03063

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

U.A.C.P. y P. I.I.M.A.S.

GENERACION DE MALLAS USANDO MAPEO ORTOGONAL Y ALGUNAS APLICACIONES

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACION

P R E S E N T A:

LUIS MIGUEL DE LA CRUZ SALAS

Director de Tesis: Dra. Clara Garza Hume

México, D.F.

1999

TESIS CON
FALLA DE ORIGEN

273132

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

ESTA TESIS NO SALE
DE LA BIBLIOTECA

A la memoria de Juanita Rojas
descanse en paz...

A Cesareo y Julia
gracias por dejarme vivir...

A mis lindas chicas
Dianita Michelle, Angélica, Edith, Norma
y como olvidar a Perlita...

Agradecimientos

Es indudable que este trabajo fue llevado a cabo gracias a la ayuda de muchas personas, a todas ellas gracias por su tiempo y apoyo. Deseo agradecer principalmente a la Dra. Clara Garza por el tiempo dedicado a esta tesis, tomó tiempo pero valió la pena. Gracias también a mis sinodales que amablemente se tomaron el tiempo para revisar mi trabajo, sus opiniones fueron de gran ayuda.

También agradezco a mis compañeros de trabajo del Laboratorio de Visualización por soportar mis penosas historias del desarrollo de este trabajo.

Y si alguien me faltó, se la debo para el doctorado...

Generación de Mallas Usando Mapeo Ortogonal y Algunas Aplicaciones

Luis M. de la Cruz Salas

8 de octubre de 1999

Índice General

1 Introducción 5
- 1.1 Aplicaciones . . . 8
- 1.2 Complejidad geométrica del dominio . . . 9
- 1.3 Avances en la generación de mallas . . . 12
- 1.4 Técnicas de programación . . . 14

2 Técnicas de Generación de Mallas 17
- 2.1 Mallas estructuradas . . . 17
  - 2.1.1 Generación algebraica de mallas . . . 18
  - 2.1.2 Generación de mallas basadas en EDP . . . 19
  - 2.1.3 Generación variacional . . . 21
- 2.2 Mallas no estructuradas . . . 22
- 2.3 Mallas adaptivas . . . 23

3 Mapeo Ortogonal 27
- 3.1 Ecuaciones del mapeo ortogonal . . . 27
  - 3.1.1 Coordenadas generalizadas . . . 28
  - 3.1.2 Restricción a coordenadas ortogonales . . . 31
  - 3.1.3 Ecuación covariante de Laplace . . . 32
- 3.2 Descomposición del mapeo ortogonal . . . 35
- 3.3 Formulación de integral de frontera . . . 41
- 3.4 Esquema de generación de mallas ortogonales . . . 43

4 Metodología de Programación 47
- 4.1 La entropía del sistema . . . 47
- 4.2 Tecnología orientada a objetos . . . 50
  - 4.2.1 Conceptos básicos de TOO . . . 51
- 4.3 Análisis y diseño . . . 53

4.3.1 Modelo de objetos 54
4.3.2 Diseño del sistema 60
4.3.3 Reuso de bibliotecas numéricas 61

5 Resultados y Aplicaciones **63**
5.1 Prueba y calibración del software 63
5.2 Ejemplos de mallas ortogonales 66
5.3 Aplicaciones 70

6 Conclusiones **77**

Capítulo 1

Introducción

Muchos de los fenómenos que suceden en la naturaleza y en la vida cotidiana pueden ser descritos mediante modelos matemáticos. La complejidad de dicho modelo depende del fenómeno que se desea describir. Algunos de estos modelos se pueden escribir en términos de ecuaciones diferenciales parciales, y en la gran mayoría la solución exacta a estas ecuaciones no existe hasta nuestros días. La alternativa es usar métodos numéricos para aproximar la solución. Dados los avances en la tecnología de las computadoras digitales, muchos problemas, cuya solución analítica no se conoce, han podido ser simulados mediante métodos numéricos con bastante precisión. El primer paso en una simulación es pasar del dominio continuo al dominio discreto. Esto implica que se tienen que distribuir una serie de puntos sobre el dominio de estudio, donde la solución numérica será calculada. La distribución de dichos puntos es un problema complicado, pues es necesario conocer de antemano el comportamiento del fenómeno bajo estudio para poder hacer una distribución adecuada de los puntos. Al proceso de distribución de puntos y la forma en que ellos se conectan se conoce como la generación de la malla.

La generación automática de mallas se ha convertido en una herramienta común para resolver numéricamente ecuaciones diferenciales parciales sobre regiones de forma irregular. Lo anterior es mayormente visto en aplicaciones de dinámica de fluidos, en donde se han desarrollado diferentes técnicas para la construcción de mallas, aunque estos procedimientos son igualmente aplicables en problemas de otros campos de la ciencia que involucran soluciones numéricas.

La generación numérica de mallas puede pensarse como un procedimiento para distribuir ordenadamente a un conjunto de observadores sobre un

dominio físico, de tal manera que la comunicación entre ellos sea eficiente y todos los fenómenos físicos sobre el dominio completo puedan ser representados con suficiente exactitud por este número finito de observadores. Una malla es una estructura sobre la cual se construirá una solución numérica. Esta solución aproxima con cierta exactitud el comportamiento del fenómeno bajo estudio. Al igual que cualquier diseño de estructuras, la construcción de una malla requiere diferentes consideraciones, como que la carga sobre ella este bien distribuida en todos sus puntos y que el número de ellos sea económico. Además, dicha distribución se verá influida por la configuración geométrica del dominio físico y por la solución que será calculada sobre la malla. Dado que los recursos son limitados en cualquier solución numérica, es responsabilidad de la generación numérica de la malla hacer el mejor uso del número de puntos disponible.

El rango de aplicación de las mallas es bastante amplio. Es posible usar una malla para estudiar numéricamente fenómenos en la micro escala, por ejemplo estudios de semiconductores, y también se pueden aplicar en estudios en la macro escala, por ejemplo estudios del clima en la superficie de la tierra.

Debido a este amplio rango de aplicación y a las formas irregulares que se presentan en la naturaleza, es necesario construir mallas sobre dominios irregulares que permitan aplicar métodos numéricos para aproximar la solución del problema.

Particularmente, el cálculo de flujos alrededor de formas irregulares como ductos, aviones, automóviles, etc., involucra fronteras computacionales que no coinciden con las líneas coordenadas en el espacio físico. En el caso de los métodos de diferencias finitas, la imposición de condiciones de frontera para tales problemas ha requerido de aplicar complicadas fórmulas de interpolación de los datos sobre mallas locales y regularmente este proceso provoca una pérdida en la precisión numérica de la solución computacional.

Las dificultades mencionadas arriba han motivado la introducción de un mapeo o transformación del espacio físico, (x, y) en 2D, a un espacio coordenado curvilíneo generalizado, (ξ, η). El espacio generalizado es construido de tal manera que las fronteras del dominio físico coincidan con las líneas coordenadas en el espacio generalizado, véase figura 1.1.

El uso de coordenadas generalizadas implica que una región distorsionada en el espacio físico sea mapeada en una región rectangular en el espacio de coordenadas generalizado. Las ecuaciones gobernantes son expresadas en términos de las coordenadas generalizadas como variables independientes y la discretización es realizada en el espacio de coordenadas generalizado. De esta

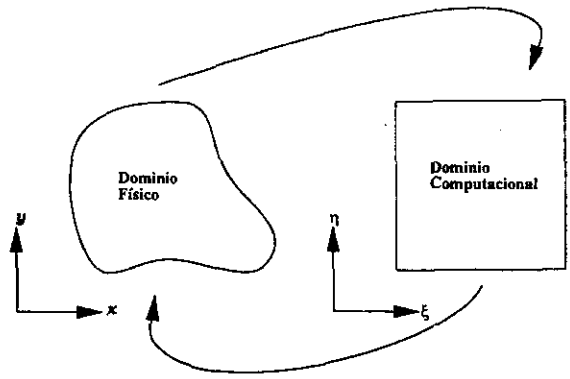

Figura 1.1: Transformación del espacio físico al espacio generalizado.

forma, los cálculos se llevan a cabo efectivamente en el espacio generalizado.

Algunas de las ventajas de esta transformación son: se pueden aplicar métodos de diferencias finitas para resolver problemas en dominio irregulares sin tener que hacer interpolaciones complicadas en las fronteras, es fácil refinar la malla en zonas estratégicas del dominio y también es fácil cambiar de dominio.

El concepto de coordenadas generalizadas sugiere posibilidades adicionales. Primero, la malla computacional en el espacio de coordenadas generalizado puede corresponder a una malla en movimiento en el espacio físico, lo cual sería apropiado para flujos no estacionarios con fronteras en movimiento.

El mapeo entre los espacios físico y generalizado, permiten que las líneas de coordenadas sean concentradas en partes del dominio físico donde se esperan gradientes grandes. Si las regiones de gradientes grandes cambian con el tiempo, por ejemplo en el caso de la propagación de ondas de choque, la malla física puede ser ajustada con el tiempo para asegurar que la malla local sea lo suficientemente fina para obtener soluciones con alto grado de precisión.

Aún en el caso de flujos estacionarios, es posible regenerar la malla durante el cálculo de la solución. El uso de coordenadas generalizadas permite el desarrollo de una solución iterativa donde la malla se va ajustando al comportamiento del flujo en todas las partes del dominio con la misma exactitud. El resultado es un uso más adecuado de los puntos de la malla.

Por otro lado, el uso de coordenadas generalizadas introduce algunas complicaciones. Primero, es necesario considerar la forma que toman las

ecuaciones gobernantes en este tipo de coordenadas. Dichas ecuaciones contendrán términos adicionales que definen el mapeo entre el espacio físico y el espacio generalizado. Estos términos adicionales (parámetros de la transformación), se escriben usando derivadas, por ejemplo $\partial x/\partial \xi$, y usualmente necesitan ser discretizadas, lo cual introduce una fuente de error adicional en la solución. Dichos errores pueden ser manejados de tal manera que su influencia sobre la solución numérica sea mínima, y así se pueden obtener soluciones bastante precisas.

Para generar sistemas coordenados generalizados o curvilíneos, existen diferentes métodos, uno de los más usados es a través de la solución de ecuaciones diferenciales parciales no lineales.

1.1 Aplicaciones

Gracias a los recientes desarrollos en la tecnología de la computación, tanto en hardware como en software, ha sido posible resolver distintos problemas por medio de simulaciones en computadora. Principalmente, se han podido resolver problemas bastante complicados de la dinámica de fluidos.

La dinámica de fluidos computacional (o CFD por sus siglas en inglés) es una nueva rama de estudio que ha surgido gracias a este desarrollo tecnológico, véase [10]. Esta rama de la dinámica de fluidos complementa los estudios teóricos y experimentales dando una alternativa a bajo costo para la simulación de flujos reales. Es en este nuevo campo de la ciencia, donde ha surgido la necesidad de la construcción automática de mallas con fronteras irregulares para simular flujos en diferentes problemas. La CFD, ofrece además, un medio para comprobar los avances teóricos, cuya comprobación experimental es muy difícil o imposible. Además, con la CFD es posible adicionar y eliminar términos en las ecuaciones, lo cual permite probar modelos teóricos y sugerir nuevos caminos para las exploraciones teóricas y experimentales.

En otras áreas de la ciencia existen desarrollos similares a los generados en la CFD, en donde los tratamientos numéricos y de generación de mallas son similares. Por esta razón, mucho de los desarrollos que se dan en la CFD son aplicables en otras áreas.

A pesar de las modelaciones numéricas tan exactas que se han sido logradas en nuestros días, existen ciertas dificultades en los estudios numéricos. Los cuatro principales obstáculos que se tienen que superar en una simulación

numérica son:

1. la complejidad física del problema;
2. la complejidad geométrica del dominio;
3. la limitada precisión, estabilidad y economía de las soluciones numéricas;
4. el post-procesamiento o visualización de las grandes cantidades de datos obtenidas por la solución numérica.

En esta tesis nos enfocaremos principalmente en el punto 2, que es un problema bastante complejo y difícil de superar, y es además, el primer paso cuando se realiza una simulación numérica.

En una simulación, las ecuaciones diferenciales que gobiernan el fenómeno bajo estudio, deben ser discretizadas esencialmente reemplazando las derivadas por diferencias, las cuales se calculan sobre un número finito de puntos conocidos como nodos. Estos nodos generalmente están distribuidos sobre una malla cubriendo el dominio de estudio.

1.2 Complejidad geométrica del dominio

Las ecuaciones diferenciales parciales que modelan el problema son fácilmente discretizadas si se escriben en coordenadas Cartesianas, cilíndricas o esféricas. Desafortunadamente, el rango de problemas que se dan en la práctica, donde las configuraciones pueden ser descritas en ese tipo de sistemas coordenados, es muy limitado y estos casos ya han sido sujeto de muchos estudios numéricos y experimentales. Las geometrías que se dan en la realidad son muy irregulares de tal manera que es necesaria una aproximación distinta. Las formas irregulares de los dominios donde diferentes fenómenos toman lugar, pueden tratarse en una de las siguientes formas:

- Usando coordenadas regulares (Cartesianas, polares, esféricas, etc.) con una aproximación escalonada en las fronteras curvas o haciendo una composición de dos o más mallas que se ensamblen perfectamente o se sobrepongan en las fronteras.

  Ventajas:

1. La generación de la malla en este tipo de coordenadas es bastante simple.
2. Este tipo de tratamiento utiliza las ecuaciones más simples (por ejemplo en forma Cartesiana) cuya discretización es fácil y directa (veáse [19] y [10]).

Desventajas:

1. El tratamiento de las fronteras irregulares es extremadamente complicado y trae consigo, no solo inconveniencia, sino también inexactitud en la solución.
2. La distribución de las líneas de la malla no es simple. La introducción de mallas finas en regiones de gradientes abruptos (cerca de las fronteras por ejemplo) resulta en refinamientos innecesarios en otras zonas lo que incrementa el uso de memoria y el tiempo de cálculo.
3. Cuando las fronteras del dominio del problema consisten de diferentes tipos de geometrías "regulares", como círculos y rectas, entonces es posible, por ejemplo, usar coordenadas polares alrededor de una frontera circular y rectilíneas en los lugares restantes, lo que ocasiona sobreposiciones.
4. El punto anterior trae como consecuencia que se tengan que escribir programas diferentes para las diferentes regiones de la malla y se necesitan realizar interpolaciones intensivas para definir los valores en las fronteras donde hay sobreposición, véase [10].

- Usando elementos triangulares para aproximar de manera más exacta en las fronteras irregulares. En 3D se utilizan tetraedros. En este caso, se dice que el dominio se triangulariza y luego se aplican métodos de elemento finito para obtener la solución del problema.

  Ventajas:

  1. Se puede aplicar sobre cualquier tipo de geometría.
  2. Pueden usarse elementos más complejos, como pentaedros, para aproximar mejor las fronteras del dominio y reducir el error en la solución numérica, véase [11].

  Desventajas:

1. La complejidad matemática de los métodos de elemento finito hace que los códigos sean bastante complejos y difíciles de modificar.
2. Este tipo de métodos son no conservativos, lo que implica que una solución puede ser muy precisa, pero posiblemente irreal (no cumple con las leyes físicas planteadas por el problema).
3. Requieren mayor uso de memoria y más cantidad de cálculos para simular un problema respecto a otras técnicas.

- Usando coordenadas curvilíneas o también conocidas como coordenadas generalizadas. Este tipo de coordenadas puede ser ortogonal o no ortogonal, optimizando la suavidad o el área, dependiendo del tipo de problema que se quiera resolver y de la complejidad del mismo. La idea de estos sistemas de coordenadas curvilíneos es mapear el dominio físico irregular a un dominio regular simple en donde se resolverán las ecuaciones gobernantes. En estos sistemas, una línea de frontera irregular representa una línea coordenada del sistema.

  Ventajas:

  1. Se puede aplicar a cualquier tipo de geometría.
  2. La solución del problema bajo estudio se puede obtener mediante métodos simples, como diferencias finitas, con alta precisión.
  3. Se puede refinar la malla en los lugares de interés. Este refinamiento también puede hacerse durante el cálculo de la solución para resolver con mayor precisión gradientes abruptos.
  4. El uso de memoria y al cantidad de cálculos son menores respecto a las otras técnicas.
  5. En el caso de sistemas ortogonales, los términos que se adicionan a las ecuaciones son mínimos por lo que la discretización de éstas es muy simple.

  Desventajas:

  1. La generación de sistemas curvilíneos implica la solución de ecuaciones diferenciales parciales no lineales.
  2. Dada la no linealidad, se tiene que hacer uso de métodos iterativos lo cual consume mucho tiempo, especialmente en tres dimensiones.

3. Una vez generado el sistema curvilíneo, hay que adicionar términos a las ecuaciones que gobiernan el problema bajo estudio, lo que complica aún más el problema.

En este trabajo se hace uso de este último tratamiento y se generan mallas ortogonales mediante una técnica que se describirá en el capítulo 3. La ventaja del método que se utiliza en esta tesis es que es no iterativo, lo cual reduce en gran medida el tiempo de generación de las mallas.

1.3 Avances en la generación de mallas

En los últimos 20 años se han realizado avances en la generación de mallas curvilíneas tanto ortogonales como no-ortogonales, véase [29]. Aunque mucho de este desarrollo proviene de la dinámica de fluidos, las técnicas son igualmente aplicables en estructuras, electromagnetismo y otras áreas que involucran soluciones numéricas. Con sistemas coordenados generados para mantener las líneas coordenadas (superficies en 3D) coincidentes con las fronteras, es posible construir códigos basados en diferencias finitas o volumen finito, los cuales pueden ser aplicados a configuraciones arbitrarias sin la necesidad de procedimientos especiales sobre las fronteras. Aún cuando las fronteras estén en movimiento, el uso de tales coordenadas permite que todos los cálculos sean realizados sobre una malla fija regular en el dominio transformado.

Recientemente se ha hecho un progreso considerable en el desarrollo de sistemas coordenados curvilíneos y una gran variedad de métodos han sido presentados en la literatura, los cuales mejoran velocidad de cálculo, adaptabilidad de las líneas de la malla, ortogonalidad, suavidad, entre otras cosas. Una inspección de ésta área de estudio se puede encontrar en Thompson *et al.* [27] y en Thompson [28]. Algunas de las ideas básicas de la construcción y uso de sistemas coordenados curvilíneos para la solución de ecuaciones diferenciales parciales, se discuten ampliamente en Thompson *et al.* [29]. Por otro lado, Thacker [26], este hace una revisión acerca de los métodos para construcción de mallas que son más convenientes para aplicaciones de elemento finito. Se pueden encontrar también estudios de geometría diferencial y análisis tensorial aplicables a la generación de sistemas coordenados curvilíneos, véase Warsi [30].

Una de las técnicas más usadas para generar coordenadas curvilíneas es a través de la solución de ecuaciones diferenciales parciales. Este camino

es el que han seguido la mayoría de los estudios realizados hasta ahora. Por ejemplo, Ryskin *et al.* [22] hace un estudio de generación de mallas utilizando ecuaciones diferenciales parciales elípticas, en donde transforma el dominio físico irregular a un dominio computacional regular y hace unos ajustes sobre esta transformación para obtener mallas ortogonales. Describe además dos métodos para implementar la transformación.

Estudios más recientes de generación de mallas usando mapeos ortogonales, donde se muestran nuevas direcciones y avances en la generación de mallas, se pueden encontrar en Duraiswami y Prosperetti [8] y en Ça [9], ambos estudios se basan en el trabajo de Ryskin y Leal [22]. Estos y otros estudios muestran diferentes métodos para resolver las ecuaciones diferenciales parciales no lineales de la transformación de coordenadas. La generalidad es que todos utilizan métodos iterativos para obtener una solución a dichas ecuaciones.

Por otro lado, Kang y Leal [14] desarrollaron una técnica para la generación de mallas ortogonales usando un método no iterativo. La idea del método se basa en la descomposición del mapeo ortogonal en dos mapeos sucesivos: un mapeo conforme entre el dominio físico y un dominio auxiliar, y un mapeo ortogonal (más simple) entre el dominio físico y el domino computacional. El primer mapeo es utilizado para encontrar la función de distorsión en todo el dominio físico. Una vez conocida la función de distorsión, el segundo mapeo consiste en resolver un par de ecuaciones diferenciales parciales elípticas pero lineales, cuya solución depende ahora solamente de las condiciones a la frontera. El problema puede resolverse en un solo paso, por lo que es no iterativo. En este método, sólo se puede especificar la correspondencia de los puntos en dos fronteras del dominio. Además solo funciona en dominios simplemente conexos (sin agujeros). Sin embargo, es posible aplicar el método a dominios que no sean simplemente conexos, dividiendo el dominio en partes sin agujeros. A cada parte se aplica el método y luego se unen. Dadas las ventajas que se encuentran en este último desarrollo, en esta tesis se seguirá el trabajo de Kang y Leal para construir un sistema de generación de mallas ortogonales. Existe otro desarrollo, basado en el trabajo de Kang y Leal que puede especificar la correspondencia de los puntos sobre tres lados de la frontera y además es más fácil refinar la malla generada. Este desarrollo fue presentado por Oh y Kang [15], sin embargo esta última modificación no se verá en esta tesis por lo que se recomienda revisar la referencia.

1.4 Técnicas de programación

Tradicionalmente la forma en que se codifican los métodos numéricos en problemas de simulaciones computacionales es usando la metodología de programación estructurada, y se utilizan lenguajes simples como Fortran 77, C e inclusive Pascal. Las dificultades con este tipo de programación se notan cuando el problema bajo estudio tiene que ser modificado. Las modificaciones a este tipo de programas causan muchos conflictos debido a la enorme cantidad de variables globales y parámetros que se tienen que pasar a cada subrutina. En algunas ocasiones, cuando las modificaciones deben ser mayores, es preferible construir un programa nuevo. Aún cuando los métodos que se usan para resolver problemas de diferentes áreas son muchas veces los mismos, es complicado reusar el código sobre distintas aplicaciones.

Una solución a los problemas que surgen en la forma de programación estructurada, puede encontrarse en la metodología de Programación Orientada a Objetos (POO). La POO tiene las ventajas del encapsulamiento de la información, la herencia, el polimorfismo, entre otras, y es muy fácil reusar y modificar los códigos para resolver distintos problemas.

Los primeros reportes de la eficiencia de programas construidos usando POO para problemas de simulación numérica, indicaban que el rendimiento de este tipo de códigos era mucho menor que los de aquellos construidos con Fortran 77, por lo que al principio se optó por evitar construir códigos orientados a objetos para problemas numéricos. Sin embargo, en los últimos 10 años se ha hecho una investigación exhaustiva acerca de la aplicación de la POO en problemas numéricos, y se han logrado obtener eficiencias muy cercanas a las que se alcanzan con lenguajes como Fortran 90 (de los más eficientes para análisis numérico), veáse [4] y [32]. Si a lo anterior le sumamos todas las cualidades que ofrecen los lenguajes orientados a objetos, entonces parece ser que estos lenguajes pueden ser una excelente alternativa para realizar simulaciones numéricas y obtener buenos resultados en cuanto a eficiencia.

En este trabajo se desarrollan una serie de bibliotecas (clases) numéricas para la generación de mallas ortogonales sobre geometrías irregulares. Se utiliza un método no iterativo presentado en [14]. Las bibliotecas se construyen usando el lenguaje C++ que posee las herramientas necesarias para programar con la metodología de orientación a objetos y se utilizan algunas clases optimizadas para la solución de sistemas lineales. Finalmente se presentan varios ejemplos de mallas construidas con este software y la solución de la

ecuación de Laplace en un dominio semicircular donde la solución analítica se conoce.

Capítulo 2

Técnicas de Generación de Mallas

Sistemas físicos continuos, tales como el flujo de aire alrededor de un avión, la concentración de esfuerzos en una presa, el campo eléctrico en un circuito integrado o la concentración de especies químicas en un reactor, son generalmente modelados usando ecuaciones diferenciales parciales. Para realizar simulaciones de estos sistemas en una computadora, las ecuaciones necesitan ser discretizadas sobre un número finito de puntos del espacio (y tiempo) donde variables como velocidad, densidad, campo eléctrico, entre otras, serán calculadas. Los métodos usuales de discretización, diferencias finitas, volúmenes finitos y elemento finito, utilizan puntos vecinos para calcular derivadas y de esta manera se tiene el concepto de *malla* sobre la cual los cálculos se realizan. Existen principalmente dos tipos de mallas que pueden caracterizarse por la forma en que se conectan los nodos. En las secciones que siguen se discutirán en forma rápida los principales tipos de mallas que se conocen y la forma en que éstas son generadas.

2.1 Mallas estructuradas

Las mallas estructuradas tienen una conectividad regular, lo cual significa que cada punto tiene el mismo número de vecinos (en algunos casos un número pequeño de puntos puede tener diferente número de vecinos). Los puntos de la malla pueden ser indexados (por 2 índices en 2D, 3 índices en 3D) y los vecinos de cada punto pueden ser calculados fácilmente, por ejemplo:

los vecinos del punto (i, j) están en $(i + 1, j)$, $(i - 1, j)$, etc. En dominios rectangulares es simple construir una malla. El problema aparece cuando se requiere construir mallas estructuradas en dominios irregulares. Generalmente las mallas se construyen de tal manera que las fronteras coincidan con las *líneas coordenadas*, es decir se usan coordenadas curvilíneas. Lo anterior produce como resultado soluciones muy precisas cerca de la frontera. Para flujo de fluidos este tipo de mallas permiten una fácil aplicación de modelos de turbulencia, los cuales usualmente necesitan mallas alineadas con las fronteras.

Para dominios simplemente conexos (que no tienen "agujeros") e irregulares, existen diferentes métodos para la construcción de mallas. En estos casos el dominio computacional es un rectángulo en dos dimensiones o un paralelepípedo en tres dimensiones. Aquí es necesario definir un mapeo uno a uno del espacio computacional (generalizado) al espacio físico y que además sea invertible. Algunos de los métodos que actualmente se usan son:

- generación algebraica de mallas, donde los puntos interiores de la malla se encuentran interpolando las fronteras del dominio físico, véase por ejemplo Fletcher [10],
- métodos donde los puntos interiores se encuentran resolviendo una ecuación diferencial parcial, Thompson *et al.* [29],
- generación variacional de mallas, donde algún funcional, por ejemplo la suavidad, es maximizado o minimizado, ver [23] para mayores detalles.
- otros métodos, mapeo conforme, mapeos cuasiconformes, etc.

2.1.1 Generación algebraica de mallas

La técnica de mapeo algebraico interpola los datos de la frontera para generar los puntos interiores de la malla. El método de interpolación que se usa es conocido como interpolación transfinita (TFI).

En dos dimensiones (la extensión a tres dimensiones es directa) se toma el cuadrado unitario $[0, 1] \times [0, 1]$ de coordenadas ξ y η, y el dominio físico con coordenadas x y y. Para generar la malla en el espacio físico, se genera una malla en el cuadrado unitario y luego se mapea al dominio físico. Existen dos requerimientos para que el mapeo funcione correctamente:

- debe ser uno a uno,

- las fronteras del espacio computacional deben mapearse en las fronteras del espacio físico.

Este método tiene algunos problemas. El mapeo propagará las singularidades de las fronteras (las esquinas) al interior del dominio, lo cual no es bueno para simulaciones de flujo de fluidos. Un problema más serio es que si el mapeo no es uno a uno, los puntos interiores calculados pueden salirse del dominio físico. Lo anterior puede corregirse reparametrizando las fronteras o adicionando restricciones a las líneas interiores del dominio.

A pesar del problema mencionado antes, la generación de mallas vía TFI es muy rápida y es un método muy efectivo para generara mallas en 3D. También es posible usar un generador basado en ecuaciones diferenciales parciales para suavizar la malla producida por TFI.

2.1.2 Generación de mallas basadas en EDP

La idea atrás de este método es definir ecuaciones diferenciales parciales para las coordenadas del espacio físico en términos de las coordenadas del espacio computacional, y luego resolver dichas ecuaciones sobre una malla en el espacio computacional para crear una malla en el espacio físico. Generalmente se utilizan ecuaciones diferenciales parciales del tipo elíptico, véase [29]. Este tipo de ecuaciones es conveniente para dominios con frontera cerrada (para dominios no acotados se utiliza una frontera ficticia a gran distancia). Las alternativas son ecuaciones del tipo hiperbólico y parabólico para dominios no acotados, véase [29].

La clasificación de las ecuaciones diferenciales parciales de segundo orden, se basa en los coeficientes de los términos principales de:

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu + G = 0.$$

que es la forma general de las ecuaciones diferenciales parciales de segundo orden para una función $u(x, y)$. Cuando los coefientes son función solamente de x y y se dice que la ecuación es cuasi-lineal. Entonces, si $B^2 - 4AC < 0$ la ecuación es **elíptica**, si $B^2 - 4AC = 0$ la ecuación es **parabólica** y si $B^2 - 4AC > 0$ la ecuación es **hiperbólica**.

En esta tesis se resuelve un par de ecuaciones elípticas para construir mallas curvilíneas ortogonales. Más adelante se explicará en detalle el método.

Estos métodos inician con una ecuación diferencial parcial para el espacio de coordenadas computacional (η, ξ) en términos del espacio de coordenadas físico (x, y) (la extensión de estos métodos a 3D es directa).

El uso de una ecuación diferencial parcial elíptica para generar los puntos interiores de la malla trae ciertas ventajas. Primero, la malla tendrá cambios suaves aún si la frontera del dominio tiene discontinuidades en la pendiente (si hay esquinas). En contraste, si se utiliza una ecuación hiperbólica estas discontinuidades serán propagadas al interior de la malla.

En el espacio computacional y en dos dimensiones, las ecuaciones más generales para la generación de mallas curvilíneas se escriben de la siguiente manera (véase Fletcher [10]):

$$\alpha\frac{\partial^2 x}{\partial\xi^2} - 2\beta\frac{\partial^2 x}{\partial\xi\partial\eta} + \gamma\frac{\partial^2 x}{\partial\eta^2} + \delta\left(P\frac{\partial x}{\partial\xi} + Q\frac{\partial x}{\partial\eta}\right) = 0, \tag{2.1}$$

$$\alpha\frac{\partial^2 y}{\partial\xi^2} - 2\beta\frac{\partial^2 y}{\partial\xi\partial\eta} + \gamma\frac{\partial^2 y}{\partial\eta^2} + \delta\left(P\frac{\partial y}{\partial\xi} + Q\frac{\partial y}{\partial\eta}\right) = 0, \tag{2.2}$$

donde $\alpha = g_{22}$, $\beta = g_{12}$, $\gamma = g_{11}$ y $\delta = g$, g_{ij} son elementos del tensor métrico y g representa al determinante del mismo tensor. La definición de este tensor se verá más adelante.

En este tipo de problemas es necesario especificar las condiciones a la frontera para que el problema este bien determinado. En este caso las condiciones de frontera serán las coordenadas de los puntos de la frontera del dominio físico.

Es posible generar mallas ortogonales a partir de la especificación de las coordenadas de los puntos de la frontera usando las ecuaciones 2.1 y 2.2. Esto se hace definiendo:

$$\frac{h_2}{h_1} = \frac{g_{22}^{1/2}}{g_{11}^{1/2}} = f(\xi, \eta), \tag{2.3}$$

donde a $f(\xi, \eta)$ se le conoce como la función de distorsión.

La función $f(\xi, \eta)$ puede seleccionarse sobre la frontera. Los valores interiores de $f(\xi, \eta)$ se encuentran por diferentes métodos, véase por ejemplo Ryskin y Leal [22].

En la práctica, inicialmente se construye una malla no ortogonal especificando $x(\xi, \eta)$ y $y(\xi, \eta)$ sobre las fronteras del dominio físico. Se evalúa $f(\xi, \eta)$

sobre la frontera y luego se calcula f en el interior usando alguna interpolación (TFI por ejemplo). Este proceso se repite hasta obtener una malla ortogonal o casi ortogonal.

El método usado en este trabajo tiene la particularidad de ser no iterativo y se discutirá en detalle en el capítulo 3.

También es posible utilizar ecuaciones hiperbólicas o parabólicas para generar mallas, véase [29]. Las ecuaciones diferenciales hiperbólicas tienen las siguientes características:

- producen mallas ortogonales
- propagan las discontinuidades de la frontera al interior
- se aplican en dominios exteriores (flujo alrededor de una avión)
- pueden fallar en fronteras cóncavas
- son rápidas, toman el mismo tiempo que una sola iteración del caso elíptico

Como se puede observar, este tipo de ecuaciones contiene más desventajas que ventajas respecto a las elípticas.

Por otro lado, los métodos parabólicos combinan la velocidad de los generadores hiperbólicos con el incremento de la suavidad de las líneas de la malla, sin embargo estos métodos no son muy utilizados.

Otro método muy conocido es el de multibloques que consiste en romper el dominio físico original en varias piezas de tal forma que cada una pueda mapearse en un rectángulo simple. Estas piezas o bloques, se unen para formar el dominio físico completo y deben cumplir con un cierto grado de continuidad en las caras. El proceso de descomposición en bloques no ha sido completamente automatizado y requiere de un considerable esfuerzo manual para producir mallas aceptables. El método de bloques puede ser automatizado usando uno de los métodos discutidos antes, véase [31].

2.1.3 Generación variacional

En este tipo de métodos se elige un funcional continuo que refleje las propiedades geométricas que se desea tenga la malla y entonces se resuelven las ecuaciones de Euler-Lagrange del funcional sujeto a las correspondientes condiciones de frontera.

Desde hace varios años se han desarrollado diferentes métodos variacionales que tratan de tener un control más directo de los puntos de la malla. Estos métodos construyen funcionales discretos, es decir, funciones que dependen directamente del tamaño de la malla y de las coordenadas de los puntos, por lo que transforman el problema de generación numérica de mallas en un problema de optimización no lineal de gran escala sin restricciones, que se puede resolver de manera eficiente usando algunos algoritmos que calculan el mínimo de una función de n variables.

Existe un grupo de investigadores dentro de la UNAM, que trabaja en este tipo de problemas, para obtener mayor información acerca de estos métodos se recomienda revisar la siguiente dirección:

http://www.mathmoo.unam.mx/unamalla/ .

2.2 Mallas no estructuradas

Las mallas no-estructuradas tienen una conectividad irregular, esto es, cada punto puede tener un número diferente de vecinos. Este tipo de mallas son usadas principalmente en métodos de elemento finito. Existe un amplio rango de formas posibles para los elementos finitos: triángulos, cuadrados, tetraedros, pentaedros, hexaedros, etc. Sin embargo, las formas que se usan para generar mallas automáticamente son triángulos en 2D y tetraedros en 3D.

El método de elemento finito (MEF) tiene ciertos requerimientos para una malla:

- La malla debe ser válida, es decir, sin agujeros, no se debe intersectar ella misma, etc. Aunque estos requerimientos son obvios, muchos esquemas de generación de mallas requieren una enorme verificación de estas condiciones.

- La malla debe acoplarse con la frontera del dominio, la cual también es una condición obvia pero algunos métodos (por ejemplo triangulación de Delaunay) no satisfacen esto y es necesario verificar y corregir las aristas o caras que crucen la frontera.

- La densidad de la malla debe ser controlable para permitir un equilibrio entre la exactitud de la solución y los cálculos en el tiempo.

- La densidad de la malla debe variar dependiendo de la exactitud deseada en cada lugar del dominio y esta variación deberá ser suave para reducir errores numéricos.

- En general, los ángulos interiores de los elementos de la malla, deben ser aproximadamente iguales (triángulos equiláteros, tetraedros regulares). Elementos altamente distorsionados (triángulos delgados por ejemplo) pueden ocasionar problemas de estabilidad numérica causados por errores de redondeo.

Triángulos y tetraedros pueden ajustarse bien a fronteras irregulares y permiten un progresivo cambio del tamaño del elemento sin una distorsión excesiva. Este tipo de elementos son convenientes para la generación de mallas que son utilizadas en conjunción con el método de elemento finito. Además, existen métodos completamente automáticos para la generación de mallas con este tipo de elementos. Sin embargo, tetraedros lineales no son muy buenos para el MEF y se necesita de muchos elementos para obtener resultados aceptables. Elementos cuadriláteros y hexaedros , son mucho mejores, pero la dificultad estriba en la generación automática de mallas con este tipo de elementos. Tetraedros cuadráticos tienen las mismas propiedades de los hexaedros cuadráticos y ellos permiten un mallado automático, por lo que este tipo de elementos se usa en la mayoría de las aplicaciones.

Existen varios métodos para la generación de mallas no estructuradas, entre estos métodos podemos mencionar: descomposición y mapeo, sobre posición de mallas regulares, métodos frontales, triangulación de Delaunay y métodos híbridos. Los más populares son los de triangulación de Delaunay y los frontales, cuya descripción detallada puede verse en [18] y [20] respectivamente.

2.3 Mallas adaptivas

En las secciones anteriores se mencionaron algunas técnicas para generar mallas estructuradas y no-estructuradas, especificando la densidad de las líneas de la malla. Esta densidad se selecciona para obtener una solución aceptable en exactitud y en tiempo. Es posible especificar ciertos aspectos de la malla al inicio de los cálculos, por ejemplo alta densidad cerca de las fronteras. Pero la solución puede desarrollar fenómenos de interés que no son predecibles con anterioridad. En los lugares donde se dan estos fenómenos

es necesario refinar la malla para que la solución sea exacta. En contraste, pueden existir áreas grandes del dominio donde la solución sea muy suave y por la tanto no es necesario hacer un refinamiento. Entonces, usando una malla que se adapte a la solución conforme ésta se va calculando puede producir soluciones muy precisas en tiempos óptimos.

El esquema de las mallas adaptivas tiene dos partes:

- Medición del error local y optimización (refinamiento o engrosamiento) de la malla si es necesario. La medida del error se debe basar en alguna técnica de extrapolación como la de Richardson. La densidad requerida de la malla puede calcularse usando la medida del error local y el orden local del método de discretización. La distribución requerida de la densidad de la malla equilibrará los errores numéricos en todo el dominio.

- Reconstruir la malla usando la distribución especificada de la densidad de la malla. El método utilizado depende de si el problema es estacionario o dependiente del tiempo. Para problemas estacionarios se usa un número pequeño de adaptaciones. En el caso de problemas que dependen del tiempo, un número grande de adaptaciones se requieren conforme la solución cambia. El método de regeneración de la malla debe ser rápido. La interpolación entre malla y malla debe ser muy precisa en casos de dependencia temporal; para problemas estacionarios esto es menos importante.

Existen tres métodos para refinar una malla: movimiento de los puntos de la malla, refinamiento local y regenerar la malla en su totalidad.

- El movimiento de los puntos de la malla mantiene la conectividad original, pero mueve los nodos. Este método se realiza con ajustes relativamente pequeños sin introducir celdas muy distorsionadas y además es rápido. Es conveniente para mallas estructuradas que requieren una conectividad fija, y también para problemas dependientes del tiempo donde la velocidad de refinamiento es importante.

- En el refinamiento local, las celdas de la malla son subdivididas. Las celdas vecinas deberán tener el mismo grado de subdivisión para mantener la validez de la malla. Este método es rápido y es ideal para problemas con dependencia temporal. La conectividad de la malla

cambia con cada modificación de ésta, por lo que este método es poco conveniente para mallas estructuradas.

- Regeneración de la malla completa usando parámetros nuevos para la densidad de líneas. Es el método más general pero es lento y no existe una forma simple para revertir los refinamientos. Con la triangulación de Delaunay, se adicionan puntos extra a la triangulación original para hacer el refinamiento.

Para tener más información de estos y otros métodos es recomendable revisar documentos especializados como [31].

Capítulo 3

Mapeo Ortogonal

La construcción de sistemas coordenados curvilíneos en donde la frontera de forma arbitraria es representada por una línea o superficie coordenada es un problema de importancia teórica y práctica. Muchos métodos han sido propuestos y la gran mayoría están revisados en el libro de Thompson *et al.* [29]. Aquí se discute un método para la generación de mallas ortogonales sobre dominios irregulares. Este tipo de mallas son muy utilizadas en problemas de simulación donde el dominio no es regular. Además, es posible usar métodos simples de discretización de las ecuaciones que gobiernan el fenómeno bajo estudio sin una modificación substancial. La particularidad del método presentado aquí, es que es no iterativo a diferencia de otros, lo que puede convertirlo en un método útil para problemas donde se requiere de mallas adaptivas. Los detalles del método se discuten en las secciones siguientes.

3.1 Ecuaciones del mapeo ortogonal

El paso más importante en el desarrollo de un mapeo entre un sistema Cartesiano y un sistema de coordenadas curvilíneo, es determinar las ecuaciones que las funciones de la transformación $x(\xi, \eta)$ y $y(\xi, \eta)$ deben cumplir. Primero se revisarán algunas relaciones básicas antes de escribir las ecuaciones que definen el mapeo.

3.1.1 Coordenadas generalizadas

Dado un dominio físico es posible encontrar una relación que sea uno a uno, entre las coordenadas del dominio generalizado o computacional y las coordenadas del dominio físico, véase figura 3.1, las cuales en dos dimensiones, se escriben como sigue:

$$\xi = \xi(x, y) \qquad \text{y} \qquad \eta = \eta(x, y), \tag{3.1}$$

y esto implica la existencia de las funciones $x(\xi, \eta)$ y $y(\xi, \eta)$. Las relaciones específicas se establecen una vez que la malla ha sido creada.

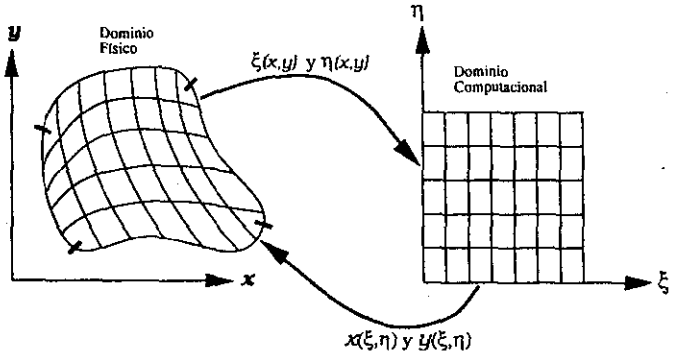

Figura 3.1: Mapeo entre el dominio físico y el dominio computacional.

Dadas las relaciones funcionales (3.1), las ecuaciones que gobiernan el fenómeno bajo estudio pueden transformarse en ecuaciones correspondientes que contienen derivadas parciales con respecto a ξ y η.

Como un ejemplo, las primeras derivadas de la velocidad, u y v, con respecto a x y y se transforman en:

$$\begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial \xi} & \frac{\partial u}{\partial \eta} \\ \frac{\partial v}{\partial \xi} & \frac{\partial v}{\partial \eta} \end{bmatrix} \mathbf{J}, \tag{3.2}$$

donde $\mathbf{J}$ representa a la matriz Jacobiana de la transformación que se define como sigue:

$$\mathbf{J} \equiv \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} \end{bmatrix}. \tag{3.3}$$

En principio, si existe una relación analítica para (3.1), los elementos de $\mathbf{J}$ pueden evaluarse directamente. En la práctica, no es usual tener una relación explícita para (3.1) y algunas veces es más conveniente trabajar con la matriz Jacobiana inversa $\mathbf{J}^{-1}$, dada por:

$$\mathbf{J}^{-1} \equiv \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix}. \tag{3.4}$$

Los elementos de $\mathbf{J}^{-1}$ pueden ser relacionados con los elementos de $\mathbf{J}$ notando que

$$\mathbf{J} = \frac{\text{Transpuesta del Cofactor}(\mathbf{J}^{-1})}{|\mathbf{J}^{-1}|}. \tag{3.5}$$

El determinante de la matriz Jacobiana inversa $|\mathbf{J}^{-1}|$ está dado por

$$|\mathbf{J}^{-1}| = x_\xi y_\eta - x_\eta y_\xi, \tag{3.6}$$

donde $x_\xi \equiv \frac{\partial x}{\partial \xi}$, etc.

Usando (3.5) y (3.6) los elementos de $\mathbf{J}$ en (3.3) pueden expresarse como:

$$\begin{aligned} \xi_x &= \tfrac{y_\eta}{|\mathbf{J}^{-1}|}, \quad \xi_y = \tfrac{-x_\eta}{|\mathbf{J}^{-1}|}, \\ \eta_x &= \tfrac{-y_\xi}{|\mathbf{J}^{-1}|}, \quad \eta_y = \tfrac{x_\xi}{|\mathbf{J}^{-1}|}, \end{aligned} \tag{3.7}$$

donde $|\mathbf{J}^{-1}|$ está definido por (3.6).

Definamos ahora el tensor métrico, el cual está relacionado con la matriz Jacobiana. En lo que sigue supondremos que el dominio físico está representado por coordenadas Cartesianas $x^i (\equiv x, y), i = 1, 2$, y el dominio computacional por las coordenadas generalizadas $\xi^i (\equiv \xi, \eta), i = 1, 2$.

Una distancia pequeña Δs entre dos puntos en el dominio físico puede escribirse en términos de los desplazamientos de las coordenadas Cartesianas como:

$$\Delta s^2 = \sum_{k=1}^{2} \Delta x^k \Delta x^k. \tag{3.8}$$

Los incrementos en las coordenadas del dominio físico Δx^k pueden relacionarse con cambios en las coordenadas generalizadas $\Delta \xi^i$ por

$$\Delta x^k = \frac{\partial x^k}{\partial \xi^i} \Delta \xi^i, \tag{3.9}$$

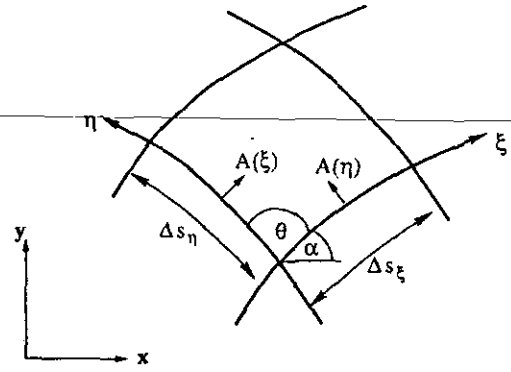

Figura 3.2: Características físicas de la malla computacional.

donde se toma la convención de Einstein (índices iguales se suman).

Consecuentemente la distancia Δs, en coordenadas generalizadas, se escribe como:

$$\Delta s^2 = \sum_{k=1}^{2} \left(\frac{\partial x^k}{\partial \xi^i} \Delta \xi^i \right) \left(\frac{\partial x^k}{\partial \xi^j} \Delta \xi^j \right) \tag{3.10}$$

$$= \sum_{i,j=1}^{2} g_{ij} \Delta \xi^i \Delta \xi^j, \tag{3.11}$$

donde se define el tensor métrico como:

$$g_{ij} = \sum_{k=1}^{2} \frac{\partial x^k}{\partial \xi^i} \frac{\partial x^k}{\partial \xi^j}. \tag{3.12}$$

El tensor métrico g_{ij} relaciona las contribuciones a la distancia Δs con pequeños cambios en las coordenadas generalizadas $\Delta \xi^i$. En dos dimensiones las distancias medidas a lo largo de las líneas ξ y η de una malla están dadas por $\Delta s_\xi = g_{11}^{1/2} \Delta \xi$ y $\Delta s_\eta = g_{22}^{1/2} \Delta \eta$, respectivamente, véase figura 3.2.

En dos dimensiones es conveniente escribir 3.12 en forma matricial

$$\mathbf{g} = \begin{bmatrix} (x_\xi^2 + y_\xi^2) & (x_\xi x_\eta + y_\xi y_\eta) \\ (x_\xi x_\eta + y_\xi y_\eta) & (x_\eta^2 + y_\eta^2) \end{bmatrix} = \frac{1}{|\mathbf{J}|^2} \begin{bmatrix} (\eta_x^2 + \eta_y^2) & -(\xi_x \eta_x + \xi_y \eta_y) \\ -(\xi_x \eta_x + \xi_y \eta_y) & (\xi_x^2 + \xi_y^2) \end{bmatrix}. \tag{3.13}$$

La razón de aspecto **AR** es definida por la razón de la magnitud de los incrementos en cada dirección:

$$\mathbf{AR} = \frac{\Delta s_\eta}{\Delta s_\xi} = \left(\frac{g_{22}}{g_{11}}\right)^{1/2}. \tag{3.14}$$

La distorsión local de la malla está determinada por el ángulo θ entre las líneas coordenadas ξ y η. De esta manera tenemos que

$$\cos(\theta) = \frac{g_{12}}{(g_{11}g_{22})^{1/2}}. \tag{3.15}$$

La forma de los parámetros anteriores para el caso tridimensional, son tratados en [16].

3.1.2 Restricción a coordenadas ortogonales

El uso de coordenadas generalizadas permite considerar geometrías arbitrarias. Sin embargo, se sabe que la exactitud de la solución es degradada por la distorsión de la malla. Para obtener alta precisión la malla debería ser ortogonal o "casi" ortogonal. En el caso de sistemas coordenados ortogonales algunos de los términos en las transformaciones desaparecen y las ecuaciones se simplifican. Si el sistema coordenado es también conforme, las ecuaciones gobernantes se simplifican aún más.

Para que una malla en dos dimensiones sea ortogonal, el ángulo θ debe ser igual a 90^o, véase figura 3.2. De la ecuación 3.15 se tiene que la condición de ortogonalidad se puede escribir como:

$$g_{12} = x_\xi x_\eta + y_\xi y_\eta = 0. \tag{3.16}$$

En tres dimensiones la condición de ortogonalidad sería $g_{ij} = 0, i \neq j$, que quiere decir que el tensor métrico contiene términos distintos de cero sólo en la diagonal, g_{ii} .

Si el sistema de coordenadas es ortogonal, es común definir los siguientes parámetros:

$$h_i = (g_{ii})^{1/2}, i = 1, 2 \tag{3.17}$$

Los términos h_i pueden interpretarse como factores de escala ya que un pequeño cambio en las coordenadas ξ^i sobre una malla ortogonal, produce un desplazamiento dado por

$$\Delta s_i = h_i \Delta \xi^i \tag{3.18}$$

En dos dimensiones la condición de ortogonalidad implica, de la ecuación 3.16, que

$$x_\eta = -y_\xi \mathbf{AR} \qquad \text{y} \qquad y_\eta = x_\xi \mathbf{AR}. \tag{3.19}$$

Si $\mathbf{AR} = 1$ entonces 3.19 se reduce a las condiciones de Cauchy-Riemann (véase por ejemplo Fletcher [10]) y se dice que la malla es conforme. Si $\mathbf{AR}$ es igual a una constante diferente de la unidad un simple escalamiento de ξ o η producirá un sistema de coordenadas conforme.

3.1.3 Ecuación covariante de Laplace

Una manera simple de determinar las ecuaciones que satisfacen las funciones de transformación $x(\xi, \eta)$ y $y(\xi, \eta)$ es adoptar el punto de vista covariante, bajo el cual se dice que la expresión de cualquier ley física o geométrica no depende del sistema particular de coordenadas que se utilice.

Para determinar las ecuaciones se observa que x, como una coordenada cartesiana en el dominio físico, es obviamente una función escalar lineal de la posición, y lo mismo sucede con y. De esta manera, ∇x y ∇y son campos vectoriales constantes y de aquí se sigue que

$$\nabla \cdot \nabla x = 0 \qquad \text{y} \qquad \nabla \cdot \nabla y = 0, \tag{3.20}$$

en cualquier lugar del espacio. Además, $\nabla \cdot \nabla = \nabla^2$ es el operador covariante de Laplace. Este operador puede escribirse en forma explícita para cualquier sistema de coordenadas incluyendo el que se desea construir, siempre y cuando se conozcan las componentes del tensor métrico g_{ij} definido en la ecuación 3.12.

La pregunta ahora es: ¿Como determinar el tensor métrico antes de construir el sistema de coordenadas?. La respuesta es que el desarrollo de una transformación apropiada de coordenadas debe iniciar especificando el tensor métrico el cual determina las propiedades del sistema de coordenadas resultante. Por ejemplo, si las componentes fuera de la diagonal de g_{ij} son todas iguales a cero, entonces el sistema coordenado será ortogonal. Si además, las componentes de la diagonal son iguales a uno, el sistema será Cartesiano, etc.

La restricción más obvia y usual en el caso general es poner todas las componentes fuera de la diagonal de g_{ij} iguales a cero, con lo que se asegura que el sistema de coordenadas resultante será ortogonal. El tensor métrico para coordenadas ortogonales en 2D se escribe como:

$$\mathbf{g} = \begin{bmatrix} h_1^2 & 0 \\ 0 & h_2^2 \end{bmatrix}, \tag{3.21}$$

donde las h_i's son definidas por 3.17. Usualmente se hace también una restricción sobre la razón entre los factores de escala h_1 y h_2. Dicha restricción es especificar la razón como una función de ξ y η, es decir, $f(\xi,\eta) \equiv h_2/h_1$. La razón h_2/h_1 tiene claramente un significado geométrico, especifica la razón de aspecto de los lados de un rectángulo pequeño en el plano (x,y) el cual es una imagen de un cuadrado pequeño en el plano (ξ,η). Es natural entonces, llamar a $f(\xi,\eta)$ la función de distorsión. Seleccionando cuidadosamente la función de distorsión, es posible controlar el espaciamiento de una malla en el dominio físico, la cual es una imagen de una malla uniforme en el dominio computacional (digamos un cuadrado unitario). Aunque la función de distorsión podría, en principio, ser ajustada automáticamente durante el curso de la solución numérica para reflejar los gradientes que evolucionan en la solución, tales algoritmos son muy caros y no serán tratados en esta tesis.

La condición $f(\xi,\eta) = 1$, que corresponde a un mapeo conforme, es obviamente una restricción mayor sobre la clase de posibles mapeos. Con esta restricción, la malla en el dominio computacional sería no homogénea, lo cual no es bueno para resolver ecuaciones diferenciales parciales por diferencias finitas. Una función ajustable de dos variables $f(\xi,\eta)$ evidentemente provee de mayor flexibilidad mientras que la condición de ortogonalidad se mantiene.

Usando las condiciones

$$g_{12} = 0,$$

$$\frac{h_\eta}{h_\xi} \equiv \frac{h_2}{h_1} \equiv \frac{(g_{22})^{1/2}}{(g_{11})^{1/2}} = f(\xi,\eta), \tag{3.22}$$

y la fórmula del operador covariante de Laplace en dos dimensiones en coordenadas ortogonales, la cual es (véase Fletcher [10]):

$$\nabla^2 = \frac{1}{h_\xi h_\xi}\left[\frac{\partial}{\partial\xi}\left(\frac{h_\eta}{h_\xi}\frac{\partial}{\partial\xi}\right) + \frac{\partial}{\partial\eta}\left(\frac{h_\xi}{h_\eta}\frac{\partial}{\partial\eta}\right)\right],$$

las ecuaciones 3.20 se transforman en

$$\frac{\partial}{\partial \xi}\left(f\frac{\partial x}{\partial \xi}\right)+\frac{\partial}{\partial \eta}\left(\frac{1}{f}\frac{\partial x}{\partial \eta}\right)=0 \qquad \text{y} \qquad \frac{\partial}{\partial \xi}\left(f\frac{\partial y}{\partial \xi}\right)+\frac{\partial}{\partial \eta}\left(\frac{1}{f}\frac{\partial y}{\partial \eta}\right)=0. \tag{3.23}$$

La solución de estas ecuaciones con condiciones de frontera apropiadas, darán la transformación de coordenadas Cartesianas (x, y) a un sistema de coordenadas curvilíneas ortogonales (ξ, η), siempre y cuando el mapeo exista para la forma particular de la frontera. Los factores de escala h_ξ y h_η, que son requeridos en las ecuaciones 3.23, pueden ser calculados fácilmente usando la definición 3.13:

$$h_\eta=\sqrt{(\partial x/\partial \xi)^2+(\partial y/\partial \xi)^2} \qquad \text{y} \qquad h_\xi=\sqrt{(\partial x/\partial \eta)^2+(\partial y/\partial \eta)^2}. \tag{3.24}$$

El problema del método de la ecuación covariante de Laplace es que, como $f=\frac{\sqrt{x_\xi^2+y_\xi^2}}{\sqrt{x_\eta^2+y_\eta^2}}$, las ecuaciones que resultan, 3.23 son no lineales y se conoce muy poco acerca de sus propiedades generales. Especialmente, mucho trabajo se ha hecho para responder las siguientes preguntas (véase por ejemplo Kang y Leal [14] y Ryskin y Leal [22]):

1. ¿Que condiciones de frontera son la adecuadas de tal manera que garanticen la existencia de una solución?

2. Si una solución existe, ¿Que tipo de restricciones es necesario imponer para obtener una solución única?

A pesar de la falta de entendimiento de las ecuaciones 3.23, el método de la ecuación covariante de Laplace ha sido aplicado en diferentes formas para generar mallas ortogonales. Ryskin y Leal [22] proponen un método de "restricción débil" para generar un mapeo ortogonal en un dominio fijo. En este método, las posiciones de los puntos de la malla se especifican en todos los lados de la frontera pero la función de distorsión, $f(\xi,\eta)$, es ajustada durante el curso de la solución para obtener ortogonalidad bajo ciertas reglas. Aunque se dan varios ejemplos en el trabajo de Ryskin y Leal, existen dos desventajas en este método. La primera es que no hay una demostración de existencia de la solución. La segunda es que el método está basado en

un ajuste iterativo de la función de distorsión. En vista de las fuertes no linealidades en las ecuaciones, la convergencia del esquema iterativo no está garantizada.

Pese a las desventajas del método de restricción débil, el trabajo de Ryskin y Leal ha motivado a desarrollar esquemas basados en la ecuación covariante de Laplace. Muchos de los esquemas desarrollados hasta ahora pueden ser clasificados en una de las siguientes categorías:

Clase 1: La función de distorsión $f(\xi, \eta)$ se especifica a priori, pero las condiciones de frontera consistentes con esta especificación son encontradas como parte de la solución.

Clase 2: La correspondencia de las fronteras es especificada en algunos o todos los lados de la frontera a priori, pero la función de distorsión $f(\xi, \eta)$ se encuentra como parte de la solución.

Para los métodos de la Clase 1, la existencia de la solución puede probarse sin dificultad si la función de distorsión se especifica en forma de un producto como $f(\xi, \eta) = \Pi(\xi)\Theta(\eta)$ (véase Kang y Leal [14]). Kang y Leal proponen un método de la Clase 2, el cual puede generar una malla ortogonal de manera no iterativa cuando se especifica la correspondencia de dos lados adyacentes de la frontera. Este esquema de la Clase 2 es la base de este trabajo y se describirá en detalle en la siguiente sección.

3.2 Descomposición del mapeo ortogonal

En el método de Kang y Leal [14], se utiliza el hecho de que un mapeo ortogonal entre un dominio físico irregular y un dominio computacional rectangular, denotado por $T_o : (\xi, \eta) \in \Omega_\xi \rightarrow (x, y) \in \Omega_x$, puede descomponerse en un mapeo conforme, denotado por $T_c : (u, v) \in \Omega_u \rightarrow (x, y) \in \Omega_x$, y un mapeo ortogonal auxiliar, denotado $T_{\tilde{o}} : (\xi, \eta) \in \Omega_\xi \rightarrow (u, v) \in \Omega_u$, como se muestra en la figura 3.3.

En la figura 3.3 Ω_x representa al dominio físico, Ω_ξ al dominio computacional y Ω_u al dominio auxiliar. Los símbolos $\partial\Omega_i$ y ∂D_i denotan los lados de la frontera del dominio físico y del dominio auxiliar respectivamente. El dominio auxiliar rectangular está escalado a $0 \leq u \leq 1$ y $0 \leq v \leq v^*$, donde v^* es determinada por las condiciones de Cauchy-Riemann.

La descomposición del mapeo ortogonal se escribe más formalmente como

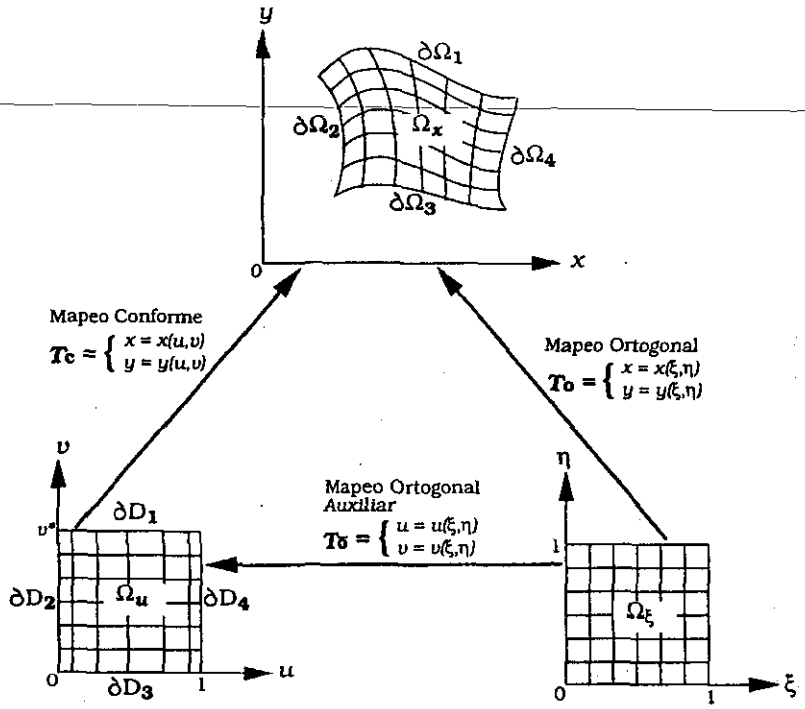

Figura 3.3: Descomposición de un mapeo ortogonal en un mapeo conforme y un mapeo ortogonal auxiliar.

$$T_o = T_c \cdot T_{\tilde{o}}. \tag{3.25}$$

La clave de está descomposición es que la función de distorsión para el mapeo global, f, es la misma que para el el mapeo auxiliar, $\tilde{f}$, es decir:

$$f(\xi,\eta) = \tilde{f}(\xi,\eta) \qquad \forall(\xi,\eta) \in \Omega_\xi, \tag{3.26}$$

donde $\tilde{f}(\xi,\eta)$ es la razón de los factores de escala $\tilde{h}_\eta$ y $\tilde{h}_\xi$ para el mapeo auxiliar definida como

$$\tilde{f}(\xi,\eta) = \frac{\tilde{h}_\eta}{\tilde{h}_\xi} = \frac{\sqrt{(\partial u/\partial\eta)^2 + (\partial v/\partial\eta)^2}}{\sqrt{(\partial u/\partial\xi)^2 + (\partial v/\partial\xi)^2}}. \tag{3.27}$$

La relación 3.26 se sigue del hecho de que la función de distorsión para el mapeo global es el producto de las funciones de distorsión de los dos mapeos secuenciales ($f = f_c\tilde{f}$) y la función de distorsión para el mapeo conforme es igual a uno ($f_c = 1$).

La importancia de la descomposición anterior es que, dada una malla ortogonal en el dominio auxiliar (Ω_u), la malla ortogonal correspondiente en el dominio físico (Ω_x) puede ser fácilmente generada usando la relación 3.26. De esta manera, el difícil problema de generar una malla ortogonal en el dominio físico puede ser reducido efectivamente al problema más simple de generar una malla ortogonal en el dominio auxiliar rectangular.

La transformación entre Ω_x y Ω_u es el mapeo conforme. En principio, se puede construir una malla en Ω_x haciendo una transformación conforme de cada punto en Ω_u para obtener el punto correspondiente en Ω_x. Sin embargo, en la práctica, no existe un esquema simple para este propósito. Pero si se visualiza el sistema de la malla, el cual se desea encontrar en Ω_x, como la transformada ortogonal global del dominio computacional, es decir, $x_i = x_i(\xi_i, \eta_j), y_j = y_j(\xi_i, \eta_j)$, en vez de considerar la transformada conforme del sistema del dominio auxiliar Ω_u, es decir, $x_i = x_i(u_i, v_j), y_j = y_j(u_i, v_j)$, entonces el problema se puede resolver de una manera fácil. Para encontrar el mapeo ortogonal global, se puede usar el método de la ecuación covariante de Laplace, descrito en la sección anterior. Las ecuaciones covariantes de Laplace, 3.23, pueden transformarse en dos ecuaciones lineales desacopladas con condiciones de frontera de Dirichlet y se resuelven fácilmente para producir una malla ortogonal en el dominio físico si se tiene lo siguiente:

1. la función de distorsión, $f(\xi, \eta)$, y
2. la correspondencia sobre todos los lados de la frontera de (Ω_x) que sea consistente con $f(\xi, \eta)$.

Se debe notar que la correspondencia de la frontera no puede ser especificada arbitrariamente, sino que esta debe especificarse de forma consistente con $f(\xi, \eta)$ para garantizar la ortogonalidad de la malla resultante. Cuando se tiene una malla ortogonal en el dominio (u, v), las condiciones de arriba se obtienen como sigue:

Suponiendo que se tiene una malla ortogonal en (u, v), es decir, se tiene una transformación ortogonal $u = u(\xi, \eta), v = v(\xi, \eta)$, entonces todas las derivadas en la ecuación 3.27 pueden evaluarse numéricamente. De esta forma, la función de distorsión para el mapeo global en cada punto de la malla

puede obtenerse usando las fórmulas 3.26 y 3.27. Sólo resta encontrar los puntos de la malla del dominio físico sobre la frontera que sean consistentes con f. Se sabe además que un mapeo conforme asegura la ortogonalidad de la malla en el dominio físico (salvo en puntos donde dos fronteras se intersectan), si la malla en el sistema auxiliar es ortogonal. Entonces, los puntos de la frontera consistentes con f, se encuentran a partir de la transformación conforme de los puntos de la frontera del dominio auxiliar al dominio físico.

Los puntos de la malla sobre la frontera se obtienen fácilmente usando la técnica de integral de frontera. Dado que la información que se desea encontrar de la parte del mapeo conforme es solamente la correspondencia de la frontera entre Ω_x y Ω_u, es conveniente considerar el mapeo inverso $T_c^{-1} : u = u(x, y), v = v(x, y)$, en virtud de que la geometría del dominio físico es arbitrariamente compleja. El mapeo inverso T_c^{-1} es también conforme y está gobernado por dos ecuaciones de Laplace conjugadas, véase [14]:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \qquad \nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} = 0. \tag{3.28}$$

Las condiciones de frontera, necesarias para resolver las ecuaciones anteriores son, véase figura 3.4:

$$\begin{pmatrix} \frac{\partial u}{\partial \mathbf{n}} \\ u \\ \frac{\partial u}{\partial \mathbf{n}} \\ u \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \text{sobre} \begin{pmatrix} \partial\Omega_1 \\ \partial\Omega_2 \\ \partial\Omega_3 \\ \partial\Omega_4 \end{pmatrix}$$

y

$$\begin{pmatrix} v \\ \frac{\partial v}{\partial \mathbf{n}} \\ v \\ \frac{\partial v}{\partial \mathbf{n}} \end{pmatrix} = \begin{pmatrix} v* \\ 0 \\ 0 \\ 0 \end{pmatrix} \text{sobre} \begin{pmatrix} \partial\Omega_1 \\ \partial\Omega_2 \\ \partial\Omega_3 \\ \partial\Omega_4 \end{pmatrix}$$

En la figura 3.4, $\mathbf{n}$ y s denotan el vector normal unitario hacia afuera y la longitud de arco a lo largo de la frontera iniciando desde un punto de referencia. En la figura, las condiciones de frontera $u = 0$, $u = 1$, $v = 0$ y

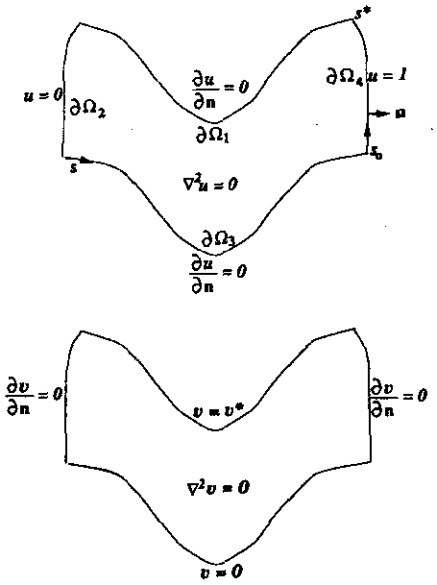

Figura 3.4: Ecuaciones de Laplace conjugadas con condiciones de frontera para el mapeo conforme.

$v = v^*$ se deben al escalamiento seleccionado para el dominio auxiliar, y las condiciones $\partial u/\partial \mathbf{n} = 0$ y $\partial v/\partial \mathbf{n} = 0$ son requeridas para obtener ortogonalidad de las coordenadas conformes en la frontera. Como se mencionó antes, v^* se determina a partir de la condición de Cauchy-Riemann

$$\frac{\partial u}{\partial \mathbf{n}} = \frac{\partial v}{\partial s}, \tag{3.29}$$

a lo largo de la frontera $\partial\Omega_4$. Como se ve en la figura 3.4, $u(s)$ sobre $\partial\Omega_1$ y $\partial\Omega_3$, y $(\partial u/\partial \mathbf{n})(s)$ sobre $\partial\Omega_2$ y $\partial\Omega_4$ se obtienen de la solución del problema u de las ecuaciones 3.28. Dicha solución es encontrada usando el método de integral de frontera, véase sección 3.3. Así, v^* puede determinarse integrando 3.29,

$$v^*(s) = \int_{s_o}^{s} \frac{\partial u}{\partial \mathbf{n}}(t)dt. \tag{3.30}$$

Por lo tanto, el problema para v puede resolverse similarmente para obtener $v(s)$ sobre $\partial\Omega_2$ y $\partial\Omega_4$.

Una vez resueltos los problemas tanto para u como para v, se tendrá una descripción completa de u y v como funciones de la longitud de arco. Se observa que una de las funciones $u(s)$ y $v(s)$ es una función monótona por lo que existe su inversa, mientras que la otra es constante sobre cada lado de la frontera. Podemos usar esta información para obtener la correspondencia de la frontera entre el dominio físico y el dominio auxiliar. Considérese primero un punto dado sobre la frontera del dominio físico, véase la figura 3.5. Entonces, se determina la longitud de arco a ese punto, s, para calcular los valores $u(s)$ y $v(s)$. Los valores obtenidos (u, v) definen el punto correspondiente sobre la frontera del dominio auxiliar. Inversamente, si se da un punto sobre la frontera del dominio auxiliar (u, v), entonces, dado que una de las dos funciones es monótona en el lado correspondiente de la frontera del dominio físico, la longitud de arco s que corresponde al punto del dominio físico se obtiene invirtiendo la función que es monótona. Dado que la coordenada (x, y), del punto de la frontera puede representarse como $x = x(s)$ y $y = y(s)$, los valores de x y y del punto correspondiente de la frontera están determinados.

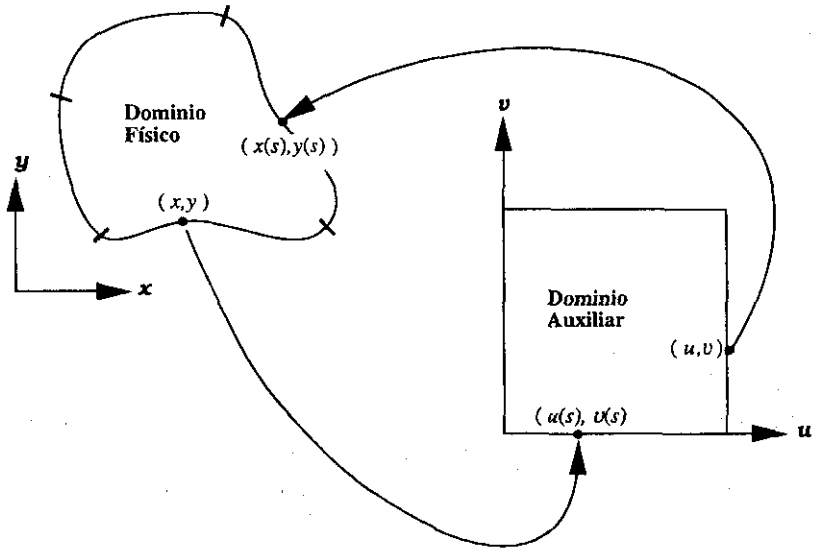

Figura 3.5: Determinación de los puntos de la frontera consistentes con f.

De esta manera, se determina la correspondencia sobre todos los lados de la frontera consistente con la función de distorsión $f(\xi, \eta)$.

3.3 Formulación de integral de frontera

Como se mencionó en la sección anterior, existe un parámetro del mapeo conforme que debe determinarse como parte de la solución denotado como v^*. Por este motivo, se inicia resolviendo primero el problema para u, con el objeto de determinar el valor de v^* usando la ecuación 3.30. Debido a que los problemas para u y v son idénticos, excepto por las condiciones de frontera, aquí se verá la discusión para el problema u solamente, el tratamiento para el problema v es idéntico. El objetivo del método de integral de frontera (BI) (véase [13]), es usar la forma apropiada de la fórmula de Green y las soluciones fundamentales de las ecuaciones de la transformación, para convertir el problema dado por las ecuaciones 3.28, en una fórmula integral, en la que se especifican los datos que se desconocen sobre cada segmento de la frontera, en términos de los datos que han sido especificados de antemano. Por ejemplo, en la figura 3.6 los valores de u son conocidos sobre los segmentos 2 y 4, mientras que $\partial u/\partial \mathbf{n}$ se conoce sobre los segmentos 1 y 3. A partir de estos valores conocidos es posible determinar los valores de ambas variables sobre los otros segmentos.

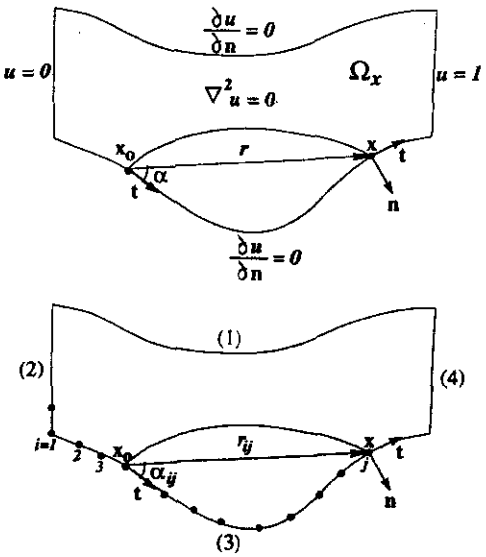

Figura 3.6:

Considérese un dominio arbitrario simplemente conexo en 2D. La fórmula

de Green para el problema especificado en las ecuaciones 3.28 en un dominio bidimensional está dada por

$$\beta u(\mathbf{x}) = \int_{\partial\Omega_x} \mathbf{G}(\mathbf{x}-\mathbf{x_o}) \left(\frac{\partial u}{\partial \mathbf{n}}\right) ds - \int_{\partial\Omega_x} u \left(\frac{\partial \mathbf{G}}{\partial \mathbf{n}}\right) ds, \tag{3.31}$$

donde $\mathbf{x} = (x, y) \in \Omega_x$, $\beta = 2\pi$ si $\mathbf{x}$ está dentro de $\bar{\Omega}_x$ pero no en la frontera, $\beta = \pi$ si $\mathbf{x}$ está sobre una parte suave de la frontera y $\beta = \theta$ si $\mathbf{x}$ está sobre una esquina de la frontera, θ es el ángulo de la esquina. La función de Green $\mathbf{G}$, para el problema en 2D descrito por las ecuaciones 3.28 es

$$\mathbf{G}(\mathbf{x}-\mathbf{x_o}) = -\log|\mathbf{x}-\mathbf{x_o}| = -\log \mathbf{r}. \tag{3.32}$$

Sustituyendo 3.32 en 3.31 se obtiene una fórmula integral para u en términos de los valores de u y $\partial u/\partial \mathbf{n}$ en la frontera

$$\beta u = -\int_{\partial\Omega_x} \left(\frac{\partial u}{\partial \mathbf{n}}\right) \log \mathbf{r}\, ds + \int_{\partial\Omega_x} u \left(\frac{\partial \log \mathbf{r}}{\partial \mathbf{n}}\right) ds. \tag{3.33}$$

Si se define $w = \partial u/\partial \mathbf{n}$ y se representa a u y w usando un número finito de puntos nodales, como se ve en la figura 3.6, entonces 3.33 puede representarse por la siguiente ecuación vectorial:

$$\mathbf{Au} = \mathbf{Bw}, \tag{3.34}$$

donde

$$\begin{aligned} \mathbf{A} &= (A^1, A^2, A^3, A^4), \\ \mathbf{B} &= (B^1, B^2, B^3, B^4), \\ \mathbf{u} &= (u_1, \tilde{u}_2, u_3, \tilde{u}_4)^T \end{aligned}$$

y

$$\mathbf{w} = (\tilde{w}_1, w_2, \tilde{w}_3, w_4)^T,$$

donde los subíndices y los superíndices denotan los lados de la frontera (i.e., los lados 1-4 del dominio), y la tilde es usada para indicar las variables conocidas sobre un segmento particular de la frontera (i.e., u se especifica sobre los lados 2 y 4, mientras que w es conocida sobre los lados 1 y 3). Rearreglando la ecuación 3.34 se obtiene

$$\mathbf{My} = \mathbf{N}\tilde{\mathbf{y}}. \tag{3.35}$$

donde

$$\mathbf{M} = (A^1, -B^2, A^3, -B^3),$$
$$\mathbf{N} = (B^1, -A^2, B^3, -A^4),$$
$$\mathbf{y} = (u_1, w_2, u_3, w_4)^T$$

y

$$\tilde{\mathbf{y}} = (\tilde{w}_1, \tilde{u}_2, \tilde{w}_3, \tilde{u}_4)^T.$$

En 3.35, el lado derecho se conoce por lo que la solución está dada por

$$\mathbf{y} = \mathbf{M}^{-1}(\mathbf{N}\tilde{\mathbf{y}}). \tag{3.36}$$

La ecuación 3.36 provee de una fórmula explícita para calcular los valores de u y w sobre las fronteras donde éstas no han sido especificadas en la formulación del problema de mapeo conforme.

Los coeficientes de las matrices $\mathbf{M}$ y $\mathbf{N}$ pueden ser calculados fácilmente, dado que éstos están dados en términos de las coordenadas (x, y) de los puntos de la frontera del dominio físico. Por lo tanto los problemas de las ecuaciones 3.28, se reducen a resolver un par de sistemas de ecuaciones de la forma mostrada en 3.35.

3.4 Esquema de generación de mallas ortogonales

La idea de la descomposición del mapeo ortogonal se puede aplicar para generar una malla ortogonal en 2D cuando se especifican las coordenadas de los puntos sobre dos lados adyacentes de la frontera, como se observa en la figura 3.7.

En la figura, las coordenadas de los puntos de la frontera se especifican sobre los lados $\partial\Omega_2$ y $\partial\Omega_3$. El método de Kang y Leal [14], para generar mallas ortogonales usado en este trabajo, se puede resumir como sigue:

1. Se resuelve el problema de mapeo conforme mediante la técnica de integral de frontera. Como se describió en la sección anterior, si u se conoce en los segmentos de frontera 2 y 4, entonces la solución de los problemas descritos por las ecuaciones 3.28 determinará los valores que toma u en los segmentos 1 y 3. Algo similar sucede con v.

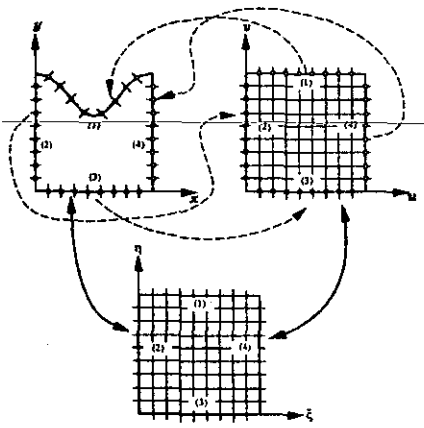

Figura 3.7:

2. Se especifican las coordenadas de los puntos sobre dos segmentos adyacentes de la frontera y, dada la información obtenida en el paso 1, se determinan (u, v) (mediante una interpolación) en el dominio auxiliar para los segmentos correspondientes. En el caso mostrado en la figura 3.7, la especificación de las coordenadas se hace en los segmentos 2 y 3.

3. Se construye una malla ortogonal en el dominio auxiliar (u, v) bajo la suposición de que los puntos correspondientes sobre los lados opuestos tienen los mismos valores u o v. Bajo esta suposición, la malla será rectangular y no uniforme. En este caso, la transformación ortogonal entre (u, v) y (ξ, η) puede representarse por $u = u(\xi) = u_b(\xi)$ y $v = v(\eta) = v_b(\eta)$, donde $u_b(\xi)$ y $v_b(\eta)$ denotan las correspondencias sobre ∂D_3 y ∂D_2 que se obtuvieron en el paso 1.

4. Usando las fórmulas 3.26 y 3.27, se calcula la función de distorsión, $f(\xi, \eta)$, para todos los puntos de la malla.

5. Los puntos de la frontera de la malla sobre $\partial\Omega_1$ y $\partial\Omega_4$ del dominio físico correspondientes a los lados ∂D_1 y ∂D_4 del dominio auxiliar se determinan usando la información obtenida en el paso 1.

6. Finalmente, se resuelven las ecuaciones 3.23 para encontrar los puntos interiores de la malla. Se utiliza la función $f(\xi, \eta)$ calculada en 3 y

las coordenadas de los puntos de todos los lados de la frontera como condiciones de frontera.

Capítulo 4

Metodología de Programación

El desarrollo de sistemas de software es una industria relativamente joven y no ha alcanzado el nivel de madurez típicamente encontrado en otras ramas de la industria. Consecuentemente, los productos desarrollados usando tecnología de software comúnmente sufren de la falta de estabilidad requerida para su desarrollo y explotación.

La orientación a objetos es una técnica para el modelado de sistemas en general. Esta técnica ofrece un número de conceptos para desarrollar de manera conveniente diferentes sistemas disminuyendo su complejidad y facilitando su modificación y reuso.

En lo que sigue se describirá el análisis y diseño del sistema de generación de mallas ortogonales, el cual se modeló haciendo uso de metodología orientada a objetos.

4.1 La entropía del sistema

La segunda ley de la termodinámica, en principio, dice que el "desorden" de un sistema cerrado no puede reducirse, esté sólo puede incrementarse o posiblemente permanecer sin cambio. Una medida de este desorden se conoce como **entropía**. Esta ley puede, en cierto sentido, aplicarse a los sistemas de cómputo; el desorden de un programa, o entropía, siempre se incrementa. Algunos autores le llaman entropía de software al desorden antes mencionado.

Dentro del desarrollo de un programa o sistema de cómputo existen algunas teorías con ciertas leyes similares a la segunda ley de la termodinámica (ver Lehman [17]), de las cuales se mencionarán dos de ellas:

1. Un programa que es usado será modificado.

2. Cuando un programa se modifica, su complejidad se incrementa.

Un sistema tendrá inicialmente una cierta entropía de software. La experiencia muestra que es razonable asumir que el incremento en la entropía del software es proporcional al estado de la entropía cuando se inicia la modificación del sistema. Esto significa que es más fácil modificar un sistema ordenado que uno desordenado. Lo anterior puede expresarse matemáticamente como sigue:

$$\Delta \mathbf{E} \propto \mathbf{E}$$

o, del cálculo diferencial tenemos que

$$\frac{d\mathbf{E}}{\partial t} = k\mathbf{E}$$

que es una ecuación diferencial simple cuya solución se muestra en la figura 4.1. En la figura 4.1 se observa que el tiempo de vida de un sistema depende de que tan bien estructurado esté al momento de su creación. Cuando una cierta entropía del software se alcanza, no es justificable, económicamente, continuar usando dicho sistema, pues una modificación posterior será demasiado cara. Una posibilidad es aplicar un proceso conocido como re-ingeniería para reducir la entropía del software de tal manera que sea posible seguir manteniéndolo a un costo razonable.

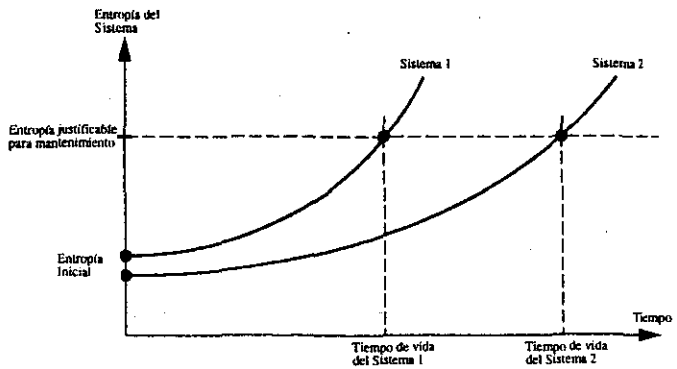

Figura 4.1: Entropía de software.

Cuando un sistema es diseñado con la intención de que éste sea manejable por un periodo largo de tiempo, se trata de obtener la entropía más baja posible desde el principio. Este es uno de los objetivos de los métodos para el desarrollo de sistemas. Por este medio se espera incrementar el tiempo de vida del sistema. Se sabe que tarde o temprano un límite será alcanzado más allá del cual será muy caro mantener el sistema.

Para diseñar un buen sistema de software, diferentes métodos han sido propuestos para describir ya sea un proyecto para desarrollar una primera versión de un producto o para dar una vista global del ciclo de vida de un sistema completo. La definición de un "buen" sistema de software varía dependiendo de las aplicaciones. Por ejemplo, en algunos casos es el rendimiento lo que importa, mientras que en otros es más importante la interfaz gráfica con el usuario. También depende de la estructura del sistema, por ejemplo, si esta será distribuida o centralizada. Lo que es común es que todos los sistemas necesitan ser modificados en algún momento.

Tradicionalmente, el trabajo de análisis y diseño es estructurado y descrito usando diferentes tipos de modelos de cascada, ver figura 4.2. Estos modelos describen el proceso de desarrollo del sistema donde la idea inicial es que cada fase deberá completarse antes de iniciar la siguiente. Sin embargo, esto último fue rápidamente abolido y en la actualidad es permitido iniciar una fase antes de completar totalmente la anterior. El modelo de cascada ha tenido un tremendo impacto sobre los métodos de ingeniería de software, aunque esté no se sigue de manera rígida cuando es aplicado.

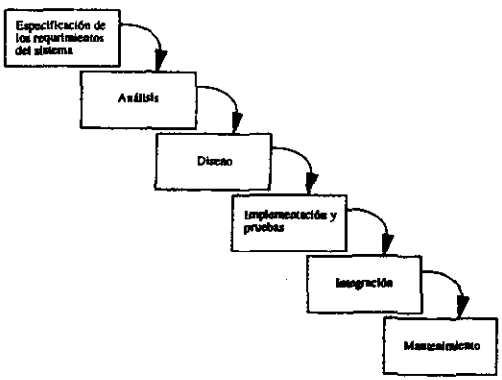

Figura 4.2: Modelo de cascada de desarrollo de un sistema.

El problema con el modelo de cascada es en la etapa de mantenimien-

to, donde la mayoría de las veces es necesario realizar modificaciones, se producen nuevas especificaciones de requerimientos, análisis, diseño, etc. Diferentes modelos fueron desarrollados para describir estos nuevos hechos, uno de los más populares es el modelo de espiral (ver Boehm [2]), mostrado en la figura 4.3. El modelo de espiral puede describir la forma en que un producto se desarrolla para formar nuevas versiones y como una versión puede modificarse incrementalmente desde el prototipo hasta el producto final.

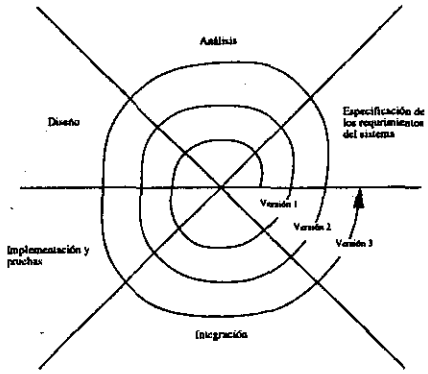

Figura 4.3: Modelo espiral para el desarrollo de un sistema.

En esta tesis se utilizó este último modelo para diseñar el sistema de generación de mallas ortogonales. El modelo de espiral es la forma natural de desarrollar sistemas, pues cada vez que se concluye una etapa del sistema, se tiene que probar con las anteriores y la mayoría de las veces es necesario regresar a una etapa anterior para que el modelo cumpla con los requisitos establecidos. La versión del programa que se tiene hasta ahora, cumple con las especificaciones establecidas al inicio, pero posiblemente tendrá que ser modificado o extendido para aplicarse en las diferentes áreas donde sea útil.

4.2 Tecnología orientada a objetos

La Tecnología Orientada a Objetos (TOO) ha crecido rápidamente en los últimos años, de tal manera que en estos días es una de las más importantes en el desarrollo de software. Esta técnica tiene un enorme potencial para incrementar significativamente la productividad del programador y facilitar el mantenimiento del código. Se ha observado que cuando se utiliza la técnica de programación estructurada o procedural, en proyectos de software

de dimensiones industriales (miles de líneas de código), el código suele ser inestable bajo pequeñas perturbaciones, es decir, cuando se trata de reparar o modificar un problema simple se pueden generar cinco o más errores en algún otro lugar del código. Cuando el tamaño del código es mayor que un límite (alrededor de 10^5 líneas de código), estás técnicas de programación fallan: la complejidad del diseño y la implementación hace al proyecto muy difícil y caro, y eventualmente, el costo del mantenimiento domina el costo del desarrollo. Las técnicas orientadas a objetos permiten construir aplicaciones más estables y de mayor tamaño (entre millones y decenas de millones de líneas de código).

Simula, un lenguaje pionero en la orientación a objetos, fue concebido especialmente para el desarrollo de simulaciones. De aquí que no es sorprendente que la programación orientada a objetos se aplique más fácilmente en simulaciones y visualizaciones de fenómenos complejos, que las técnicas estructuradas (lenguajes como Fortran por ejemplo).

El principio tanto de las técnicas orientadas a objetos como de las estructuradas es "divide y vencerás". La diferencia fundamental entre ellas es la elección de los bloques con los que se construye un software. En las técnicas estructuradas las unidades fundamentales son los procedimientos (subrutinas o subprogramas). En el modelo de objetos las unidades básicas se conocen como **objetos** los cuales contienen tanto datos (es decir, variables de estado que describen el objeto en un cierto instante) como métodos (es decir reglas dinámicas, reglas que explican como el objeto interactúa con el mundo exterior). Por lo tanto es común encontrar la siguiente definición de objeto: *Un objeto es todo aquello tangible que tiene un estado y un comportamiento, y además reacciona de cierta manera a mensajes que le envían otros objetos. Se dice que un objeto encapsula sus datos y su comportamiento.*

Los objetos interactúan (intercambian mensajes) para producir un comportamiento colectivo. Es este comportamiento colectivo el que puede resolver un problema de la vida real.

La tecnología orientada a objetos ve al mundo como compuesto de objetos con propiedades muy bien definidas. Las interacciones son realizadas mediante mensajes que los objetos intercambian uno con otro.

4.2.1 Conceptos básicos de TOO

El modelo de objetos provee fundamentos teóricos a partir de los cuales se construye un software orientado a objetos, véase por ejemplo Booch [3] y

Jacobson [12]. Este modelo se basa en los principios de *abstracción*, *encapsulación*, *modularidad*, *jerarquía*, *polimorfismo*, entre otros.

Una *Abstracción* consiste en extraer las características relevantes del sistema a ser modelado, dando una generalización adecuada sin tomar en cuenta los detalles irrelevantes. Una *Abstracción* determina las características de un objeto que lo distingue de otros tipos de objetos. Esta es la vista exterior del objeto. El primer paso en la modelación de un sistema es elegir adecuadamente el conjunto de abstracciones para nuestro problema. Dentro de la TOO, la *Encapsulación* es también conocida como *ocultamiento de la información*. Los clientes del objeto no pueden ver la estructura interna de éste. Los objetos se construyen, se instancian, a partir de clases. Una **clase** es un molde que describe cómo estos objetos son estructurados internamente. Objetos de la misma clase tienen características y comportamientos comunes. En general, el estado de un objeto sólo puede ser modificado por métodos o funciones del objeto mismo, por lo que el cliente de dicho objeto deberá enviar un **mensaje** al objeto para que modifique su estado. El envío de mensajes entre objetos es un concepto diferente de la invocación de una función con un número de argumentos. En el envío de mensajes entre objetos, se sabe quién es el emisor y quién el receptor, mientras que en la llamada a una función, no se está seguro de quién invoca a quién. Por ejemplo, la función `magnitud(vector)` regresará el valor de la magnitud de un cierto vector, pero no se sabe de que vector se trata ni quién recibe el valor. Por otro lado, `vector1.magnitud()` es el envío de un mensaje al `vector1` para que diga el valor de su magnitud.

La *Jerarquía* es concepto importante dentro de la TOO, y se puede definir como un ordenamiento de abstracciones, o en el contexto de la TOO, es un ordenamiento de clases. Se puede desarrollar toda una jerarquía de clases, donde existe una clase raíz o superclase y distintas clases hijas o subclases. Un concepto importante aquí es la **herencia** que se refiere a la capacidad de derivar una nueva clase de una superclase existente. La superclase sirve como patrón para la clase derivada y puede ser modificada de varias formas. La herencia es muy importante en la TOO ya que permite la reutilización de las definiciones de las clases, para cambios simples, sin requerir mayores modificaciones sobre el código. Entonces en una jerarquía los elementos básicos son las clases derivadas. Las superclases representan tareas más generalizadas, mientras que las clases derivadas corresponden a objetos con tareas específicas.

La capacidad de responder de maneras distintas a mensajes distintos de

un objeto o una función es lo que se conoce como *Polimorfismo*. Esta característica está inmersa dentro de la TOO. El polimorfismo es usado para trabajar con distintos tipos de datos que requieren de acciones o cálculos similares.

Finalmente, *Modularidad* significa que un sistema puede ser descompuesto en un conjunto de módulos acoplados para resolver un problema. El objetivo es que la estructura del sistema sea simple y fácil de modificar. En general, uno de los problemas más difíciles en el diseño de sistemas complejos es encontrar los módulos adecuados para modelar el sistema eficientemente. Aunque un módulo contiene datos y funciones, esto es, tiene un estado y un comportamiento, difiere de un objeto en el sentido de que un módulo sólo existe una vez en la ejecución de un programa, mientras que puede haber varios objetos del mismo tipo coexistiendo en tiempo de ejecución.

Los conceptos antes descritos, son de mucha utilidad al momento de desarrollar un software. Dichos conceptos han tenido mucho éxito en el desarrollo de aplicaciones de graficación y bases de datos, entre otras. Para el caso de aplicaciones numéricas, estos conceptos son bastante útiles. Por ejemplo, la encapsulación de la información asegura que un método numérico sólo afectará los datos requeridos. Gracias a la encapsulación, dicho método numérico no podrá modificar datos que no sean permitidos. Por otro lado, se debe tener cuidado con estos conceptos, pues una mala decisión al momento de usarlos puede producir códigos altamente ineficientes, lo cual no es deseable en este tipo de aplicaciones.

En la siguiente sección se describe como estos conceptos son aplicados en la construcción del sistema para generar mallas ortogonales.

4.3 Análisis y diseño

La fase del análisis se concentra en entender el problema y modelar en el dominio del problema. El objetivo es proporcionar el modelo del sistema enfocándose en su comportamiento y no en su estructura. En la parte del diseño se decide todo acerca de la programación del sistema, como el lenguaje, sobre que plataformas será ejecutado, que tipo de algoritmos son los adecuados, etc.

4.3.1 Modelo de objetos

Se puede describir en forma general el sistema de generación ortogonal de mallas con el siguiente texto:

> Generación de Mallas Usando Mapeo Ortogonal.
> Dadas las coordenadas de los puntos de la frontera de un dominio irregular en dos dimensiones, se desea construir una malla ortogonal sobre dicho dominio por medio de un mapeo ortogonal entre el dominio irregular, también conocido como dominio físico, y un dominio regular o dominio computacional. Para facilitar el proceso de construcción de la malla, el mapeo ortogonal inicial se descompone en dos etapas: la primera consiste en hacer un mapeo conforme entre un dominio regular auxiliar y el dominio físico, en tanto que en la segunda se realiza un mapeo ortogonal, muy simple, entre el dominio computacional y el dominio auxiliar. El método para construir la malla se describe en detalle en la sección 3.4.

Del texto anterior y de la descripción del método de generación de ortogonal de mallas dada en el capítulo 3, es posible encontrar las abstracciones necesarias para modelar este sistema. La decisión de llevar cada abstracción a una clase se basa en la eficiencia y en el reuso futuro que pueda hacerse de éstas en otro tipo de problemas.

Las clases construidas en este trabajo se definen a continuación:

Punto : La clase punto contiene las coordenadas (x, y) de los puntos de la frontera. Para los puntos interiores no se usa esta clase dado que es necesario acceder de manera intensiva a las coordenadas y el uso de funciones miembro para obtener los valores de x y y produce ineficiencia en el código.

Vector : Durante el curso del cálculo de los puntos interiores de la malla, son necesarios distintos arreglos tipo entero y de doble precisión. La clase Vector es usada para contener las entradas de estos arreglos. Además, contiene algunas funciones importantes para realizar algunos cálculos numéricos.

Matriz : En esta clase se almacenan los diferentes sistemas lineales que surgen en el método de integral de frontera y en la discretización de las

ecuaciones del mapeo ortogonal. Dado que los sistemas que se tienen que manejar son tanto densos como dispersos, es necesario tener dos clases que manejen eficientemente la memoria para ambos casos. Las funciones miembro de estas clases han sido optimizadas para realizar cálculos numéricos de manera eficiente.

Frontera : Esta clase contiene los puntos de la frontera y se usa en conjunción con la clase Laplace para resolver el problema de mapeo conforme por el método de integral de frontera.

edp-numérica: Esta es una clase puramente abstracta y es usada para definir ecuaciones diferenciales parciales que se van a resolver de manera numérica. Las ecuaciones diferenciales parciales que esta clase abarca son de la forma:

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu + G = 0.$$

Elíptica : Esta clase define ecuaciones diferenciales elípticas y se deriva de la anterior. Las ecuaciones elípticas se distinguen por que se cumple que: $B^2 - 4AC < 0$. También es una clase puramente abstracta que sirve para definir toda tipo de ecuaciones elípticas numéricas.

ElípticaVF : En este problema se tienen que resolver numéricamente algunas ecuaciones parciales elípticas como las mostradas en 3.23. Esta clase define ese tipo de ecuaciones y el calificativo VF significa que la ecuación se discretiza por medio del método de Volumen Finito.

LaplaceIF : El problema de mapeo conforme se define a través de un par de ecuaciones de Laplace y éstas se resuelven por medio del método de Integral de Fontera, de ahí el calificativo IF. Esta clase se deriva de la clase **Elíptica** usando los parámetros: $A = C = 1$ y los demás iguales a cero.

Dominio : Clase abstracta que define todo tipo de dominios, ya sean regulares o irregulares.

DominioRegular : Esta clase define dominios exclusivamente regulares que se pueden generar fácilmente. Se deriva de la clase anterior.

FunciónDistorsión : Clase para contener y calcular la función de distorsión que es un parámetro importante en la generación de la malla.

Malla : Clase abstracta que define mallas estructuradas en general.

MallaOrtogonal : Clase que define mallas ortogonales.

Interpolación : Esta clase define dos tipos de interpolación: lineal y splines cúbicos. Estas interpolaciones son usadas en distintos puntos del cálculo.

Integral : Para calcular v^* se necesita hacer una integral. Esta clase define una integración. Los métodos usados para realizar la integración son el de Simpson y el del trapecio.

Splines : Esta clase define splines cúbicos naturales y es usada por la clase **Interpolación**.

Las asociaciones entre las clases se muestran en las figuras siguientes:

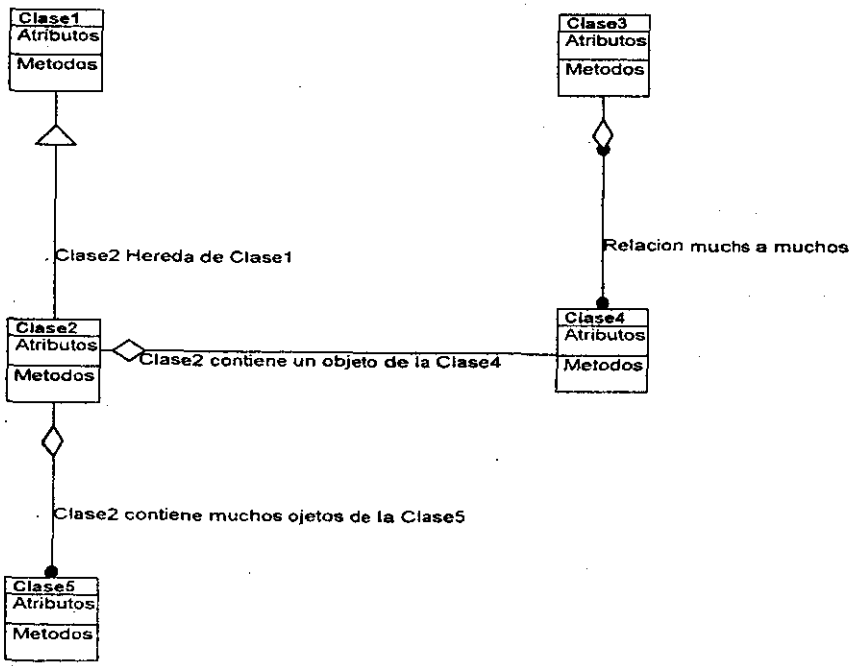

Figura 4.4: Tipos de relaciones.

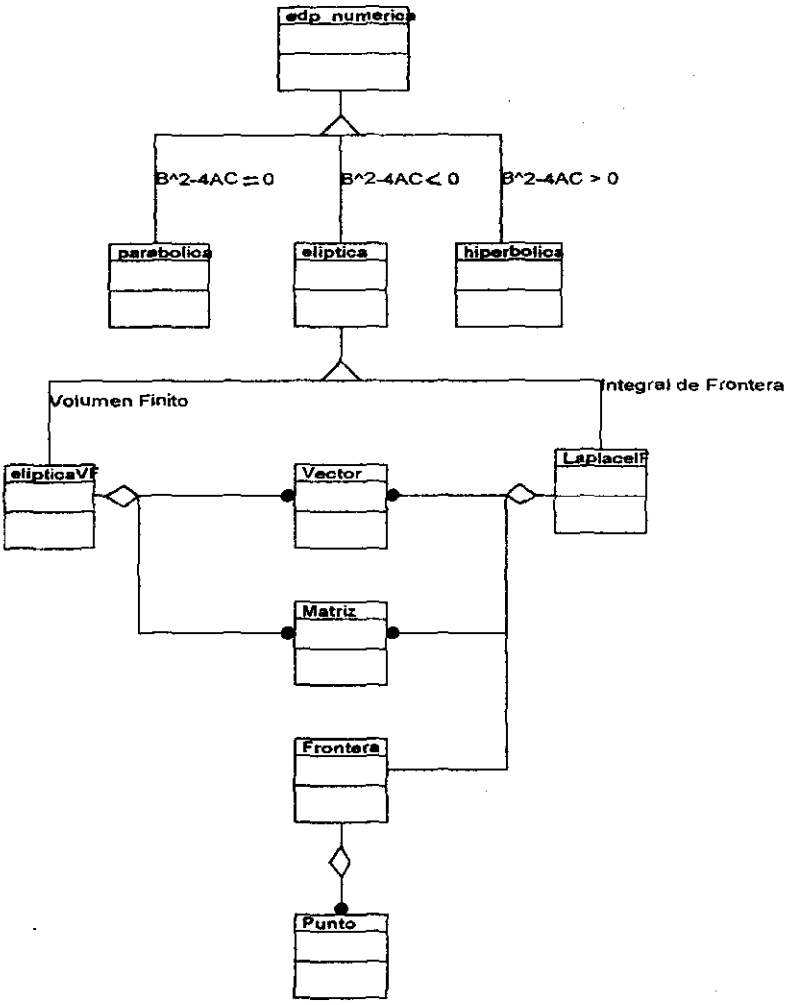

Figura 4.5: Jerarquía de clases para el manejo de ecuaciones diferenciales parciales.

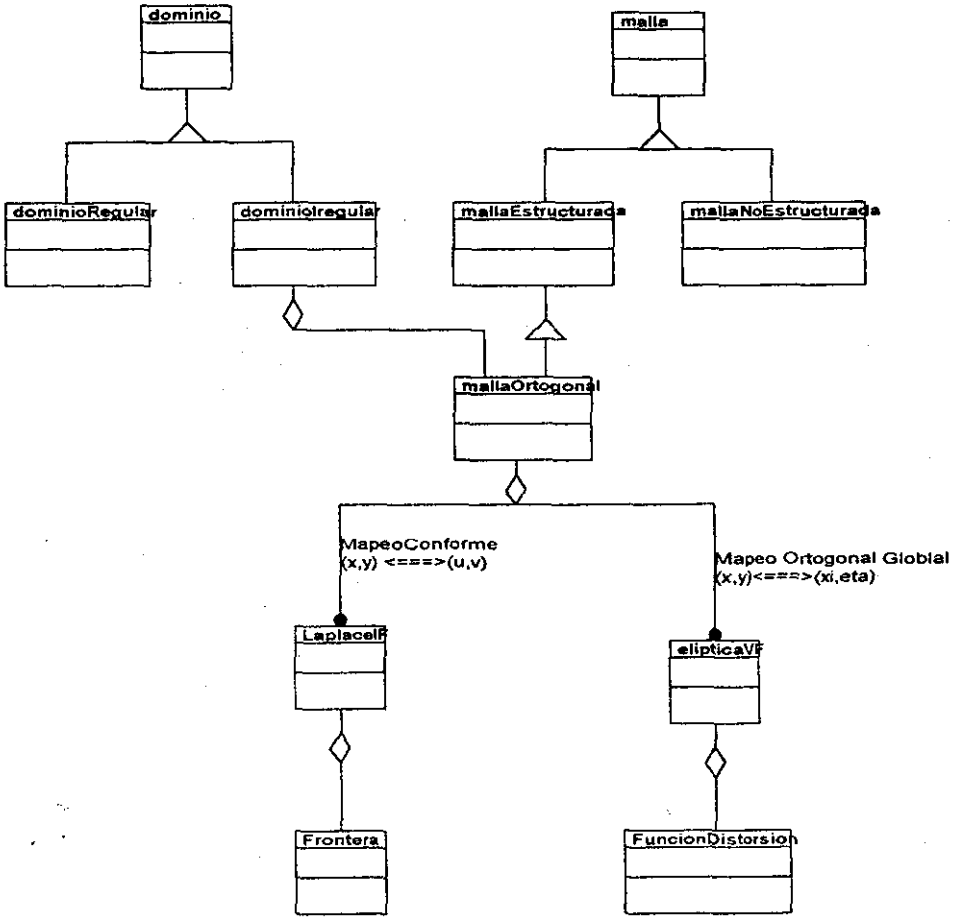

Figura 4.6: Jerarquía de clases para el manejo de mallas sobre dominios irregulares.

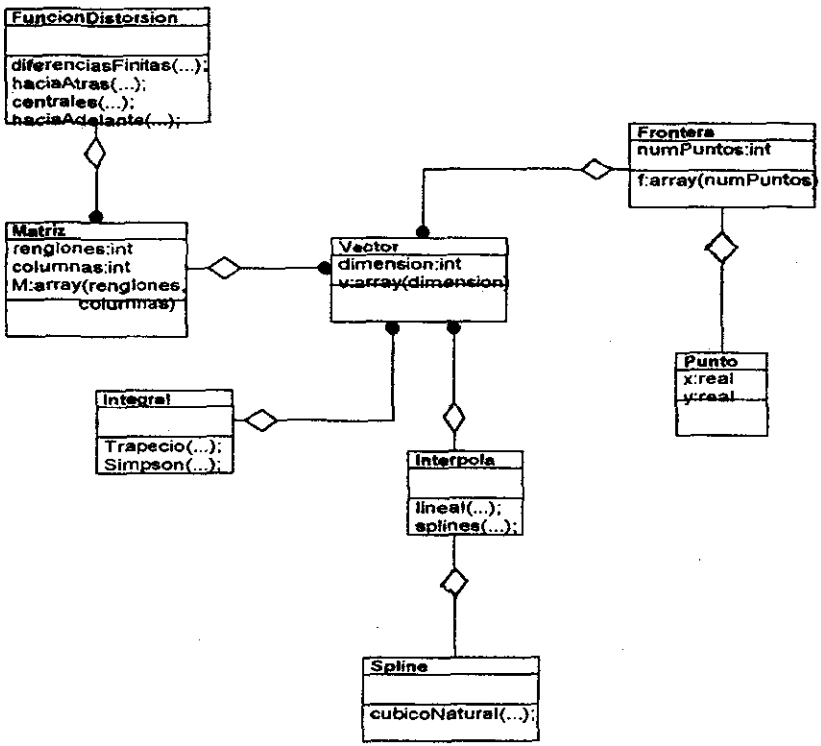

Figura 4.7: Clases auxiliares para implementar los métodos numéricos.

ESTA TESIS NO SALE
DE LA BIBLIOTECA

4.3.2 Diseño del sistema

El lenguaje de programación

El lenguaje que se utilizó para codificar los algoritmos numéricos y la interacción entre los objetos es C++. Este lenguaje fue desarrollado a principios de los 80's por B. Stroustrup [25]. El lenguaje no es completamente orientado a objetos en el sentido de que es posible programar con este lenguaje sin usar la POO. Sin embargo, C++ ofrece las herramientas necesarias para utilizar todos los conceptos de la POO.

Aunque existen muchos lenguajes OO, la decisión de utilizar C++ se debió principalmente por que este lenguaje fue construido pensando en la eficiencia. La eficiencia es uno de los puntos importantes que se requieren en este trabajo. Además existen compiladores de este lenguaje sobre la mayoría de las arquitecturas existentes.

Algoritmos numéricos

En el esquema numérico que se adoptó para la generación de mallas, básicamente se necesitan de dos tipos de herramientas numéricas:

- La técnica de integral de frontera para resolver las ecuaciones de Laplace conjugadas mostradas en la ecuación 3.28.
- Un sistema para resolver las ecuaciones covariantes de Laplace mostradas en 3.23 con condiciones de frontera tipo Dirichlet.

Las aproximaciones de las derivadas que aparecen en distintos puntos del método de generación de mallas se hicieron usando diferencias finitas, ver [10].

El método de integral de frontera para resolver la ecuación de Laplace en un dominio irregular, se implementó usando una frontera discreta y permitiendo a la clase **LaplaceIF** acceder a los datos de la frontera directamente para hace más eficientes los cálculos(usando el modificador `friend` de C++).

Para la discretización de las ecuaciones 3.23, se utilizó el método de Volumen Finito o volumen de Control, véase [19]. Los sistemas lineales que genera este tipo de discretización son dispersos, por lo que se optó por usar algoritmos iterativos para la solución estos sistemas.

El uso de matrices y vectores en este tipo de métodos numéricos es de vital importancia. Existen algunas clases ya implementadas que pueden ser

de utilidad en este trabajo. En la siguiente sección se describe que bibliotecas se reusaron para la construcción del sistema.

4.3.3 Reuso de bibliotecas numéricas

En años recientes, la POO ha sido explotada para construir bibliotecas numéricas eficientes y fácilmente reusables. Dado que uno de los principios de la POO es poder reusar el código, en este trabajo se hizo uso de algunas bibliotecas numéricas desarrolladas por diferentes personas para análisis numérico. Estas bibliotecas consisten principalmente en el manejo de matrices y vectores para resolver sistemas lineales de ecuaciones. Enseguida se describen estas bibliotecas.

MV++

Esta biblioteca es usada principalmente para el manejo de matrices y vectores. Como es sabido, en la mayoría de problemas donde una discretización es realizada, son necesarias las matrices y los vectores para almacenar los sistemas lineales provenientes de la discretización. El manejo eficiente de este tipo de objetos es de vital importancia dado que de ello depende la eficiencia del código.

MV++ ofrece un conjunto de clases fácil de usar y combinar con otras clases para resolver sistemas lineales y para hacer operaciones de álgebra lineal en general.

Los objetos que se usan de esta biblioteca son:

MV_Vector_TYPE : Esta clase sirve para el manejo de vectores de tipo entero doble precisión y para números complejos. TYPE puede ser sustituido por cualquiera de los tipos antes mencionados. Esta clase contiene los métodos suficientes y necesarios para nuestro problema.

MV_ColMat_TYPE : Esta clase hace el manejo de la memoria para almacenar, modificar y hacer operaciones con matrices de tipo entero, doble precisión y complejo. Esta clase es compatible con la forma de almacenamiento tipo Fortran 77y 90, en donde las entradas de la matriz se guardan por columnas. De esta manera esta clase puede ser usada sin mayor problema con subrutinas de Fortran para álgebra lineal.

Sparse++

En este trabajo se discretizan ecuaciones elípticas usando el método de volumen finito lo que produce sistemas lineales dispersos, por lo que es necesario hacer un almacenamiento inteligente de estos sistemas para evitar desperdiciar memoria con los múltiples ceros que aparecen en ellos.

Las clases de Sparse++ usan métodos de almacenamiento para matrices dispersas. Se tienen principalmente tres tipos de almacenamientos: por coordenadas, por columnas y por renglones. Además, estas clases contienen operaciones importantes, como multiplicación de matrices y multiplicación matriz por vector. Usando estas clases se optimiza el uso de la memoria y se asegura eficiencia del código.

IML++

Existen muchos métodos de solución de sistemas lineales y muchos de ellos han sido codificados en lenguajes de programación, principalmente en Fortran. IML++ es una de las primeras bibliotecas que usa programación orientada a objetos y genérica para codificar métodos de solución de sistemas lineales. Las funciones de IML++ están basadas en el uso de plantillas (`templates`) que es una característica del lenguaje C++ y que permite hacer programación genérica, ver [24]. Las funciones de esta biblioteca se pueden combinar eficientemente con objetos para manejo de matrices y vectores que cumplan con cierta funcionalidad. En particular, IML++, puede ser usada con Sparse++ y MV++.

En este trabajo se combina la eficiencia de almacenamiento de Sparse++ con los algoritmos optimizados de IML++, los cuales están basados en BLAS (Basic Linear Algebra Subprograms) que son algoritmos optimizados para operaciones de álgebra lineal, ver [5].

La ventaja de usar las bibliotecas mencionadas arriba, es que son de dominio público, lo cual significa que se puede obtener el código fuente para ser modificado. Además estas bibliotecas son portables y pueden ser ejecutadas sobre cualquier plataforma con sistema operativo tipo UNIX.

Para mayores detalles de los métodos y bibliotecas véase [21], [6], [7] y [1].

Capítulo 5

Resultados y Aplicaciones

En este capítulo se reportan los resultados obtenidos de la construcción del sistema de generación de mallas ortogonales. Primeramente se realizan algunas pruebas de calibración del software. Luego se muestran algunas ejemplos de mallas ortogonales en dominios irregulares y se mide la desviación de la ortogonalidad de éstas. Finalmente se presenta una aplicación de uso de este tipo de mallas.

Todos los ejemplos mostrados en este capítulo fueron realizados en una estación de trabajo Unix SUN Sparc 4 a 100 Mhz usando doble precisión.

5.1 Prueba y calibración del software

El sistema de generación de mallas tiene dos partes esenciales. La primera es la solución de las ecuaciones de Laplace conjugadas, mostradas en 3.23. La segunda resuelve la ecuación covariante de Laplace para obtener las coordenadas de los puntos interiores de la malla.

En la primera parte se leen de un archivo las coordenadas de los puntos de la frontera del dominio irregular. Las coordenadas de estos puntos deben ser especificados como se muestra en la figura 5.1.

Como se observa en la figura, la frontera del dominio debe ser cerrada y el interior del dominio no debe contener agujeros. Además, debido a que el dominio irregular será transformado en un rectángulo unitario, la frontera debe dividirse en 4 subfronteras, y el número de puntos sobre fronteras opuestas debe ser igual.

Un objeto de la clase **frontera** se encarga de leer los puntos del archivo

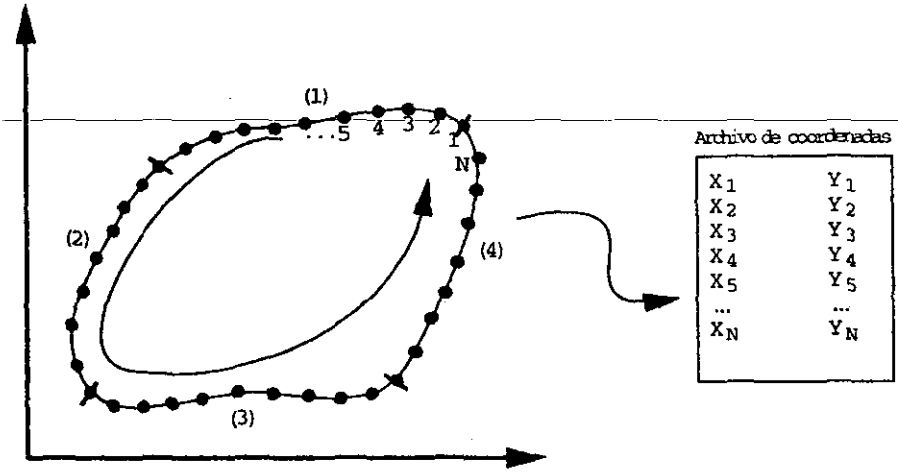

Figura 5.1: Especificación de las coordenadas de los puntos .

y de acomodarlos para que sean usados por la clase **LaplaceIF**. Esta última contiene un método para resolver las ecuaciones conjugadas de Laplace usando la técnica de integral de frontera. Para un dominio con 96 puntos sobre la frontera, el método numérico construye un sistema lineal de ecuaciones denso. El tamaño del sistema lineal para este número de puntos es de 92×92.

En la segunda parte, objetos de la clase **elípticaVF** utilizan la información obtenida por **LaplaceIF** para obtener las condiciones de frontera tipo Dirichlet, y de esta manera resolver las ecuaciones covariantes de Laplace que se muestran en 3.23. Para el caso de un dominio con 96 puntos sobre la frontera, se obtiene un sistema lineal de ecuaciones de tamaño 529×529 el cual es disperso. En este caso se utiliza el algoritmo de gradiente conjugado para resolver este tipo de sistemas. El tiempo aproximado para resolver un sistema de este tamaño, con una tolerancia de 10^{-8} y un límite de 200 iteraciones para alcanzar convergencia, es de 1.28 segs.

Como se mencionó en el capítulo anterior, para almacenar y resolver los sistemas lineales se reusaron tres bibliotecas numéricas. Para guardar sistemas densos de ecuaciones y arreglos de números flotantes se utilizó **MV++**. El acceso a los elementos de matrices y arreglos en esta biblioteca se hace usando referencias por lo que se tiene gran eficiencia con el uso de estas clases. Por otro lado, se combina **Sparse++** con **IML++** para resolver sistemas lineales de ecuaciones dispersos. En la tabla 5.1 se muestran los tiempos tomados para resolver sistemas dispersos de diferentes tamaños.

N	Tiempo (segs.)
40	0.07
81	0.11
361	0.77
529	1.28
784	2.19
1849	7.42
2304	8.61

Tabla 5.1:

La primera prueba con el sistema completo es con un cuadrado unitario. El número de puntos en cada lado del cuadrado es de 11. El tiempo que le tomó al sistema resolver este ejemplo completo fue de menos de 1 segundo. La malla obtenida se muestra en la figura 5.2.

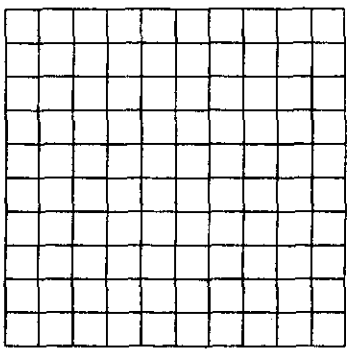

Figura 5.2: Cuadrado unitario de (11 × 11).

Como se observa, la malla es ortogonal a simple vista y parece razonable. Más adelante se utiliza una medida cuantitativa.

5.2 Ejemplos de mallas ortogonales

En esta sección se muestran algunos ejemplos de aplicación del sistema desarrollado en esta tesis. Se generaron mallas sobre diferentes regiones irregulares. En principio las mallas deben ser ortogonales, pero debido a errores numéricos y debilidades del método de generación de mallas, los resultados no son exactos. Para cada malla se calculan dos índices que indican el grado de ortogonalidad de éstas. Los índices son: MDO que es la Máxima Desviación de la Ortogonalidad y DPO que representa la Desviación Promedio de la Ortogonalidad. Estos índices se definen como sigue:

$$\mathbf{MDO} = \max_{ij}|\frac{\pi}{2} - \theta_{ij}|, \tag{5.1}$$

$$\mathbf{DPO} = \frac{1}{N_\xi N_\eta}\Sigma|\frac{\pi}{2} - \theta_{ij}|, \tag{5.2}$$

donde N_ξ es el número de puntos de la malla en la dirección ξ, N_η número de puntos en la dirección η, y θ_{ij} es calculado a partir de la ecuación 3.15.

En el caso del cuadrado de la sección anterior se obtuvo $\mathbf{MDO} = 0.77$ y $\mathbf{PDO} = 0.22$, ambas medidas son en grados.

El primer ejemplo que se considera en esta sección se muestra en la figura 5.3. Esta geometría ha sido usada por diferentes autores para probar sus métodos de generación de mallas y en este trabajo se usará para comparar con otros resultados, véase [22] y [15].

La parte superior de la este dominio está dado por $y = 0.8 + 0.2\cos(2\pi x)$, $0 \leq x \leq 1$. En las frontera laterales la distribución de los puntos es uniforme, mientras que en la parte inferior se usa la siguiente fórmula para distribuir los puntos: $y = \xi - 0.08\sin(2\pi\xi)$. El número de puntos sobre cada frontera es igual a 25. Debido a que sólo se puede especificar la correspondencia de los puntos de la frontera en 2 lados adyacentes de la frontera, en este caso la especificación se hizo en lado izquierdo y en la parte inferior de la frontera.

Dos elementos importantes en el método de generación de mallas son el dominio auxiliar (u, v) y la función de deformación f. En la figura 5.3 se muestran también las gráficas de estos elementos.

Para estudiar la dependencia de la función de distorsión con la especificación de la correspondencia de las coordenadas en las fronteras, se utilizó la misma geometría de la figura 5.4 y se especificaron los puntos de la frontera en diferentes lados.

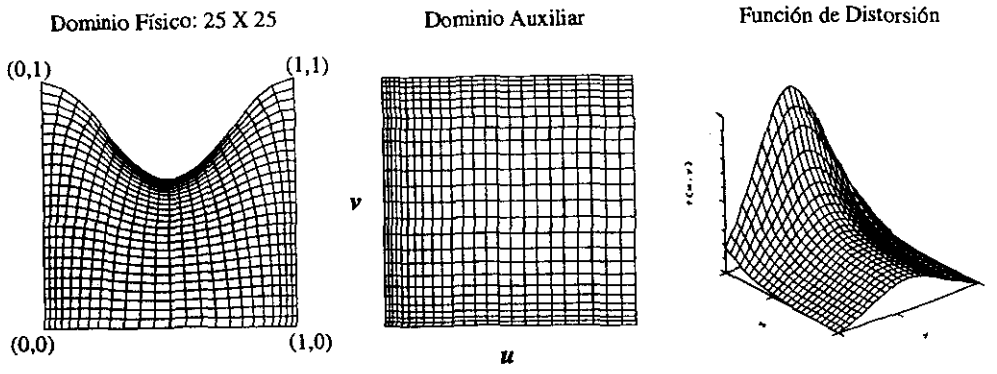

Figura 5.3: Malla ortogonal generada en un dominio simétrico. La especificación de las coordenadas se hace en los lados izquierdo e inferior de la frontera.

Como se observa, el cambio en la especificación de los puntos de la frontera, afecta al dominio (u, v), por lo que la función de distorsión se modifica y la malla interior también es distinta. En cuanto a la ortogonalidad, el cálculo de el MDO y el PDO se muestra en la tabla 5.2 para las dos mallas.

	figura 5.3	figura 5.4
MDO (grados)	12.72	18.75
PDO (grados)	5.33	6.06

Tabla 5.2:

El cambio en la ortogonalidad en las dos casos mostrados se nota más para el caso de la MDO, mientras que el PDO cambia en menos de un grado. Oh y Kang [15] reportan un resultado de 8.4 grados para la MDO y 0.4 para el PDO. La diferencia se debe a que ellos usan un método mejorado de generación de mallas que hace posible mejorar la ortogonalidad especificando las coordenadas en tres lados de la frontera.

Se construyeron otros ejemplos de mallas sobre dominios irregulares para verificar la consistencia del sistema y los resultados se muestran en la figura 5.5.

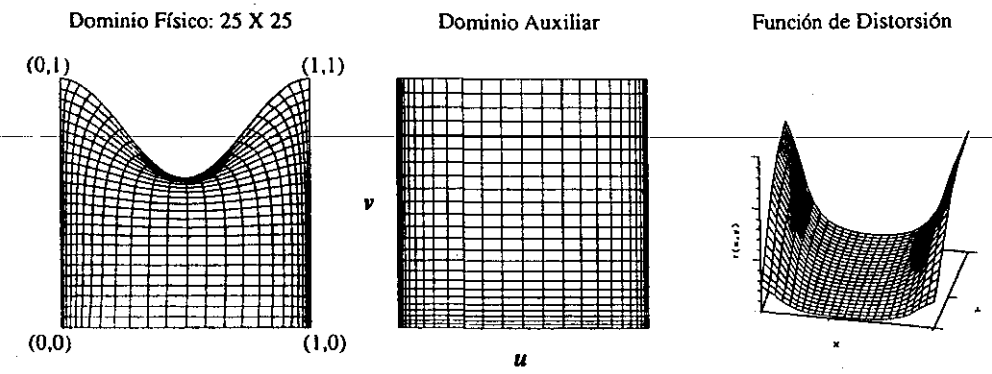

Figura 5.4: Malla ortogonal generada en el mismo dominio mostrado en la figura 5.3. La especificación de las coordenadas se hizo en la parte superior y en lado izquierdo de la frontera.

El cálculo de la ortogonalidad de estos últimos ejemplos se muestra en la tabla 5.3.

	a	b	c	d
MDO	12.87	16.40	9.77	15.37
PDO	6.66	7.85	4.38	6.32

Tabla 5.3:

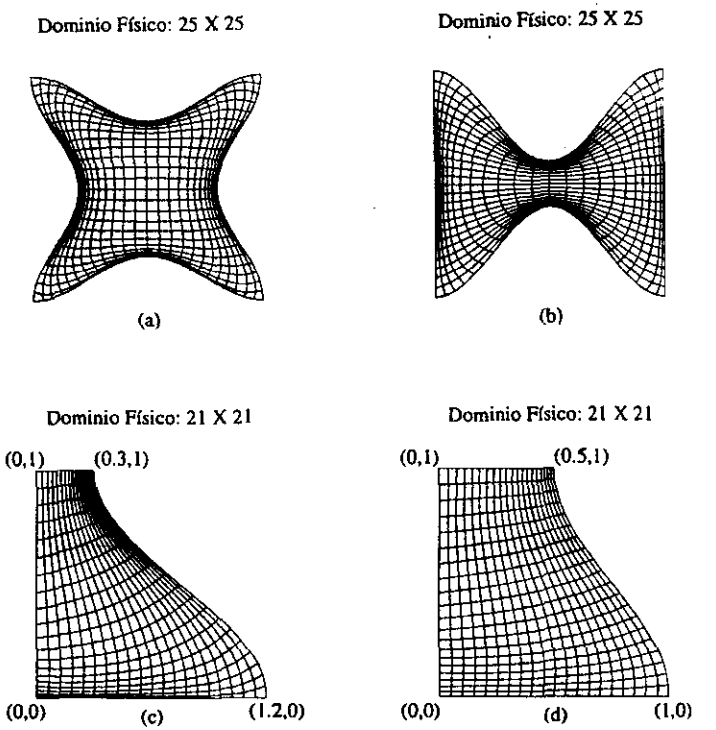

Figura 5.5: (a) Los cuatro lados de esta figura se construyeron usando una función senoidal y la distribución es uniforme en todas las fronteras; (b) La parte superior e inferior son funciones senoidales desfasadas; (c) y (d) El lado derecho de ambas figuras se construyó usando la función $x = 0.75 + H\cos(\pi y)$. En el caso de (c) H = 0.25, mientras que para (d) H = 0.45. La distribución de los puntos en el lado izquierdo se hizo con la función $y = 1 - [0.2(1-\eta) + 0.8\sin((\pi/2)(1-\eta))]$.

Contorno de Inglaterra

El sistema trabaja bien con fronteras no muy complicadas, como se mostró en los ejemplos anteriores. En el caso de geometrías muy irregulares, como es el caso del contorno de Inglaterra, véase figura 5.6, el sistema no funciona.

En la figura 5.6, se observa que la malla en el dominio (u, v) es inhomogénea. En ciertos lugares el cálculo de las derivadas para generar la función de distorsión, produce números muy grandes lo cual se ve reflejado en la función de distorsión. Estos valores dan como consecuencia una matriz

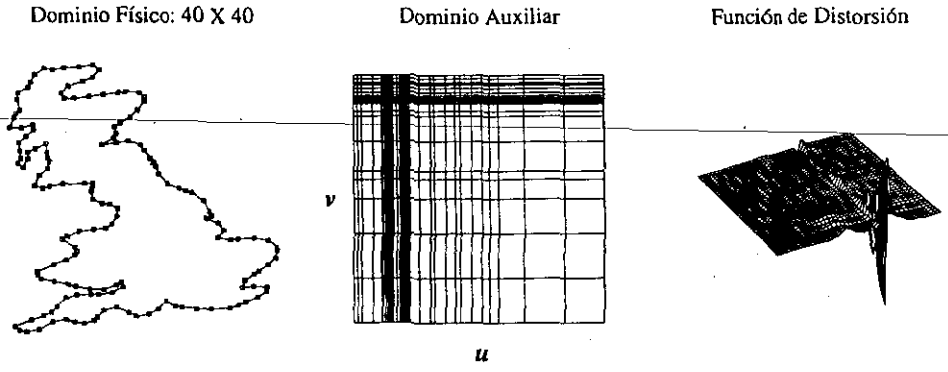

Figura 5.6: Contorno de Inglaterra.

mal condicionada. El algoritmo de solución del sistema, que es gradiente conjugado, no converge después de 200 iteraciones.

5.3 Aplicaciones

En esta sección se resuelve la ecuación de Laplace en un dominio semicircular donde la solución analítica se conoce. El objetivo es comparar el resultado numérico con el analítico y resaltar la conveniencia de usar mallas ortogonales.

En dos dimensiones y en coordenadas Cartesianas, la ecuación de Laplace se escribe como sigue:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0. \tag{5.3}$$

Esta ecuación será resuelta en el dominio mostrado en la figura 5.7 y las condiciones de frontera son las siguientes:

$$\begin{aligned} T &= 0 && \text{en} \quad \text{WX}, \\ T &= \tfrac{\sin(\theta)}{r_{XY}} && \text{en} \quad \text{XY}, \\ T &= \tfrac{1}{r_{YZ}} && \text{en} \quad \text{YZ}, \\ T &= \tfrac{\sin(\theta)}{r_{WZ}} && \text{en} \quad \text{ZW}, \end{aligned}$$

donde r_{XY} y r_{WZ} son constantes y r_{YZ} varía del punto Y al punto Z, véase figura 5.7.

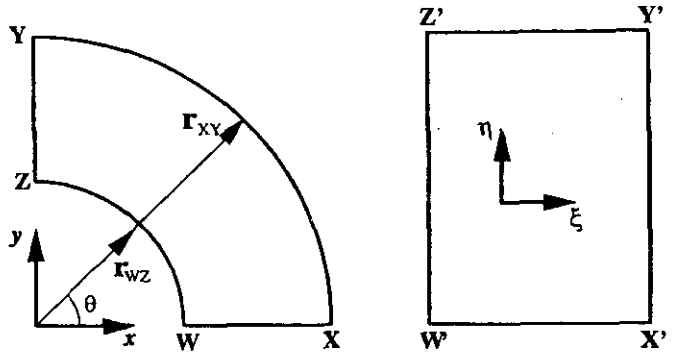

Figura 5.7: Dominio físico y computacional para resolver la ecuación de Laplace

Este problema tiene solución exacta:

$$T = \frac{\sin\theta}{r}, \tag{5.4}$$

esta fórmula será usada para comparar con la solución numérica.

En coordenadas generalizadas (ξ, η), la ecuación de Laplace se escribe como sigue

$$\frac{\partial}{\partial\xi}\left(\frac{g_{22}}{g^{1/2}}\frac{\partial T}{\partial\xi} - \frac{g_{12}}{g^{1/2}}\frac{\partial T}{\partial\eta}\right) + \frac{\partial}{\partial\eta}\left(\frac{g_{21}}{g^{1/2}}\frac{\partial T}{\partial\xi} - \frac{g_{11}}{g^{1/2}}\frac{\partial T}{\partial\eta}\right) = 0. \tag{5.5}$$

donde g_{ij} son están definidos por la ecuación 3.13 y $g^{1/2}$ es evaluado a partir de la ecuación 3.13.

Ahora, si el sistema de coordenadas es ortogonal, $g_{12} = g_{21} = 0$, lo que simplifica la ecuación anterior a

$$\frac{\partial}{\partial\xi}\left(\frac{h_2}{h_1}\frac{\partial T}{\partial\xi}\right) + \frac{\partial}{\partial\eta}\left(\frac{h_1}{h_2}\frac{\partial T}{\partial\eta}\right) = 0, \tag{5.6}$$

donde los factores de escala están dados por la ecuación 3.17.

Por otro lado, la ecuación 3.22 define a la función de distorsión a través de los factores h_1 y h_2, por lo que haciendo uso de este hecho, la última ecuación se puede escribir como

$$\frac{\partial}{\partial\xi}\left(f\frac{\partial T}{\partial\xi}\right) + \frac{\partial}{\partial\eta}\left(\frac{1}{f}\frac{\partial T}{\partial\eta}\right) = 0, \tag{5.7}$$

Esta última ecuación es idéntica a las mostradas en 3.23, por lo que es posible reusar las clases construidas para resolver dichas ecuaciones. Es aquí donde se nota una de las ventajas de la programación orientada a objetos. En este caso sólo se hará uso de las clases **FunciónDistorsión** y **ElípticaVF**.

La malla generada con el sistema desarrollado en este trabajo se muestra en la figura 5.8 y los parámetros que miden la ortogonalidad de la malla son:

MDO	PDO
5.06368	0.942297

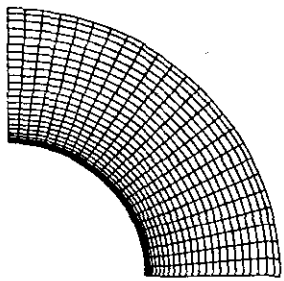

Figura 5.8: Malla ortogonal del semicírculo.

En la figura 5.9 se muestra una comparación entre el resultado numérico y la solución analítica.

Como puede observarse, la diferencia entre la solución analítica y la solución numérica es pequeña. El error máximo es de aproximadamente del 8%. En las figuras 5.10 y 5.11 se muestra una comparación en dos cortes del dominio. Los resultados mostrados en la figura 5.10 son tomados de en la recta definida por $\theta = 45^o$ o $x = y$. Por otro lados, la figura 5.11 muestra una comparación en el semicírculo definido por $r = 0.67$. Las tablas 5.4 y 5.5 muestran los datos numéricos.

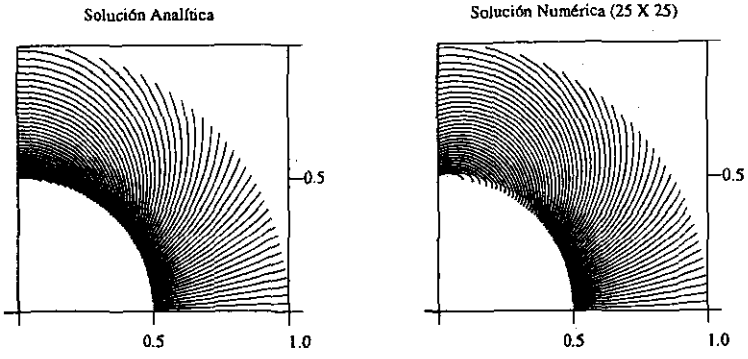

Figura 5.9: Comparación de la solución analítica con la solución numérica en una malla ortogonal. Las gráficas muestran contornos de $T = \text{cte}$.

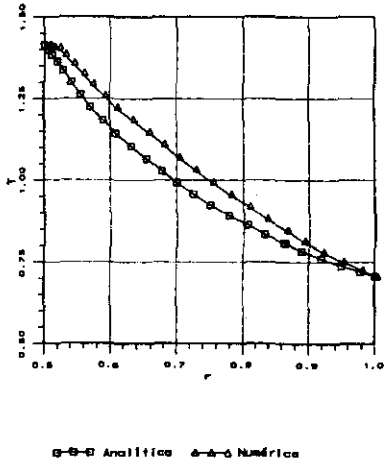

Figura 5.10: Comparación de la solución analítica con el resultado numérico en $\theta = 45°$.

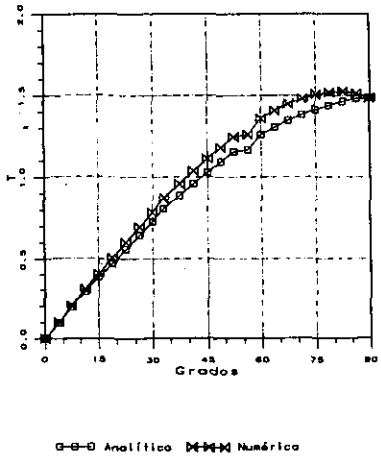

Figura 5.11: Comparación de la solución analítica con el resultado numérico en $r = 0.67$.

r	Sol. Analítica	Sol. Numérica	Error (%)
0.500000	1.414215	1.414215	0.0
0.505023	1.400576	1.416550	1.1
0.510755	1.384183	1.411233	1.9
0.519186	1.363683	1.407617	3.2
0.528296	1.336391	1.387052	3.8
0.541061	1.301629	1.360084	4.5
0.555448	1.263673	1.329578	5.2
0.570418	1.224203	1.296612	5.9
0.588923	1.184082	1.261728	6.5
0.607912	1.143937	1.225357	7.1
0.631125	1.104274	1.187889	7.5
0.653397	1.065489	1.149688	7.9
0.676857	1.027884	1.111082	8.1
0.699828	0.991682	1.072366	8.0
0.724829	0.957031	1.033800	8.0
0.751272	0.924027	0.995607	7.7
0.778367	0.892715	0.957981	7.3
0.806918	0.863114	0.921082	6.7
0.832927	0.835207	0.885049	6.0
0.863191	0.808955	0.849980	5.1
0.889806	0.784180	0.816030	4.1
0.919464	0.760722	0.783130	2.9
0.949456	0.738514	0.753263	2.0
0.977672	0.720758	0.728202	1.0
1.000000	0.707107	0.707107	0.0

Tabla 5.4: Datos numéricos en $\theta = 45^o$.

θ (grados)	Sol. Analítica	Sol. Numérica	Error (%)
0.00	0.000000	0.000000	0.0
3.75	0.098360	0.102695	4.4
7.50	0.194725	0.204994	5.3
11.25	0.288343	0.306458	6.3
15.00	0.380192	0.406634	6.9
18.75	0.470309	0.505074	7.4
22.50	0.558502	0.601353	7.7
26.25	0.644458	0.695053	7.8
30.00	0.727818	0.785768	7.9
33.25	0.808201	0.873103	8.0
37.50	0.885238	0.956677	8.1
41.25	0.958577	1.036122	8.1
45.00	1.027884	1.111082	8.1
48.75	1.092865	1.181210	8.1
52.50	1.153252	1.246164	8.0
56.25	1.166244	1.260033	8.0
60.00	1.259403	1.359140	7.9
63.75	1.304894	1.406389	7.7
67.50	1.345297	1.446851	7.5
71.25	1.380739	1.479887	7.2
75.00	1.411536	1.504604	6.6
78.75	1.438210	1.519697	5.7
82.50	1.461395	1.523261	4.2
86.25	1.480210	1.512751	2.2
90.00	1.485538	1.485538	0.0

Tabla 5.5: Datos numéricos en $r = 0.67$.

Capítulo 6

Conclusiones

En este trabajo se desarrollaron una serie de bibliotecas numéricas orientadas a objetos para la construcción de mallas ortogonales. Aunque las herramientas desarrolladas aquí están dirigidas a resolver el problema particular de la construcción de mallas, es posible reusar el código para resolver otro tipo de problemas. En particular fue posible reusar dos clases para resolver la aplicación que se mostró en la sección 5.3 de esta tesis.

El método presentado aquí, en principio puede ser usado para generar mallas ortogonales sobre cualquier dominio de forma arbitraria y que sea simplemente conexo. El esquema está basado en el concepto de descomposición de la transformación ortogonal global, en dos mapeos consecutivos: un mapeo conforme y un mapeo ortogonal auxiliar, sugeridos por Kang y Leal [14]. El método es no iterativo y puede ser bastante útil para problemas donde las mallas sean adaptivas.

El método se aplicó a diferentes dominios irregulares, pero no muy complejos, obteniendo resultados aceptables. El método permite especificar a priori la correspondencia de las coordenadas de la frontera en dos segmentos adyacentes, la correspondencia de los otros dos se calcula durante la solución. Se observó que un cambio en la especificación de las coordenadas, afecta la distribución de los puntos interiores de la malla, así como su ortogonalidad.

Por otro lado, si el contorno del dominio es bastante complejo, como es el caso de la frontera de Inglaterra, el sistema falla cuando se quiere resolver el sistema lineal resultante de las ecuaciones 3.23. Posiblemente la falla se deba a que la función de distorsión, debido a las irregularidades del dominio, contiene valores que difieren mucho entre sí, lo cual produce un sistema mal condicionado, el cual es difícil de tratar. El problema podría resolverse si se

rompe en regiones más simples al dominio irregular, y entonces se construyen mallas sobre cada subregión para luego unirlas. En este caso se debe tener bastante cuidado con la distribución de los puntos sobre las fronteras para no tener problemas en la unión. El problema también podría ser resuelto usando el método propuesto por Oh y Kang [15] donde el mapeo ortogonal auxiliar es más elaborado y la función de distorsión se puede suavizar de alguna manera. Sin embargo, estas soluciones son temas de trabajo futuro.

Se utilizó la malla ortogonal de un semicírculo para resolver la ecuación de Laplace en ese dominio. La forma de la ecuación de Laplace en coordenadas curvilíneas se simplifica bastante debido al uso de la malla ortogonal. Los resultados numéricos obtenidos difieren en un máximo del 8 % de la solución analítica. Esta diferencia puede reducirse si se refina la malla, ya que la solución numérica se obtuvo usando el método de volumen de control cuya precisión depende de la fineza de la malla.

Los resultados en cuanto a eficiencia del sistema son aceptables. Esto se debe principalmente al uso de bibliotecas orientadas a objetos optimizadas que se basan en BLAS. En un trabajo futuro se espera implementar todo el sistema completamente en C++ haciendo uso de la biblioteca estándar. Esta biblioteca contiene distintos algoritmos optimizados para el uso de la memoria y para cálculos numéricos.

Bibliografía

[1] Barret, R., Berry, M., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C. & Van Der Vorst, H., *Templates for the Solution of Linear Systems: building Blocks for Iterative Methods*, SIAM Press, 1994.

[2] Boehm, B.W., "A spiral model of software development and enhancement", *Software Engineering Notes*, **11**(4).

[3] Booch, G., *Object-Oriented Analysis and Design with Applications*, Benjamin-Cummings, Redwood City, CA, 1994.

[4] Cary,J.R., Shasharina,S.G:, Cummings,J.C., Reynders,J., & Hinker, P.J., "Comparison of C++ and Fortran 90 for Object-Oriented Scientific Programming", por publicarse en *Comput. Phys. Comm.*.

[5] Dongarra, J.J., Du Croz, J., Duff, I.S. & Hammarling, S., "A set of Level 3 Basic Linear Algebra Subprograms", *ACM Trans. Math. Soft.*, **16**, p.p. 1-17, 1990.

[6] Dongarra, J.J., Lumsdaine, A., Pozo, R. & Remington, K.A., "A Sparse Matrix Library in C++ for High Performance Architectures", Proceedings of the Object Oriented Numerics Conference, pp. 214-218, 1994.

[7] Dongarra, J.J., Lumsdaine, A., Pozo, R. & Remington, K.A., *IML++: Iterative Methods Library Reference Guide*, National Institute of Standards and Technology, `http://math.nist.gov/iml++`, 1994.

[8] Duraiswami, R. & Prosperetti, A., "Orthogonal Mapping in Two Dimensions", *J. Comput. Phys.*, **98**,254-268, 1992.

[9] Ça, L., "2D Orthogonal Grid Generation with Boundary Point Distribution Control", *J. Comput. Phys.*, **125**,440-453, 1996.

[10] Fletcher, C.A.J., *Computational Techniques for Fluid Dynamics*, Vol II, Second Edition, Springer-Verlag, 1991.

[11] George, P.L., *Automatic Mesh Generation: application to finite element methods*, Wiley, 1991.

[12] Jacobson, I., Christerson, M., Jonsson, P. & Overgaard, G., *Object-Oriented Software Engineering, A use case driven approach*, Addison-Wesley, ACM press, 1992.

[13] Jaswon, M.A. & Ponter, A.R., "An Integral Equation Solution of the Torsion Problem",

[14] Kang, I.S. & Leal, L.G., "Orthogonal Grid Generation in a 2D Domain via the Boundary Integral Technique", *J. Comput. Phys.*, **102**,78-87, 1992.

[15] Kang, I.S. & Oh, H.J., "A Non-Iterative Scheme for Orthogonal Grid Generation with Control Function and Specified Boundary Correspondence on Three Sides", *J. Comput. Phys.*, **112**,138-148, 1994.

[16] Kerlick, D.G. & Klopfer, G.H., "Assessing the Quality of Curvilinear Coordinate Meshes by Descomposing th Jacobian Matrix", in *Numerical Grid Generation*, ed. by J.F. Thompson, North-Holland, Amsterdam, pp 787-807, 1982.

[17] Lehman, M.M. & Belady, L., *Program Evolution. Process of Software Change*, London: Academic, 1985.

[18] O'Rourke, *Computational Geometry in C*, Cambridge, Cambridge University Press, 1994.

[19] Patankar, S.V., *Numerical Heat Transfer and Fluid Flow*, McGraw-Hill, 1980.

[20] Peraire, J., Vahdati, M., Morgan & Zienkiewicz, O.C., "Adaptive Remeshing for Compressible Flow Calculations", *J. Comput. Phys.*, **72**, 449-466, 1987.

[21] Pozo, R., *MV++ Matrix / Vector Classes Reference Guide*, National Institute of Standards and Technology, `http://math.nist.gov/mv++`, 1994.

[22] Ryskin, G. & Leal, L.G., "Orthogonal Mapping", *J. Comput. Phys.*, **50**,71-100, 1983.

[23] Steinberg, S. & Roache, P.J., "Variational Grid Generation", *Num. Meth. for P.D.E.'s.*, **2**,71-96, 1986.

[24] Stepanov A., Plauger, P.J., Lee, M. & Muser, D., *The Standard Template Library*, Prentice-Hall, 1999.

[25] Stroustrup, B., *The C++ Programming Language*, Third Edition, Addison-Wesley, 1997.

[26] Thacker, W.C., *Int. J. Numer. Meth. Eng.*, **15**,pp. 1335, 1980.

[27] Thompson, J.F., Warsi, Z.U.A., & Mastin, C.W., "Boundary Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations — A Review", *J. Comput. Phys.*, **47**,1-108, 1982.

[28] Thompson, J.F., "Grid Generation Techniques in Computational Fluid Dynamics", *AIAA Journal*, **22**,No. 11, pp. 1505, 1984.

[29] Thompson, J.F. Warsi, Z.U.A., & Mastin, C.W., *Numerical Grid Generation, Foundations and Applications*, North-Holland, 1985.

[30] Warsi, Z.U.A., "Tensor and Differential Geometry Applied to Analytic and Numerical Coordinate Generation", MSUU-EIRS-81, Mississippi State University, 1981.

[31] *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*, Editors: Ivo Babuska, Joseph E. Flaherty, John E. Hopcroft, William D. Henshaw, Joseph E. Oliger & Tayfun Tezduyar, Springer-Verlag, IMA Vol. **75**, 1995.

[32] The Object-Oriented Numerics Page, `http://www.oonumerics.org`.