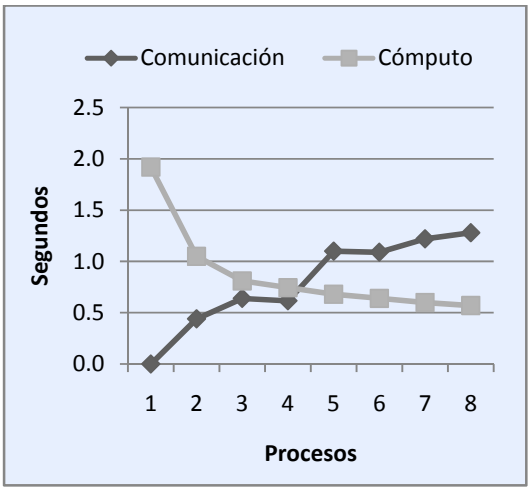
Como los mejores tiempos de la solución del sistema de ecuaciones se obtienen con el algoritmo de la figura 6.10 (versión con Allreduce), se utilizó esta versión paralela en la implementación final.

7.3.2.2 Distribución tiempos de computo vs comunicación FEM – LSQR

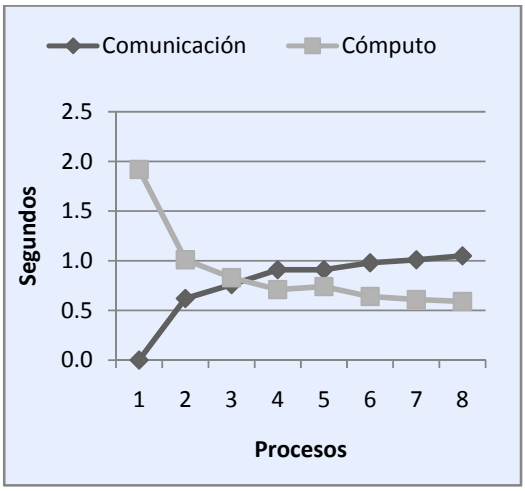
En la figura 7.4 se muestran las distribuciones de los tiempos de cómputo vs los tiempos de comunicación, donde en todas las mediciones, los tiempos de cómputo disminuyen a

medida que se aumenta el número de procesos paralelos. Sin embargo, a medida que se aumentan el número de procesos, la comunicación empieza a incrementarse considerablemente, llegando a un punto en la cual el costo de comunicación es tan alto que la paralelización del método es solo eficiente hasta un número muy bajo de procesadores.

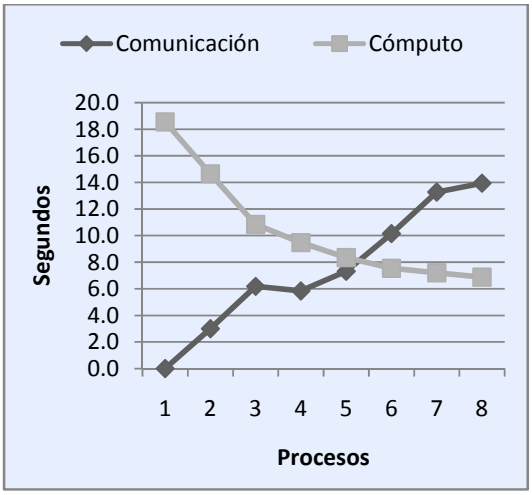
Con Allreduce **Con Gather + Broadcast**



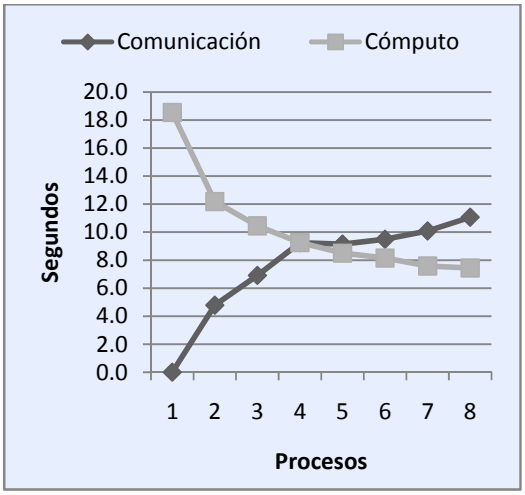
a.1) N=1089



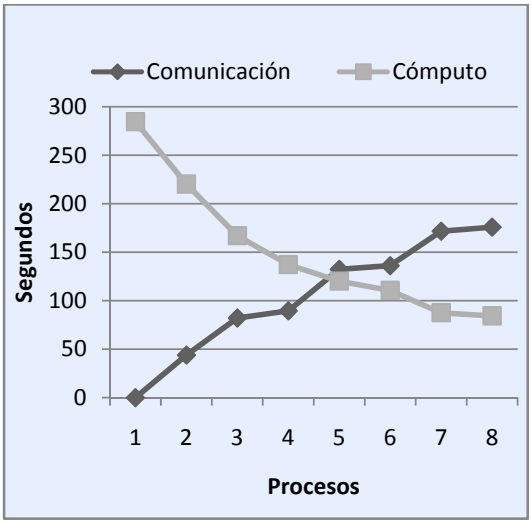
b.1) N=1089



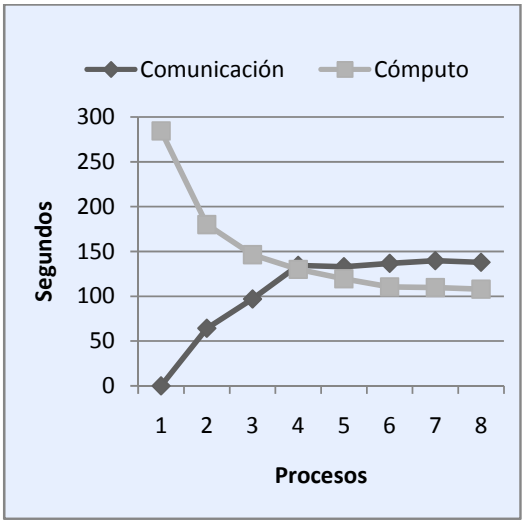
a.2) N= 4225



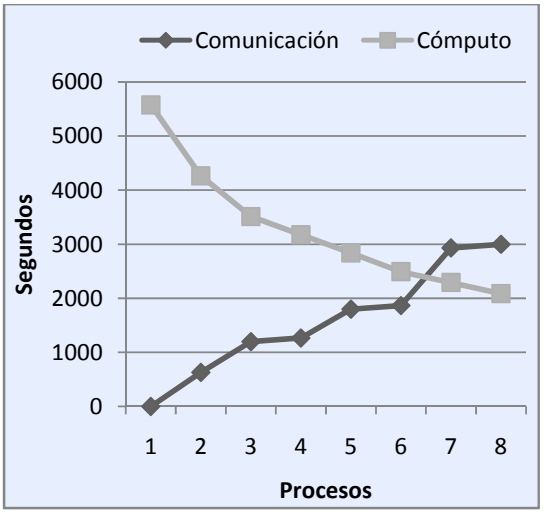
b.2) N= 4225



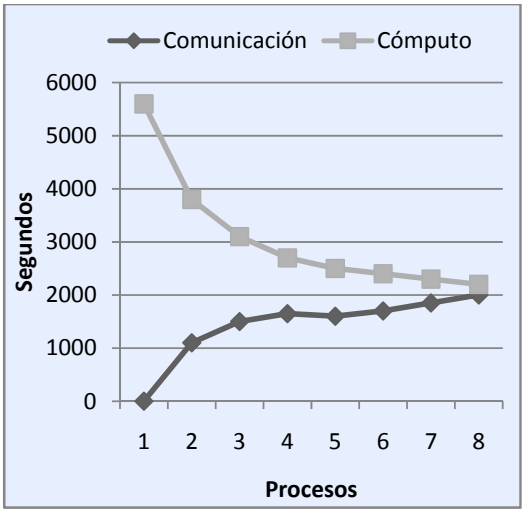
a.3) N =16641



b.3) N =16641



a.4) N= 66641



b.4) N= 66641

Figura 7.4.- *Distribución de comunicación y cómputo en la aproximación FEM utilizando el método LSQR para resolver el sistema de ecuaciones: a) utilizando Allreduce en los procesos de comunicación, b) utilizando Gather + Broadcast en los procesos de comuniacion (Unidad de medida = Segundos).*

En el problema de difusión de fotones de luz aproximado con el método de elementos finitos, las soluciones obtenidas comienzan a converger a la solución exacta en geometría nodales mayores de 4000 nodos en un diámetro de hasta 10 cm del dominio.

Estas observaciones fueron validadas, sumando todas las densidades relativas nodales del dominio en los ejemplos en estudio. Este valor no sufre variación cuando se

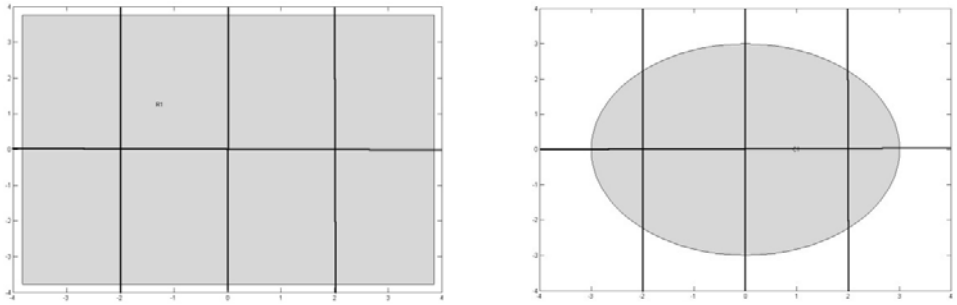
utilizan 16641 nodos ya que al comparar la sumatoria de densidades relativas con la suma de una geometría nodal de 66641 el error obtenido es cercana a 1×10^{-8}.

Entonces podemos deducir que una mayor resolución de la geometría, lo cual implicaría un sistema de ecuaciones de mayor dimensión, no es estrictamente necesaria, ya que los valores a aproximar tendrán un error muy bajo. Sin embargo un aumento en la resolución, implicaría un consumo excesivo de recursos, como lo son la reserva de memoria para el sistema de ecuaciones y la comunicación realizada en los procesos paralelos, donde el tamaño de los mensajes aumentaría.

7.3.2.3 Speedup del proceso de cálculo de la grilla

Se realizaron pruebas sobre dos tipos de partición de geometrías de elementos finitos; una en que el número de elementos es muy similar en cada partición del dominio de los elementos (distribución uniforme del dominio de elementos en cada partición) y una en que el número de elementos de cada partición no es el mismo en todas las particiones (distribución no uniforme del dominio de elementos en cada partición). En la figura 7.5 se muestran los dominios de elementos finitos y como se distribuyen la búsqueda y posterior cálculo de la grilla de puntos.

La comunicación requerida en el proceso de cálculo de la grilla es mínima y no influye en los tiempos totales del cálculo de la grilla, ya que sólo existen dos comunicaciones; una inicial que envía la información necesaria para que cada proceso realice sus aproximaciones locales de los subdominios y una en el cual envía la información de los cálculos realizados por cada proceso.



a) b)

Figura 7.5.- *Distribución dominio en la grilla: a) Distribución de dominios cercana a uniforme, b) distribución de dominios no uniforme en todos los procesos.*

Si se plantea un situación hipotética en la cual se desea buscar N puntos de una grilla Ω^G en los M elementos de una geometría con dominio Ω^E en P procesos, donde el número de puntos $Q_k \in \Omega_k^g$ $(k = 1,2, \dots P)$ es igual todas las particiones de Ω^G y el número de elementos R_k que están contenidos en cada partición Ω_k^g es igual en cada partición Ω_k^E, donde Ω_k^E es el subdominio que contiene a todos los elementos que pertenecen a una grilla Ω_k^g. Entonces $R_k = M/P$ y $Q_k = N/P$ y el orden de la complejidad del algoritmo de búsqueda por cada proceso P en el peor de los casos se aproxima a la siguiente expresión:

$$\frac{M}{P} \times \frac{N}{P} = O\left(\frac{MN}{P^2}\right) \tag{7.1}$$

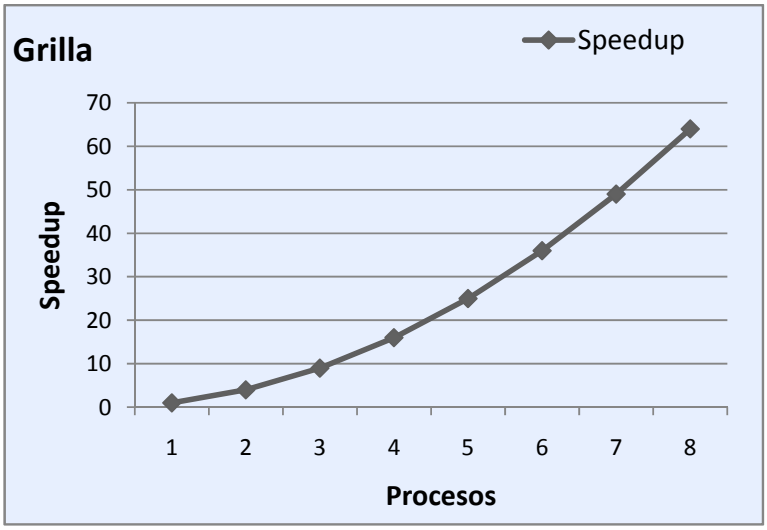
La complejidad del algoritmo del proceso de determinación de pertenencia de un punto dentro de un triangulo visto en la sección ***6.3.3.2*** necesario para la obtención de densidades de la grilla es de orden $O(1)$ y no influye mayormente en la expresión (7.1). Entonces con el orden de complejidad hipotético (7.1) se obtendría un Speedup similar al que se muestra en la figura 7.6.a.

Además es importante destacar que el procedimiento que realiza la búsqueda de un punto interior del triangulo cuando se utiliza un proceso (versión serial) es distinta a la utilizada en procesos paralelos (versión paralela) y esta diferencia radica en la forma en que se obtiene la información de los nodos de un elemento finito triangular.

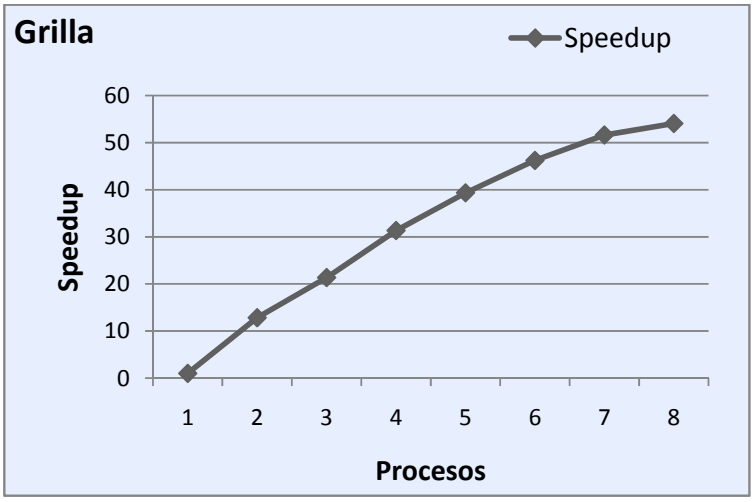
En la versión serial la obtención de la información es a través de los métodos en que se retornan atributos (información nodal) del objeto "***Elemento_Triangular***". En la versión paralela la obtención la información se obtiene directamente a través de punteros donde esta información ha sido codificada y distribuida a todos los procesos en matrices en que cada fila de la matriz contiene la información del elemento finito triangular.

En la figura 7.6.b se muestra el Speedup del cálculo de una grilla de densidades con una distribución uniforme de elementos en donde no necesariamente tienen el mismo número de elementos en todos los subdominios pero si existe una distribución de carga en las búsquedas más equilibrada. En la figura 7.6.b se muestra el Speedup del cálculo de una grilla de densidades con una distribución no uniforme de elementos en

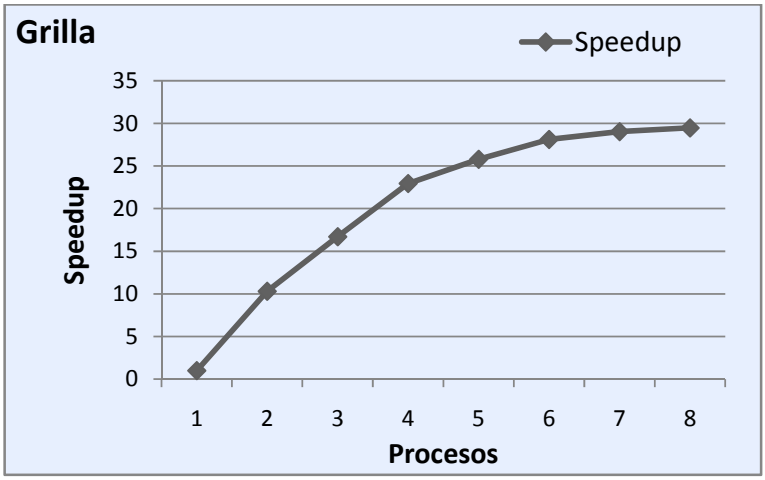
donde no existe una buena distribución de carga en los procesos, lo cual determina una menor eficiencia del proceso paralelo del cálculo de grilla si se compara con los resultados obtenidos en el cálculo de la grilla en una distribución uniforme de los elementos (Figura 7.6.b).



a)



b)



c)

Figura 7.6.- *Speedup del cálculo de la grilla: a) caso hipotético de distribución del dominio de los elementos, en que el número de elementos es igual en todos los subdominios de la grilla, b) dominio de elementos distribuido uniformemente en los subdominios de la grilla, c) dominio de elementos distribuido no uniformemente en los subdominios de la grilla.*

Como se puede observar en la figura 7.6.b y 7.6.b los Speedup que se obtienen son muy buenos, ya sea para el caso de distribuciones homogéneas como no homogéneas de los elementos en cada partición de la grilla. Esto se debe principalmente a varios factores: a) al existir solo una comunicación al inicio y otra al final del proceso, implica que la ejecución del proceso paralelo de cálculo de la grilla sea embarazosamente paralelo y cada sub cálculo de la grilla realizado por cada proceso es independiente de todos los otros cálculos, b) los procesos de búsqueda de la información de los elementos utilizando punteros en la versión paralela es más rápida que el retorno de información utilizando objetos que se utiliza en la versión serial del algoritmo de determinación de pertenencia de un punto en el interior de un elemento finito triangular, c) en el peor de los casos se tendrá una complejidad cercana al orden de la ecuación 7.1 y esto no sucede en el cálculo de la grilla en paralelo, pues no necesariamente se busca en todos los elementos finitos triangulares para encontrar si un punto está en el interior del triangulo, ya que el encontrar al elemento que contiene a un punto especifico puede ocurrir en cualquier iteración del ciclo de búsqueda de los elementos que contienen a un punto y en mínimos casos será en la última iteración del proceso de búsqueda.

Además, a medida que se aumenta el número de procesos, los tiempos de cálculo comienzan a estabilizarse y esto se debe a que se empieza a llegar al límite mínimo de tiempo requerido por cada proceso. Este tiempo mínimo está condicionado al número de elementos del dominio global Ω^E, ya que cada proceso verifica la pertenencia de cada elemento e al dominio Ω_k^E aplicando la regla lógica verdadera "s*i cualquiera de los nodos del elemento tiene asignada un pertenencia k*" entonces el elemento $e \in \Omega_k^E$ y se procede a determinar la pertenencia del punto en un triangulo y se calcula la densidad en ese punto utilizando la aproximación lineal (6.2) en caso contrario el punto está fuera del dominio y su valor es 0. Es por esta razón que el aumento del número de procesos no asegurara un crecimiento cuadrático de Speedup visto en la figura 7.6.a si no que más bien se tiende a estabilizar. En la figura 7.7 se visualiza esta última observación.

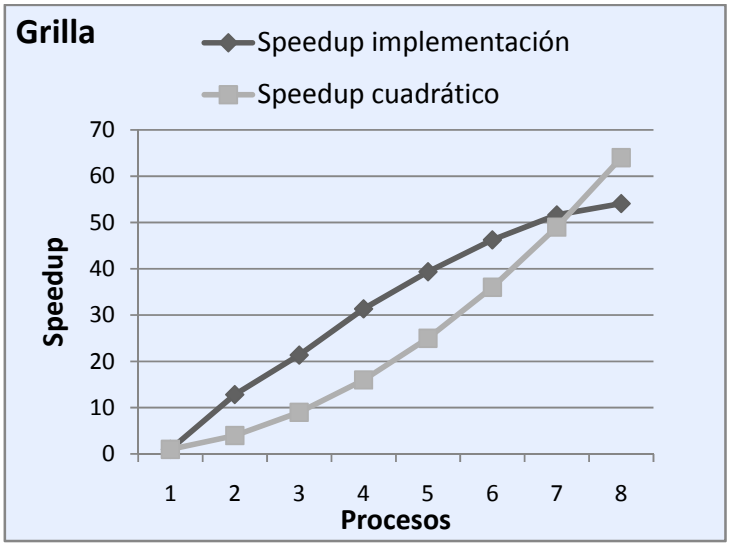
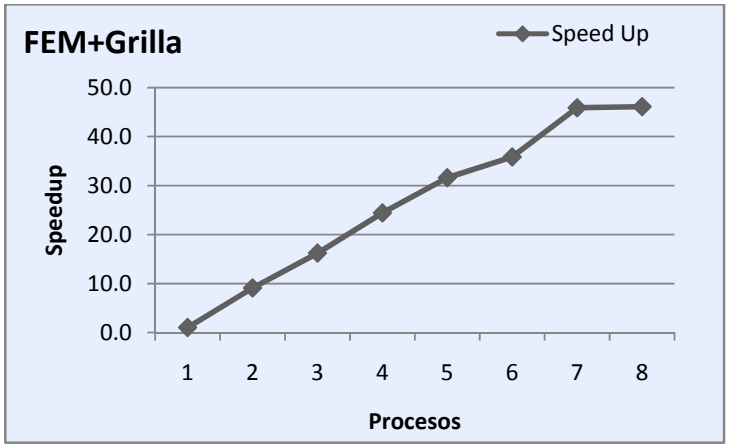


Figura 7.7.- *Comparación del Speedup obtenido en la implementación utilizando una distribución uniforme de elementos en cada partición y un Speedup cuadrático obtenido utilizando la formulación matemática hipotética 7.1.*

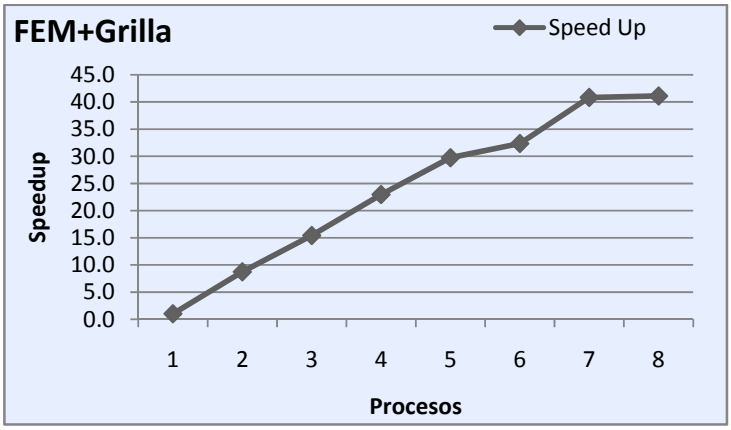
7.3.2.4 Speedup de la aproximacion FEM y posterior cálculo de la grilla de densidades (FEM+Grilla)

En la figura 7.8 se muestran los Speedup de la solución paralela y el cálculo posterior de una grilla de 1000×1000 puntos que son buscados y calculados en los 2048, 8192,

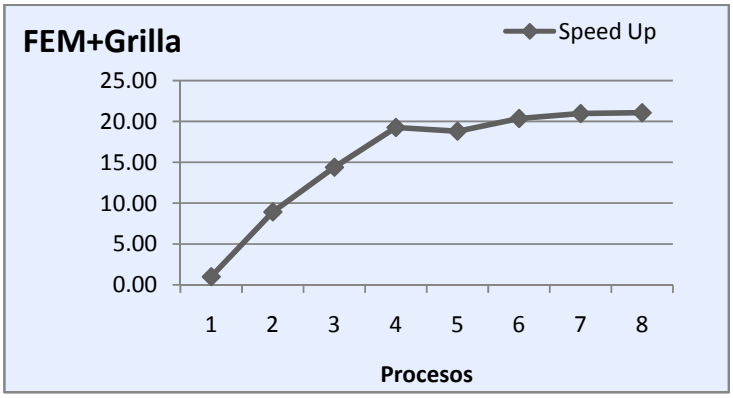
32768 y 131072 elementos que componen el dominio geométrico de la aproximación calculada en el proceso FEM.



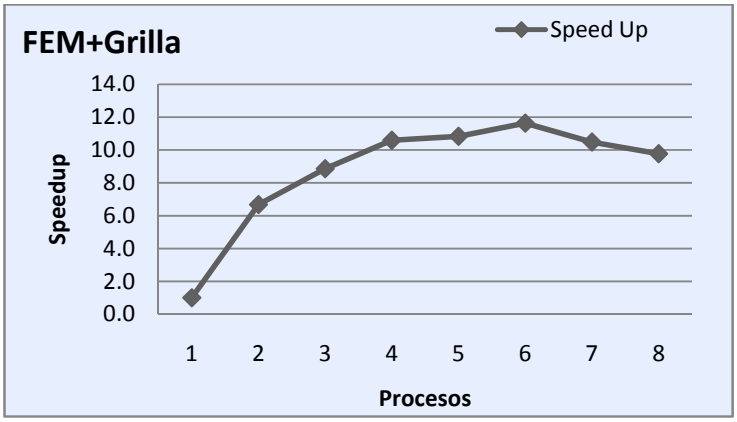
a) 1089 ecuaciones



b) 4225 ecuaciones



c) 16641 ecuaciones



d) 66049 ecuaciones

Figura 7.8.- *Speedup de la solución paralela donde se calcula en todas una grilla de 1000×1000 (1000000 de aproximaciones) utilizando una geometría de orden nodal N (numero de nodos de la geometría) que es el numero de ecuaciones del sistema a resolver.*

Como se puede observar en la figura 7.8.a y 7.8.b se obtiene un Speedup Superlineal y en las figura 7.8.c y 7.8.c si bien se produce este tipo de Speedup, el rendimiento de la solución comienza a converger a un valor estable. Esto se debe principalmente a que en la solución paralela el tiempo de cálculo de la resolución del sistema de ecuaciones empieza a consumir gran parte del tiempo total todo el proceso (FEM+Grilla). Donde el tiempo de cálculo total es dominado por el tiempo requerido en comunicación en el proceso FEM. A pesar de esta convergencia en el rendimiento de la solución paralela, como se dijo en la sección ***7.3.2.2*** la aproximación de elementos finitos empieza a converger a la solución exacta en geometrías de orden nodal mayor a 4000 nodos en un diámetro de hasta 10 cm en el dominio. Entonces como la aproximación de la difusión de fotones de luz con elementos finitos es válida a escala de centímetros, el esfuerzo computacional debe ser dirigido en este caso al cálculo paralelo de la grilla de densidades, la cual es necesaria en aplicaciones de Tomografía Óptica y una mayor resolución geométrica nodal no significara una mejor aproximación.

Por tales razones los Speedup obtenidos en la aplicación global (FEM+Grilla) son buenos en relación a la calidad de la solución entregada que tiene un error cercano a 1×10^{-8} y a la dimensión de la problemática planteada, que es la solución de la ecuación diferencial del transporte de la luz en medios turbios en dos dimensiones.

7.3.3 Consumo de memoria

En la figura 7.9 se compara el uso de memoria con las variantes de almacenamiento de matriz dispersa y matriz normal utilizados en el proceso FEM. Si bien la matriz del sistema de ecuaciones depende del número de datos no nulos, a medida que se aumenta la resolución de la malla los valores no nulos tienden a agruparse en las cercanías de la diagonal de la matriz y el número de valores no nulos es proporcional a la resolución de la geometría. Por lo tanto el consumo de memoria aumenta, pero al estar en formato disperso mientras mayor sea la resolución geométrica del problema, la proporción de valores no nulos será cada vez más pequeña en función de una mayor dimensión de la matriz. Por lo tanto a medida que se aumenta la resolución del problema, la optimización de memoria es evidente para problemas de mayor dimensión.

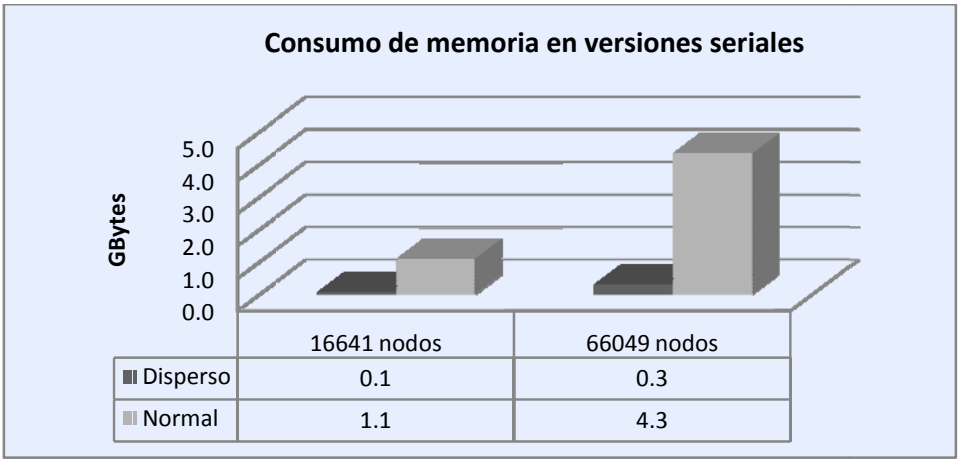


Figura 7.9.- *Consumo de memoria utilizando almacenamiento normal y disperso de las matrices que componen al sistema de ecuaciones en el proceso FEM.*

7.4 EXPERIMENTOS APLICADOS A LA DIFUSIÓN DE FOTONES

A continuación se mostrará una serie de gráficos de difusión de fotones de luz, en una geometría de dos dimensiones. Para las simulaciones de difusión de fotones de luz se utilizaron los valores de la Tabla 7.1 de la sección 7.2.1 que corresponden a factores del cráneo (Bevilacqua, y otros, 1999), materia blanca y materia gris del cerebro (Zee, 1993). Además de definieron fuentes de de fotones. La figura 7.10 es un esquema interno de un corte transversal del cerebro humano en el cual se puede ver una

distribución de la materia gris y materia blanca del cerebro y con la geometría presentada en la figura 7.11 se pretende simular la difusión de fotones con características similares a los tejidos presentados en la figura 7.10.

Para la simulación computacional se genera la geometría y se definen los dominios que serán utilizados en el método de elementos finitos utilizando el toolbox Pdetool de Matlab. Los dominios son representativos de los distintos tejidos y son usados como identificadores para la asignación de los coeficientes de dispersión y absorción utilizados en la simulación de difusión de fotones (Figura 7.11).

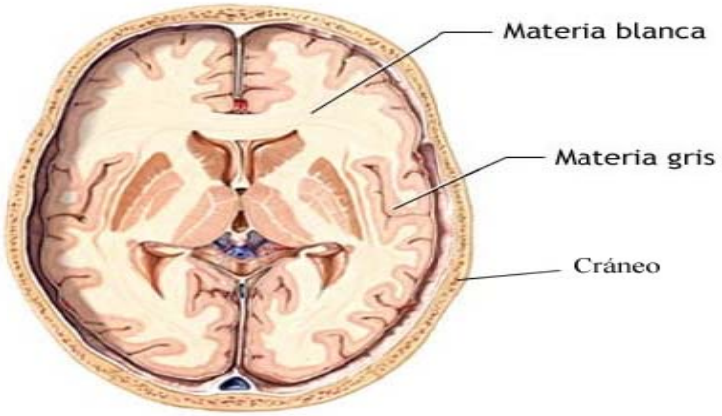


Figura 7.10.- *Distribución de tejidos en el cerebro.*

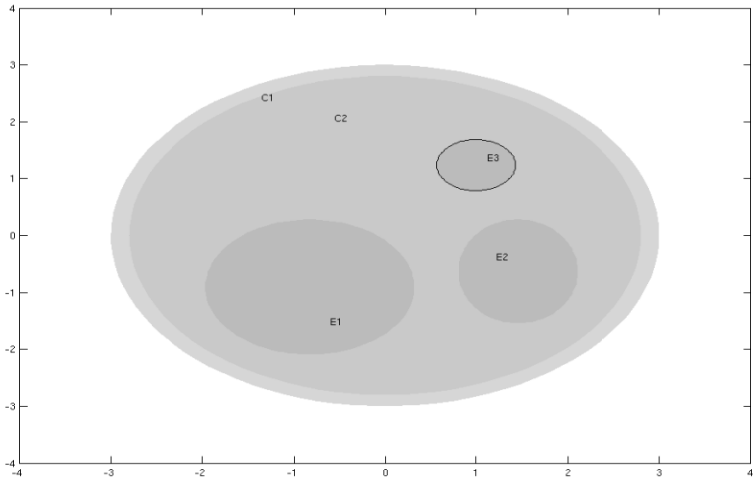


Figura 7.11.- *Generación de la geometría con Pdetool de Matlab, los dominios E1, E2 y E3 son referentes a la materia gris, C2 es la materia blanca y C1 es el cráneo.*

En la simulación se utilizó una geometría de 16641 nodos (Figura 7.12), donde se realizaron experimentos con condiciones de borde de tipo Dirichlet y de tipo Robín y con ambas condiciones de borde se obtienen difusiones de fotones similares (Figura 7.13). Además, se realizaron experimentos con mayor número de fuentes de fotones con los factores de absorción y dispersión presentados en la figura 7.11 (Figura 7.14) y otro experimento donde se intercambiaron las características de los dominios (Figura 7.15).

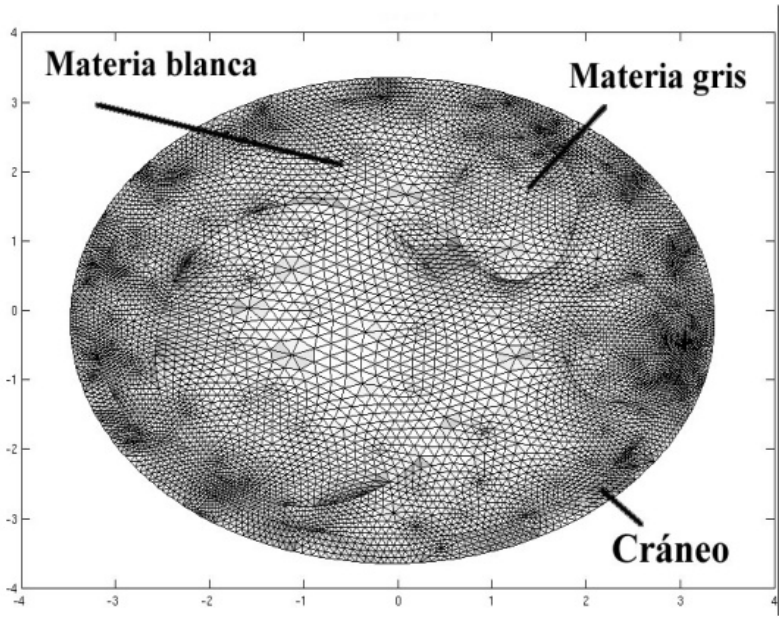
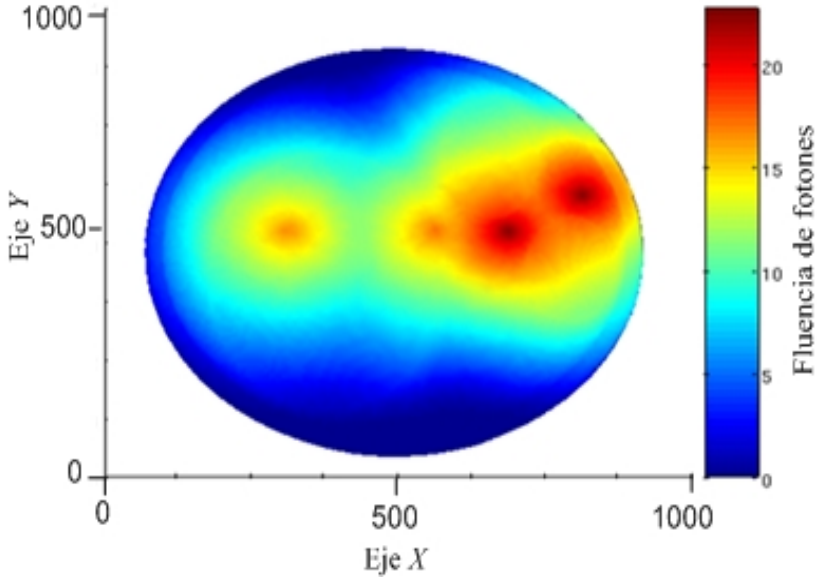
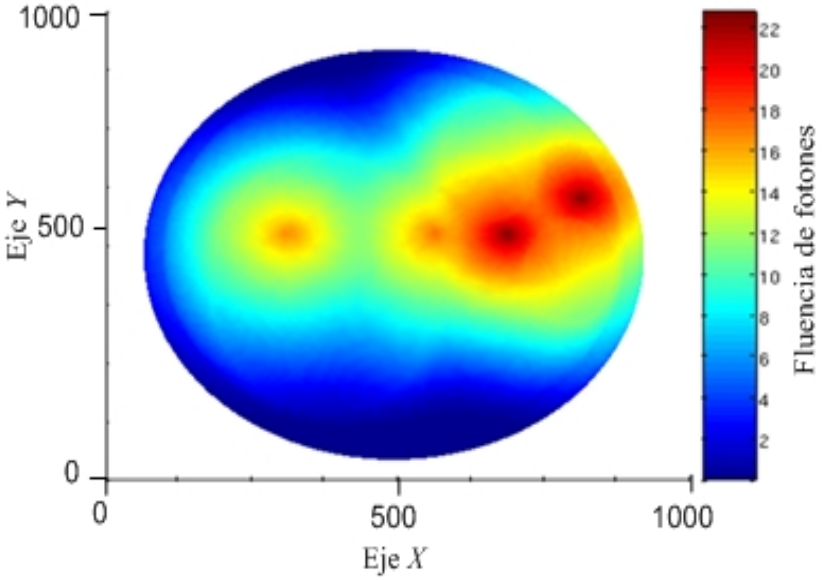


Figura 7.12.- *Geometría nodal utilizando elementos finitos triangulares de tres nodos.*

Si nos referimos a la aproximación de la grilla de puntos, ésta es útil cuando se requiere visualizar la solución, por ejemplo un monitor. La solución obtenida por la FEM es en los nodos de los triángulos, los cuales no necesariamente coinciden con la distribución cartesiana de los pixels en pantalla. En la figura 7.16 se muestra la resolución de una aproximación FEM cercana a una fuente de fotones con elementos triangulares y en la figura 7.17 se muestra la resolución de una grilla de puntos a partir de la misma aproximación FEM.



a) Condición de borde Dirichlet



b) Condición de borde Robín

Figura 7.13.- *Difusión de fotones utilizando 4 fuentes y constantes de absorción y dispersión representados en la figura 7.11 y la fluencia de fotones esta en escala logarítmica en base 10.*

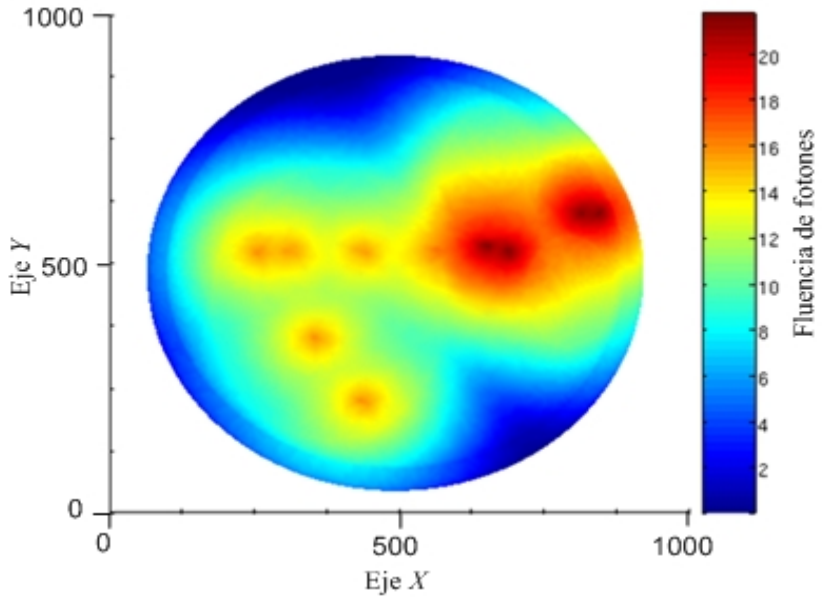


Figura 7.14.- *Difusión de fotones utilizando 10 fuentes y constantes de absorción y dispersión representados en la figura 7.11 y la fluencia de fotones esta en escala logarítmica en base 10.*

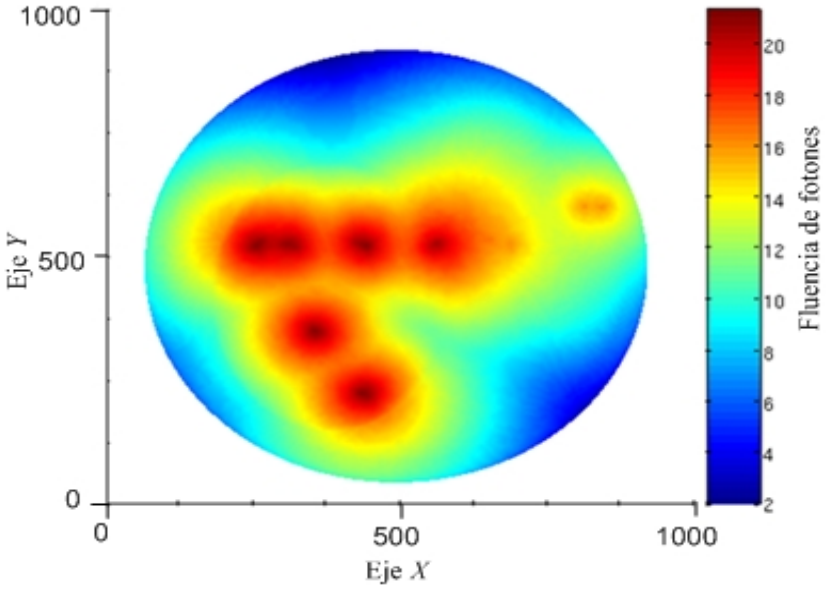


Figura 7.15.- *Difusión de fotones utilizando 10 fuentes, donde de forma experimental se intercambiaron los factores de absorción y dispersión de la materia gris con la materia blanca representada en la figura 7.11 y la difusión esta en escala logarítmica en base 10.*

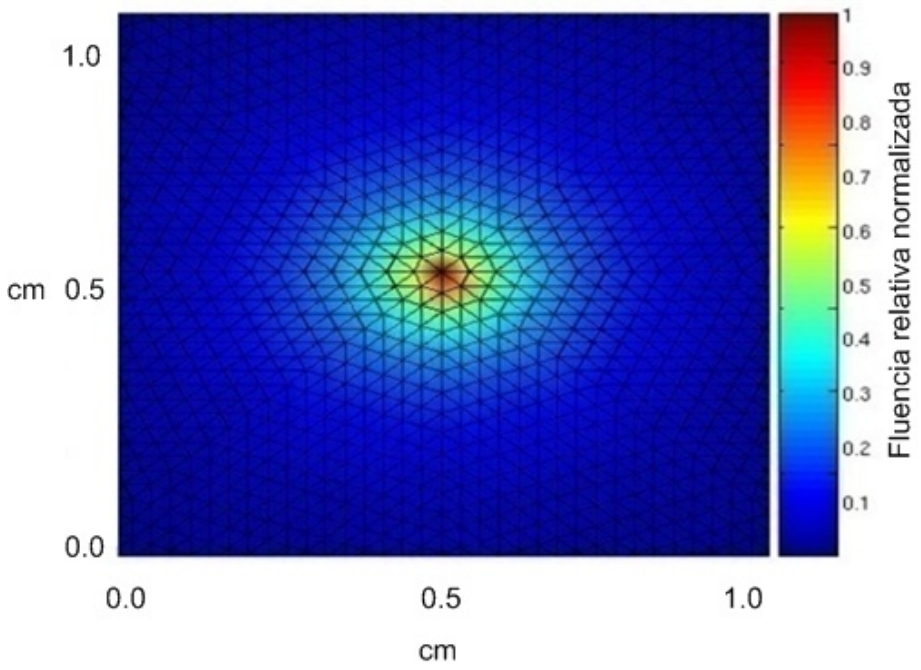


Figura 7.16.- *Aproximación FEM cercana a una fuente de fotones.*

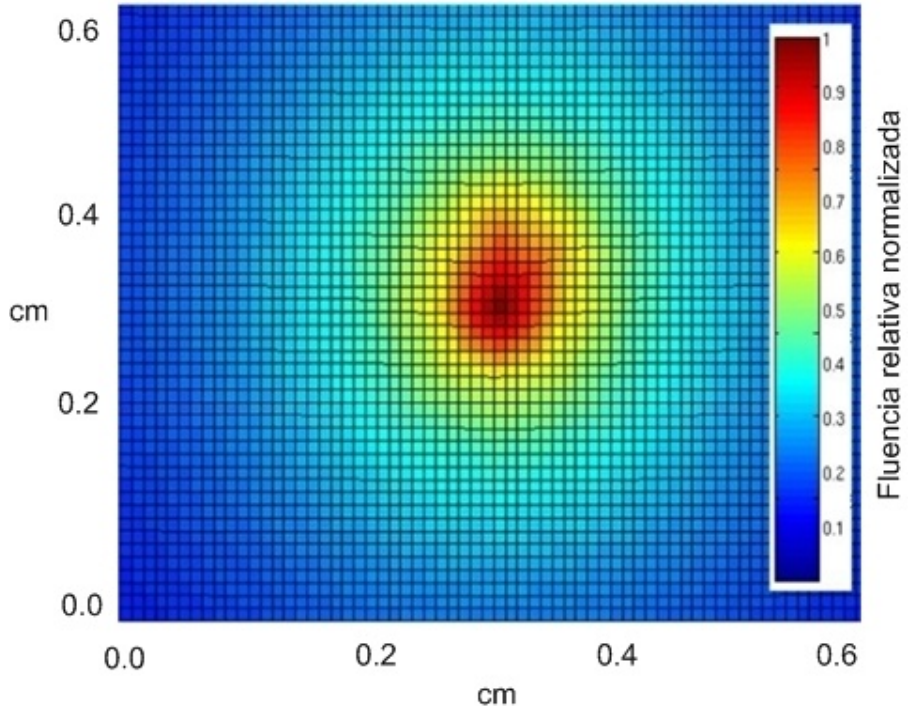


Figura 7.17.- *Grilla de densidades cercana a una fuente de fotones.*

Es evidente que la siguiente etapa del software el es modelamiento en tres dimensiones, la ventaja del modelo de difusión utilizado en este trabajo radica, en que no se debe modificar mayormente el modelo, ya que solo se cambia el tipo de geometría elemental y se deben implementar los cálculos de integrales elementales en tres dimensiones y algunas constantes de la ecuación de difusión, pero el procedimiento general es el mismo para el caso de dos dimensiones.

8 CONCLUSIONES

8.1 CONCLUSIONES IMPLEMENTACIÓN PARALELA

El objetivo de resolver la ecuación diferencial de difusión de la luz en medios difusos mediante una solución paralela con elementos finitos ha sido ha cumplido en plenitud. Las aproximaciones han sido validadas con el método estocástico Monte Carlo que es el más utilizado en tomografía óptica para las simulaciones de trayectorias de sub partículas. Además, se implementó un procedimiento paralelo para que a partir de una aproximación inicial de elementos finitos sea posible estimar una grilla de puntos contenidos en el dominio del problema. En la solución del sistema de ecuaciones en paralelo no se obtuvieron los resultados que se esperaban, ya que el rendimiento del cálculo paralelo del método LSQR está condicionado al costo requerido en comunicación. Pero a pesar de ello la versión serial del método LSQR utilizando almacenamiento disperso es muy rápida y este tema podría ser un posible trabajo de investigación en el cual se mida el rendimiento del costo de comunicación.

En el software existen clases en C++ que pueden ser usadas para la construcción de otras aplicaciones donde se necesiten resolver sistemas de ecuaciones, donde dicho problema es uno de los procesos más recurrentes en la solución de problemas de ingeniería. Adema el software está diseñado para solucionar ecuaciones diferenciales elípticas con elementos finitos donde se definan condiciones de contorno, por lo cual su uso no es exclusivo para el problema de difusión de fotones.

Otro aspecto importante es que para resolver el problema de difusión, no se requiere estrictamente un hardware de arquitectura de memoria distribuida si se utilizan geometrías hasta aproximadamente 16000 nodos y con estas resoluciones nodales se obtiene la solución exacta del problema de difusión de fotones en tejidos orgánicos con las unidades de medidas comentadas en este trabajo de investigación. El cómputo paralelo del cálculo de la grilla se puede realizar con el hardware disponible en mercado, los tiempo obtenidos con este tipo de arquitectura con una resolución de aproximadamente 16000 nodos es de aproximadamente 14 minutos, lo cual es un tiempo aceptable para la solución de la ecuación de difusión de la luz en medios difusos en comparación con la versión serial más rápida obtenida que demora aproximadamente 5 horas.

8.2 PROYECCIONES Y TRABAJO FUTURO

8.2.1 Mejoras al modelo de difusión

Una de las desventajas del modelo de difusión es la aproximación de densidades cercanas a la fuente, ya que es calculada solo en el elemento que contiene a la fuente y a medida que se refina la malla el área del elemento que contiene a la fuente es menor y teóricamente las fuentes de radiación tienen un radio efectivo en que no sufren variación en su energía o tienen algún cambio de su dirección debido al fenómeno de dispersión.

Por lo cual las fuentes de fotones se pueden irradiar fotones desde más elementos de la geometría y no solo de un solo elemento finito. El problema de utilizar este radio efectivo, esta que al estimar la aproximación nodal en elementos que no contengan al punto central de la fuente los valores que son calculados por la aproximación lineal definida por el método de elementos finitos entregan valores incorrectos lo cual perturba la solución y la distribución isotrópica de la fuente de fotones.

Por lo tanto este debe ser un tema de estudio, donde posiblemente una tratamiento exclusivo a nivel de construcción de la malla en lugares cercanos a la fuente podría ser una alternativa o bien utilizar una fusión de métodos como la propuesta de (Tarvainen, 2006) donde realizan una aproximación de elementos finitos utilizando la ETR cercano a la fuente y una aproximación de AD en el resto del dominio.

Otro factor que puede ser tomado en consideración, es el cálculo del área efectiva en que se realiza la aproximación de densidad en los nodos, para lo cual se podría utilizar por ejemplo diagramas de Voronoi, en donde los puntos nodales sean los puntos de control del método, y de esta manera, multiplicar por el área que abarca cada nodo y así obtener una área nodal con menos margen de error al estimar la densidad de partículas en los nodos cuando se realiza la aproximación inicial FEM.

8.2.2 Implementación en tres dimensiones

Diseñar los módulos orientados a objetos para geometrías de tres dimensiones, en donde el esfuerzo sólo debe ser dedicado a los cálculos geométricos de elementos finitos en tres dimensiones y los aportes integrales de los elementos, ya que todo el proceso

numérico de obtención de los valores nodales a partir del sistema de ecuaciones es el mismo para cualquier dimensión.

8.2.3 Pruebas en otras plataformas paralelas

Es importante señalar que en hardware de memoria compartida, ya se sea en computadores de arquitecturas Dual Core o Quad Core siempre se obtuvieron Speedup crecientes con 2 y 4 procesos respectivamente, por lo cual es posible que el algoritmo LSQR es óptimo en este tipo de arquitecturas, lo cual es beneficioso por el costo de este tipo de hardware que es mucho menor que la construcción de un clúster de computadores. Sin embargo, esta arquitectura solo es eficiente cuando se utilizan geometrías de no más de 66000 nodos aproximadamente. Si se desean realizar aproximaciones de la grilla en problemas extremadamente grandes donde se necesita mucha memoria a partir de la solución FEM un clúster es el hardware más adecuado para resolver el problema, ya que solo existe una comunicación inicial y después cada procesador solo se preocupa de resolver su sección del algoritmo y enviar los resultados finales y este procedimiento tiene un Speedup lineal con mínima comunicación.

8.2.4 Diseño de módulos gráficos y de análisis de imágenes

Construir módulos de computación grafica y análisis de imágenes, donde a partir de imágenes médicas se genere una malla geométrica utilizando puntos de control y texturas que permitan identificar a los distintos tejidos u órganos y de esta manera generar mapas de difusión con datos reales.

8.3 TRABAJO MULTIDISCIPLINARIO

Si en Chile se desea construir y disponer de este tipo de tecnologías de detección y seguimiento de procesos patológicos relacionados con el cáncer u otros tipos de exámenes relacionados con Tomografía Óptica es necesario el trabajo conjunto de las áreas relacionadas (Medicina, Física, Bioquímica y Computación) además del apoyo gubernamental y el convencimiento como sociedad Chilena de que debemos generan nuestras propias tecnologías y no solo exportar teoría para después comprar dicha tecnología.

Bibliografía

Alexandrakis, G, Farrrel, T J y Patterson, M S. 1998. Accuracy of the diffusion approximation in determining the optical properties of a two-layer turbid medium. s.l. : Appl Opt, 1998, Vol. 31, págs. 7401- 7409.

Amaldi, E. 1959. The production and slowing down of neutrons. [ed.] S. Flugge. *Encyclopedia of Physics.* Berlin : s.n., 1959, Vol. 2.

Arridge, R. S. y Hebden, J. C. 1997. Optical imaging in medicine : II. Modelling and reconstruction. 1997, Vol. 42, págs. 841-853.

Arridge, S R. 1999. *Optical tomography in medical imaging.* Londres : Department of Computer Science, University College, 1999. R41–R93.

Bevilacqua, F, y otros. 1999. and C. Depeursinge, In Vivo Local Determination of Tissue Optical Properties: Applications to Human Brain. s.l. : Appl. Opt, 1999, Vol. 38, págs. 4939-4950.

Buyya, R. 1999. *High Performance Cluster Computing: Architectures and Systems.* s.l. : Prentice Hall, 1999. Vol. I.

Buyya, R. 2000. *High Performance Cluster Computing: Programming and Aplications.* s.l. : Prentice Hall, 2000. Vol. II.

Cercignani., C. 1988. The Boltzmann Equation and Its Applications. New York : Springer-Verlag, 1988.

Chandrasekhar, S. 1950. Radiative Transfer. Londres : Oxford University Press, 1950.

Chaudhari, Abhijit J. 2006. *Hyperspectral and multispectral optical bioluminescence and fluorescence tomography in small animal imaging.* Faculty of the Graduate School, University of Southern California. 2006.

Chaudhuri, P. 1992. *Parallel Algorithms: Desing and Analysis.* s.l. : Prentice Hall, 1992.

Constanzo, J. 2006. *Métodos Iterativos para resolución de sistemas de ecuaciones lineales.* Valparaiso : s.n., 2006.

Dehghani, H, y otros. 2000. Optical tomography in the presence of void regions. s.l. : J Opt Soc Am A, 2000, Vol. 9.

Dorn, O. 2000. Scattering and absorption transport sensitivity functions for optical tomography. s.l. : Opt Express, 2000, Vol. 7, pp. 492-506.

Firbank, M, y otros. 1996. *An investigation of light transport through scattering bodies with non-scattering regions.* s.l. : Phys Med Biol, 1996. págs. 767-783. Vol. 41.

Hayakawa, C K, Hill, B Y and Joon, S. 2004. Use of the delta-P1 approximation for recovery of optical absorption, scattering, and asymmetry coefficients in turbid media. California : Optical Society of America, 2004, Vol. 43, 24.

Hielscher, A H, Alcouffe, R E y Barbour, R L. 1998. *Comparison of finite-diference transport and diffusion calculations for photon migration in homogeneous and heterogeneous tissues.* s.l. : Phys Med Biol, 1998. págs. 1285-1302. Vol. 43.

Higham, N J. 2008. *Cholesky Factorizacion.* Manchester : The University of Manchester, 2008. ISSN:1749-9097.

Ishimaru, A. 1978. Wave Propagation and Scattering in Random Media. 1978, Vol. 1.

Jacques, S L. 1998. *"Light distributions from point, line and plane sources for photochemical reactions and fluorescence in turbid biological tissues,".* 1998. pp. 23-32. Vol. 1.

Jacques, S L y Prahl, S A. 1998. http://omlc.ogi.edu/classroom/ece532/class4/ssmc/. [En línea] 1998.

Jaramillo, J D y Correa, F J. 2006. Métodos directos para la solución de sistemas de ecuaciones lineales simétricos, indefinidos, disperso y de gran dimensión. Medellin : s.n., 2006.

Khan, T y Jiang, H. 2003. *A new difusion aproximation to the radiative transfer equation for scattering media with spatially varying refractive indices.* s.l. : J Opt A;Pure Appl Opt, 2003.

Kim, A D. 2004. *Transport theory for light propagation in biological tissue.* s.l. : Opt Soc Am, 2004. pp. 820-827.

Massoud, Tarik F and Gambhir, Sanvij S. 2003. *Molecular imaging in living subjects: seeing fundamental biological processes in a new light.* California : GENES & DEVELOPMENT, 2003. pp. 545-580. Vol. 17. 0890-9369/03.

Métodos directos para la solución de sistemas de ecuaciones lineales simétricos, indefinidos, dispersos y de gran dimensión.

Paige, C. 1982. *LSQR: Sparse Linear Equations and Least Square Problems.* s.l. : ACM Trans Math, 1982. Vol. 2.

Patterson, M. S. 1991. *The propagation of optical radiation in tissue I. Models of radiation transport and their application.* 1991. pp. 155-168. Vol. 6.

Reddy, J N. 1993. *An Introduction to the Finite Element Method.* s.l. : McGraw-Hill Inc, 1993.

S. Fantini, M.A. Franceschini, and E. Gratton. 1997. Effective source term in the diffusion equation for photon transport in turbid media. s.l. : Appl Opt, 1997, Vol. 1, págs. 156-163.

Sterenborg, H J, y otros. 1989. The spectral dependence of the optical properties of human brain, . Sci.,. s.l. : Lasers Med, 1989, Vol. 4, págs. 221-227.

Tarvainen, T. 2006. *Computational Methods for Light in Transport in Optical Tomography.* [ed.] Ph.D. Professor Pertti Pasanen. Kuopio : Kuopio University Publications C. Natural and Environmental Sciences, 2006.

Wang, L. 1998. Monte Carlo Modeling of Light Transport. Texas : s.n., 1998.

Zee, P V. 1993. *Measurement and modelling of the optical propieties of human tissue in the near infrared.* University of London. Londres : s.n., 1993.

Zweifel, K.M. 1967. Linear Transport Theory. s.l. : Addison-Wesley, 1967.