Universidad Juárez Autónoma de Tabasco

División Académica de Ciencias Básicas

Mallado triangular y refinado de dominios en 2D

TESIS

QUE PARA OBTENER EL TíTULO DE:

Maestro en Ciencias Matemáticas Aplicadas

PRESENTA:

Yessenia Rincón Torres

DIRECTOR DEL TRABAJO:

Dr. Jorge López López

Cunduacán, Tabasco, Octubre 2019.

DIRECCIÓN

14 de agosto de 2019

Lic. Yessenia Rincón Torres
Pasante de la Maestría en Ciencias
en Matemáticas Aplicadas
P r e s e n t e.

Por medio del presente y de la manera más cordial, me dirijo a Usted para hacer de su conocimiento que proceda a la impresión del trabajo titulado **"MALLADO TRIANGULAR Y REFINADO DE DOMINIOS EN 2D",** en virtud de que reúne los requisitos para el EXAMEN PROFESIONAL POR TESIS DE MAESTRÍA para obtener el grado de Maestro en Ciencias en Matemáticas Aplicadas.

Sin otro particular, reciba un cordial saludo.

A t e n t a m e n t e.

Dr. Gerardo Delgadillo Piñón
Director

DIVISIÓN ACADÉMICA DE CIENCIAS BÁSICAS

C.c.p.- Archivo
Dr'GDP/Dr'JGPS/emt

Carta de autorización

El que suscribe, autoriza por medio del presente escrito a la Universidad Juárez Autónoma de Tabasco para que utilice tanto física como digitalmente la tesis de maestría: ***Mallado triangular y refinado de dominios 2D***, de la cual soy autor y titular de los Derechos de Autor.

La finalidad del uso por parte de la Universidad Juárez Autónoma de Tabasco de la tesis antes mencionada, será única y exclusivamente para difusión, educación y sin fines de lucro; autorización que se hace de manera enunciativa más no limitativa para subirla a la Red Abierta de Bibliotecas Digitales (RABID) y a cualquier otra red académica con las que la Universidad tenga relación institucional.

Por lo antes manifestado, libero a la Universidad Juárez Autónoma de Tabasco de cualquier reclamación legal que pudiera ejercer respecto al uso y manipulación de la tesis mencionada y para los fines estipulados en éste documento.

Se firma la presente autorización en la ciudad de Villahermosa, Tabasco a los 28 días del mes de Octubre del año 2019.

Yessenia Rincón Torres
102A10005

Índice general

Dedicatoria

Para Abdías y Ammi.

Agradecimientos

A Dios que abrió las oportunidades cada vez que parecía cerrarse el camino.

A mi esposo quien me apoyó y alentó para continuar, en los momentos más complicados.

A mis padres quienes me apoyaron todo el tiempo y no dejaron de creer en mí.

A mi amigo y asesor de tesis, el Dr. Jorge López López por la paciencia que ha tenido para conmigo, estuvo dispuesto a darme su tiempo y conocimiento, sin él no lo hubiese logrado. Gracias.

A mis maestros quienes nunca dejaron de enseñarme, sin importar que muchas veces me vi tentada a renunciar; ellos siempre estuvieron dispuestos a apoyarme para lograr concluir este trabajo.

A los sinodales quienes estudiaron mi tesis, sugirieron correcciones y la aprobaron.

A la Filantrópica Educativa de Tabasco por su apoyo incondicional.

A mis amigos Alma Yanely Martínez y Gerardo Barragán; ellos que estuvieron conmigo para darme su compañía, consejos y adecuaron mis horarios para que pudiese volver a la universidad.

A cada uno de los que de alguna u otra forma aportaron un granito de arena para hacer de este proyecto una realidad. Muchas gracias.

Yessi Rincón

Introducción

Una gran diversidad de fenómenos deterministas en la Física, Química, Biología, Economía y otros campos como la Medicina e Ingeniería, se modelan por medio de ecuaciones diferenciales. Estas ecuaciones diferenciales reflejan el carácter complejo de dichos fenómenos, y en su mayoría son imposibles de resolver por medio de métodos analíticos o tradicionales.

En estos casos se cuenta con el recurso de los métodos de aproximación. Dentro de la gran variedad de estos métodos se encuentra el método de elemento finito. La idea fundamental de este método es dividir la región de estudio en un conjunto de elementos o subregiones, llamadas elementos finitos. En cada una de estas regiones se aproximan las funciones involucradas en la ecuación diferencial, incluyendo las incógnitas. Para poder realizar esta idea en la computadora se deben construir mallas computacionales. La generación de mallas tiene como finalidad principal la resolución de ecuaciones diferenciales parciales que describan algún fenómeno en particular.

Al aproximar numéricamente la solución de un problema de modelado continuo, es necesario convertirlo de continuo a un conjunto finito de puntos. La selección de puntos es determinada una vez definida una malla. Una malla es un conjunto de caras poligonales (cuadriláteros, triángulos o tetraedros) que definen una superficie en el espacio tal que si t_i y t_j son dos elementos de una malla T, entonces: $t_i \cap t_j$ es un vértice común, o una arista común, o una cara común, o el conjunto vacío. Una malla tiene asociada un conjunto de elementos topológicos tales como: vértices y aristas en $2D$ y vértices, aristas y caras poligonales en $3D$.

En problemas simples, la malla puede ser elegida a priori, por ejemplo cuando la región donde estamos trabajando es un rectángulo, pero ¿qué pasa cuando tenemos regiones irregulares? Entonces el conjunto discreto de puntos de la región debe ser adaptada a la forma de esa zona.

La generación numérica de mallas juega un papel fundamental para so-

lucionar un problema en el cual la geometría de una región es compleja o cuando la solución tiene una estructura muy complicada. Sin embargo, con las actuales herramientas de software, la generación de mallas sigue siendo un problema computacional científico que conlleva gran parte de esfuerzo y esto hace que solucionar el problema original sea aún más complicado. Entonces, la investigación acerca de la generación de mallas está principalmente dirigida hacia la elaboración de algoritmos que las generen de manera automática, es decir, que el esfuerzo necesario por parte del usuario para generar una malla sea reducido al mínimo, para que sea la computadora la que realice la mayor parte del trabajo.

Una malla puede ser de diferentes formas poligonales como ya se ha mencionado, sin embargo en este proyecto nos enfocaremos en las mallas triangulares. La triangulación de un dominio es el primer paso en la solución computacional de problemas de ciencia e ingeniería, en donde es posible aplicar la técnica del elemento finito. Triangular una región plana significa dividir en triángulos esa región. A esta región plana la denotaremos por Ω. Una malla triangular es un conjunto de vértices descritos por sus coordenadas, a los vértices se les llama nodos de la malla y de un conjunto de triángulos que les llamaremos elementos de la malla. Tanto a los vértices como a los triángulos se les da una numeración para identificarlos. Los triángulos están compuestos por tres vértices, que se ordenan en sentido contrario a las manecillas del reloj. Al mallado del dominio o región plana Ω se le llama discretización. A menudo la discretización inicial no es la apropiada a las características del problema y por ello puede no ser definitiva. Tal es el caso en problemas en donde se desea precisar más la resolución o en regiones con alta no linealidad donde la única forma de representar fielmente la región es refinando la discretización local. Un refinamiento o derrefinamiento local de la malla no debería implicar una reconstrucción completa de las estructuras de datos que modelan la malla, al contrario, ésta debe de ser de carácter local debido al costo computacional que ello conlleva. Así que un refinamiento es hacer una subdivisión regular de los elementos de la malla dada, esto consiste en convertir un elemento en cuatro pequeños elementos que dependerán de cada triángulo de la malla grande. La subdivisión de mallas tiene varias aplicaciones en la resolución numérica de problemas, la más importante es generar mejores resultados, es decir, obtener una mejor resolución de las variables que intervienen en el problema. Por lo que que podremos tener datos más precisos sobre una parte de la región Ω.

Para la modelación de problemas físicos es necesario conocer la solución

de algunas ecuaciones diferenciales parciales. Considerando que en una región irregular no es posible obtener una solución analítica, se recurre a los métodos numéricos para aproximar la solución. La discretización del dominio es un problema a resolver necesario para algunos métodos numéricos. Cuando se tiene una malla no-estructurada, la técnica más usual para resolver numéricamente EDP's es el método de elemento finito (MEF). La precisión del método mejora si la calidad de la malla es buena, es decir, cuanto más fina sea la malla el resultado será mejor. Con ello podemos verificar la propiedad de convergencia del MEF salvo limitaciones de redondeo. Además de poder resolver un problema de EDP's con dos incógnitas en donde la solución a las incógnitas está una en la malla gruesa y la otra en la malla fina.

El trabajo que presentamos, se centra en la refinación de mallas triangulares. Utilizamos triángulos por su simplicidad y flexibilidad para cubrir regiones irregulares. En el capítulo uno se encuentra información de las mallas triangulares: su definición, la estructura de las mismas y mostramos un algoritmo para generar una malla triangular para cubrir un rectángulo; posteriormente se construye una malla triangular para un cuadrilátero convexo, usando las ideas para mallar un retángulo pero involucrando una transformación afín para transformar el cuadrilátero en un rectángulo. En el capítulo dos se muestra cómo generar un refinamiento a partir de una malla triangular dada, el lector podrá encontrar la descripción de las subrutinas y el programa que se ha utilizado para la generación del mismo. En el capítulo tres, se encuentran dos aplicaciones del mallado triangular en 2D: la convergencia del método de elemento finito y se describe un problema de ecuaciones diferenciales parciales con mallas mixtas, es decir se utilizarán las dos mallas de manera simultánea para aproximar correctamente la solución.

Capítulo 1
Mallas triangulares

Introducción

Una malla triangular consiste de un conjunto de vértices descritos por sus coordenadas y de un conjunto de triángulos, a los cuales se les llama nodos y elementos de la malla, respectivamente. Es necesario que los vértices y los triángulos se puedan identificar y esto lo haremos mediante una numeración para los nodos y los elementos de la malla; así podremos también conocer los nodos que conforman a cada elemento. Además de establecer un orden en los vértices de los elementos, es decir, decidir que nodo corresponde al primer vértice y la orientación en la cual recorreremos los demás y asociarles el número de nodo correspondiente. Esto nos conlleva a tener una numeración de nodos globales y numeración de nodos locales, donde la numeración global corresponde al orden establecido para contar los nodos de la malla sin tener en cuenta los triángulos y la numeración de nodos locales son los que nos indican que nodos forman a cada elemento, abordaremos dichas numeraciones en las siguientes secciones.

1.1. Triangulación

La triangulación de una región no es arbitraria, es decir, se deben cumplir las siguientes condiciones: Si T_1 y T_2 son dos triángulos de la malla, sólo se permiten las siguientes posibilidades,

1. Los triángulos son ajenos, $T_1 \cap T_2 = \emptyset$.

2. Se intersectan en un vértice.

3. Se intersectan en toda una arista.

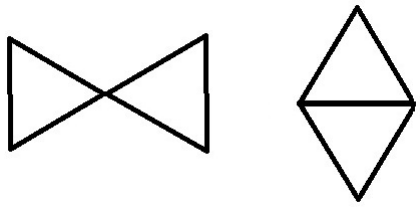


Figura 1.1: Triangulación permitida.

Las condiciones anteriores nos dicen que los triángulos de la malla no pueden estar ubicados como en la figura 1.2.

1.1.1. Numeración de nodos y triángulos

La numeración de nodos y elementos no es única, pues se podrían numerar como uno quiera, sin embargo si se desean hacer cálculos computacionales con ellos es bueno establecer la numeración de la manera más sencilla posible, por ejemplo numerar los nodos de forma consecutiva respecto a su cercanía, como se muestra en las siguientes secciones.

Ahora bien, los elementos se numeran estableciendo reglas para recorrerlos todos de la forma más sencilla. Los identificamos por la numeración escogida y por cada nodo que lo conforma. Luego, la malla entonces será el arreglo que contiene las coordenadas de cada nodo global y el arreglo que nos dá las coordenadas de cada nodo local, es decir, los nodos de cada elemento (figura 1.3).

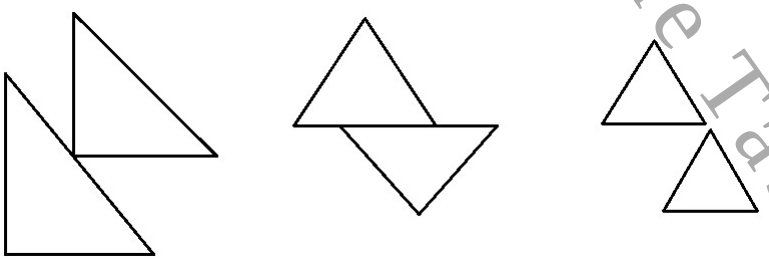


Figura 1.2: Triangulación no permitida.

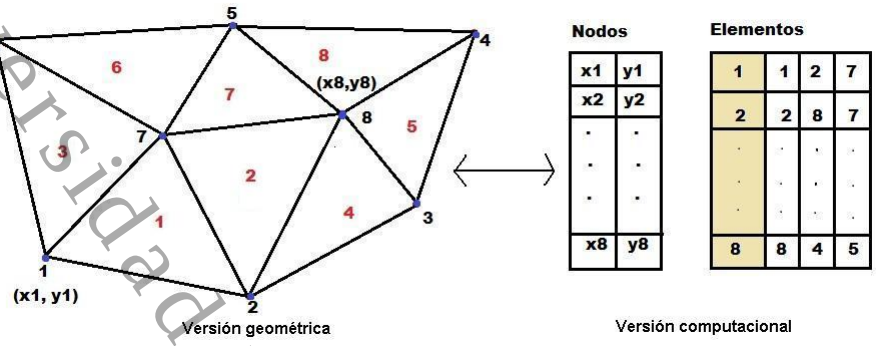


Figura 1.3: Malla triangular.

Cuando se define un triángulo de la malla, éste está conformado por tres vértices, los cuales determinan tres aristas. Pero se ha de tener en cuenta que dichos vértices pueden ser recorridos en sentido horario (de las manecillas del reloj) o en antihorario (contrario a las manecillas del reloj). De acuerdo a como se tome el sentido de orientación, varían los futuros cálculos de forma directa, es decir, es necesario definir una orientación para numerar nuestros vértices pues de ello dependerá la interpretación correcta de los datos obtenidos en una malla triangular. Nosotros elegimos numerar los vértices en sentido contrario a las manecillas del reloj.

Existen tres formas de numerar los nodos de un elemento dado en sentido anti-horario que dependen de cuál es el primer vértice. En nuestro caso, posteriormente se definirá la regla que usamos al respecto.

La forma de representar una malla no es única, dos de ellas son la graficación de los nodos y triángulos que la conforman, y mediante los arreglos de coordenadas de nodos globales y de los vértices de los triángulos. Cada una de ellas tiene sus ventajas y desventajas, por ejemplo si tan sólo se tiene la gráfica de la malla no es posible hacer cálculos computacionales con ella, sin embargo si contamos con los arreglos de coordenadas los cálculos computacionales se pueden llevar a cabo, pero no podríamos ver su estructura. Aunque podemos graficarla y llegar a visualizarla, esto implicaría un poco más de trabajo. La malla puede ser portable, es decir, podemos llevarla a todas partes si contamos con los arreglos de coordenadas de los nodos y los vértices.

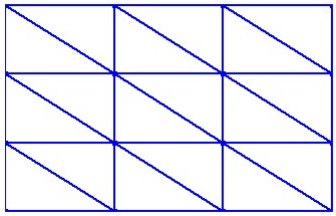


Figura 1.4: Malla uniforme para un rectángulo.

1.2. Generación de una malla uniforme para un rectángulo

Una de las regiones más simples de triangular es un rectángulo con aristas paralelas a los ejes. Además, en muchos casos es posible triangular una región más general, haciendo una transformación de este tipo de rectángulos a esa región. Así que es importante, saber mallar un rectángulo de este tipo.

Si pensamos en triangular un rectángulo de la manera más sencilla, seguramente procederemos dividiendo la base del rectángulo en nx partes y la altura en ny partes, lo cuadriculamos y luego trazamos diagonales en cada rectángulo interno, como se muestra en la figura 1.4. A esta malla se le llama una *malla uniforme*. Enseguida mostraremos un algoritmo para realizar esta triangulación.

1.2.1. Algoritmo para mallar un rectángulo

Un rectángulo está determinado de manera única por dos vértices opuestos, por ejemplo, el vértice inferior izquierdo (x_i, y_i), y el vértice superior derecho (x_f, y_f).

Nuestro objetivo es, para los puntos dados (x_i, y_i), (x_f, y_f), nx y ny, generar la malla: vértices, aristas y triángulos en forma automática. Esto lo haremos llevando a cabo los siguientes tres pasos.

Paso 1. Calculamos la base y la altura de los triángulos.

$$hx = \frac{xf - x0}{nx}$$

$$hy = \frac{yf - y0}{ny}$$

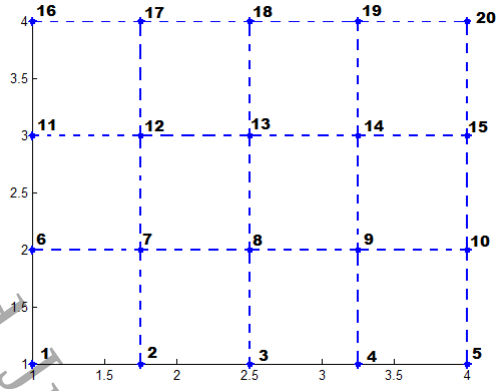


Figura 1.5: Posible numeración para los nodos de la malla de un rectángulo.

Paso 2. Generamos las coordenadas de los nodos globales (y con ello las coordenadas de los vértices de los elementos).

k=0; Para j=1:ny+1

Para i=1:nx+1
k=k+1;
DCOOR(k,1) = x0+(i -1)*hx;
DCOOR(k,2) = y0+(j -1)*hy;
Terminar i

Terminar j

En el arreglo DCOOR guardamos las coordenadas de cada nodo de la malla que estamos construyendo. El algoritmo anterior nos genera los nodos de la malla para el rectángulo dado y lo hace como en la figura 1.5.

Paso 3. Etiquetamos los triángulos y simultáneamente establecemos qué nodos corresponden al primero, segundo y tercer vértice de cada uno de ellos. La regla de numeración que usamos es numerar los triángulos de abajo hacia arriba de izquierda a derecha horizontalmente (véase la figura 1.6), es decir, el número de un elemento dependerá de la columna y fila en que se encuentre dentro de la malla, por ejemplo, si queremos numerar los triángulos que están en la fila i y la columna j, los números que le corresponden dependerán de la ubicación del elemento, esto es:

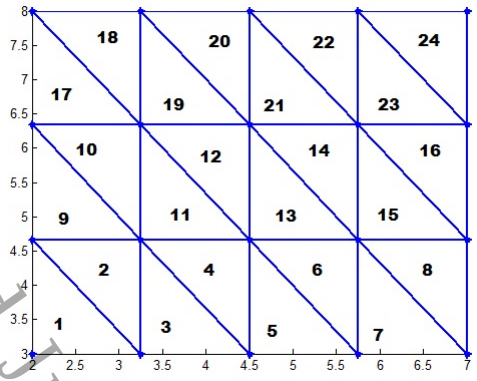


Figura 1.6: Numeración de los elementos de una malla

Para un triángulo superior: $2nx*(i-1)+2j$ y para un triángulo inferior: $2nx*(i-1)+2j-1$

Una vez realizada la numeración de triángulos, cada triángulo puede ser descrito por medio de los vértices que lo definen, iniciando del vértice uno hasta el vértice tres, estableciendo la numeración de éstos como sigue: para el triángulo inferior (como el triángulo 1 de la figura 1.6) se tomó como primer nodo al que está más arriba y para el triángulo superior (como el triángulo 2 de la figura 1.6) el primer nodo es el que está más abajo. Por ejemplo, en la figura 1.7 se observa la descripción para el triángulo 13.

Éstas ideas se resumen en el siguiente algoritmo, con el cual se numeran los elementos y se establece la relación de nodos locales con nodos globales, es decir, define qué nodos globales forman cada triángulo.

```
Inicio de algoritmo
e = 0;
Para j = 1:ny

    a = j*(nx+1)+1;
    b = a - (nx+1);
    c = b+1;
    d = a+1;


        Para i = 1:nx
        e = e+1;
```

```
            elementos(e,1) = a;
            elementos(e,2) = b;
            elementos(e,3) = c;
            e = e+1;
            elementos(e,1) = c;
            elementos(e,2) = d;
            elementos(e,3) = a;
            a = a+1;
            b = b+1;
            c = c+1;
            d = d+1;
            terminar i
        terminar j

    terminar algoritmo
```

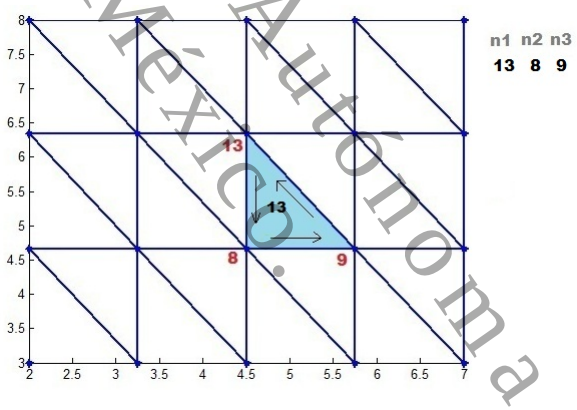


Figura 1.7: Triángulo 13 descrito por sus nodos

Hemos generado una malla triangular, con los vértices numerados horizontalmente y la diagonal orientada hacia la izquierda (figura 1.8). Es importante recordar que el dibujo sólo es una ilustración de la malla o una representación de la malla, pero también el arreglo DCOOR y el arreglo elementos son una representación de la malla, teniendo cada uno ventajas y desventajas como ya se mencionó. Cabe mencionar que junto a lo descrito se han definido las aristas frontera en términos de sus nodos guardando la información en el arreglo *bedges*.

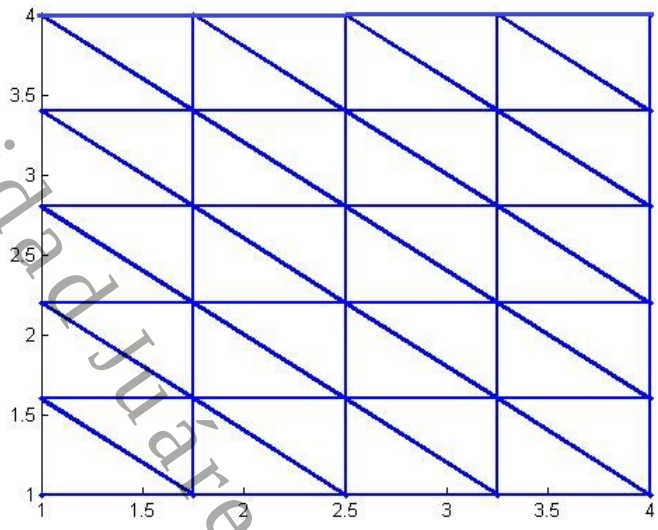


Figura 1.8: Malla triangular uniforme de un rectángulo.

1.3. Malla triangular uniforme para un cuadrilátero convexo

En esta sección daremos un algoritmo para mallar un dominio con forma de cuadrilátero convexo, modificando el algoritmo para mallar un rectángulo. Esto es posible ya que el rectángulo es un caso particular de un cuadrilátero convexo. Por lo que podemos pensar que no es necesario hacer muchos cambios en el algoritmo para mallar uniformemente un rectángulo,con el fin de obtener una malla para un cuadrilátero convexo como el que se muestra en la figura 1.9.

Siguiendo las ideas previas escogemos un lado que tomaremos como base de nuestro cuadrilátero, lo cual define también los lados que se tomarán como las alturas, luego dividimos la línea de la base y la línea que sirve de tapa con el mismo número de divisiones uniformes nx, análogamente se dividen las alturas uniformemente con ny divisiones. Este proceso define un mallado rectangular de líneas que se intersectan como se muestra en la figura 1.9. Posteriormente, cada subcuadrilátero se divide en dos triángulos al trazar las diagonales orientadas hacia la izquierda con respecto a la línea que sirve

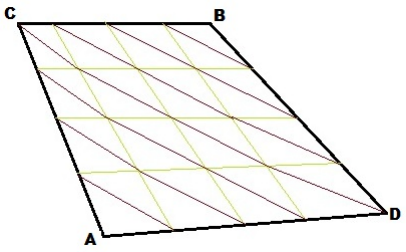


Figura 1.9: Mallado de un cuadrilátero convexo siguiendo el mallado de un rectángulo. Se indica también cómo se etiquetan los vértices.

de base, obteniéndose así la malla triangular.

El algoritmo para lograr el mallado del cuadrilátero es básicamente el mismo, sin embargo ahora no sólo necesitamos dos vértices como en el caso del rectángulo, sino cuatro.

1.3.1. Algoritmo para mallar un cuadrilátero convexo

Recordemos que todo punto S de un segmento de recta puede ser descrito en función de los extremos P_1, P_2 que lo contienen, por medio de la siguiente ecuación

$$S = (1 - \alpha) * P_1 + \alpha * P_2 \tag{1.1}$$

Con esto, los pasos para obtener el mallado serán:

Paso 1. Definir la base y las alturas.

Paso 2. Definir nx, el número de divisiones en la base y my el número de divisiones en las alturas.

Paso 3. Haciendo uso de la ecuación (1.1), se numeran los nodos de la base recorriéndolos de izquierda a derecha y a continuación se numeran los nodos de la siguiente línea "horizontal", continuando con las siguientes líneas horizontales hasta llegar a la tapa. La numeración de los nodos en las horizontales será siempre de izquierda a derecha.

Los pasos anteriores se resumen en el siguiente algoritmo.

```
Para j=1:ny+1
alpha=(j-1)/ny;
X0=(1-alpha)*A+alpha*C;
Y0=(1-alpha)*D+alpha*B;


    Para i=1:nx+1
    alpha=(i-1)/nx;
    k=k+1;
    DCOOR(k,1) =(1-alpha)*X0(1)+alpha*Y0(1);
    DCOOR(k,2) = (1-alpha)*X0(2)+alpha*Y0(2);
    Terminar

Terminar
```

1.4. Renumeración de los nodos

En una malla triangular podemos encontrar o derivar información adicional, como por ejemplo las aristas que forman un triángulo, o el conjunto de todas las aristas que conforman la malla. En los problemas que se describen por medio de Ecuaciones Diferenciales Parciales la frontera de la región Ω denotada por $\partial\Omega$, usualmente está dividida en dos partes: una parte Γ_0 donde la solución es conocida, en donde se imponen condiciones escenciales o de tipo *Dirichlet* y otra parte Γ_1 donde la solución no es conocida , en donde se imponen condiciones naturales o de tipo *Neumann*. No se admite que una arista de una malla pueda contener parte de la frontera Γ_0 y parte de la frontera Γ_1 simultáneamente. Es decir, cualquier arista que contenga parte de Γ_0 debe estar completamente contenida dentro de Γ_0.

Sabemos que en los nodos de la frontera Dirichlet, se proporcionan valores conocidos, y es necesario encontrar los valores en el resto de nodos que denominamos "nodos interiores". Por cuestiones de implementación práctica de los algoritmos, es conveniente hacer una renumeración de los nodos, de tal manera que los nodos de frontera Dirichlet aparezcan al principio o al final de la lista de nodos, como se muestra en la figura 1.10.

Para renumerar los nodos de la malla se deben conocer el número total de nodos de la malla *nn*, el número total de nodos frontera *nb* y el número total

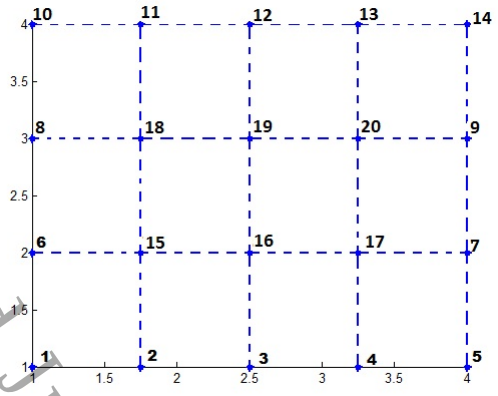


Figura 1.10: El orden de los nodos ha cambiado para ser los nodos frontera los primeros de la lista de nodos.

de nodos interiores ni. Para esto, lo primero que hacemos es identificar todos los nodos frontera y también los interiores, con esto se obtendrán los valores de nb y ni. Los datos de la malla original están en el arreglo DCOOR y la malla renumerada se almacenará en un nuevo arreglo newDCOOR. Para llenar el arreglo newDCOOR se recorrerán todos los nodos en DCOOR, preguntándose si es frontera o no, si lo es, sus coordenadas se almacenarán en los primeros nb renglones de newDCOOR y si es interior se irán almacenando a partir del renglón $nb+1$ de newDCOOR. Esta renumeración de nodos implica que se ha modificado la relación de nodos locales con nodos globales, pero para la nueva relación basta saber qué nuevo número j se le asignó a cada nodo i de la numeración original, esta información está contenida en el arreglo *permut*. De tal manera que $permut(i) = j$ significa que el nodo i ahora es el nodo j, entonces la nueva relación de nodos locales con nodos globales para un elemento cualquiera e, está dada por $newelement(e, l) = permut(element(e, l))$, para $l = 1, 2, 3$, lo mismo sucede con los nodos que forman las aristas frontera, deben permutarse adecuadamente.

Lo anterior se resume en el siguiente algoritmo.

```
function [newnode, newelement, newbedge, nn, ne, ni, nb, nbed] =
renummesh(node,element,bedge)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PROGRAM TO RENUMERATE THE NODES OF A GIVEN MESH. THE  RENUMERATION %%
%% IS  DONE  SO  THAT  THE BOUNDARY NODES ARE THE FIRST NODES AND THE %%
%% INTERIOR NODES COME AFTER THE BOUNDARY NODES                       %%
%% INPUT ARGUMENTS: node, element, bedge.                            %%
%% OUTPUT ARGUMENTS: newnode, newelement, newbedge, nn, ne, ni, nb,  %%
%% nbed                                                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Number of node;
nn = length(node);
% Number of element
ne = length(element);
% Number of edges on the Dirichlet boundary
nbed = length(bedge);
% Total number of edges
nted = nn + ne - 1;

% typenode is a flag to distinguish nodes on the boundary from those in the
% interior. A node is on the boundary if the flag is 0 and interior if it
% is 1
typenode(1:nn) = 1; % Initialize all as interior.
% Change the flag to 0 for boundary node
for i = 1:nbed
    typenode(bedge(i,1)) = 0;
    typenode(bedge(i,2)) = 0;
end

% Numero de nodo frontera
nb = 0; % inicializamos en cero
for i = 1:nn
   if typenode(i)==0
      nb = nb + 1; % node i is on boundary
   end

end
% Number of interior node
ni = nn - nb;

%% RENUMERATION
```

```
kb = 0;
ki = nb;
for i=1:nn
    if typenode(i) == 0
        kb = kb+1;
        permut(i)=kb;
        newnode(kb,1) = node(i,1);
        newnode(kb,2) = node(i,2);
    else
        ki = ki+1;
        permut(i)=ki;
        newnode(ki,1) = node(i,1);
        newnode(ki,2) = node(i,2);
    end
end

for e = 1:ne
    newelement(e,1) = permut(element(e,1));
    newelement(e,2) = permut(element(e,2));
    newelement(e,3) = permut(element(e,3));
end

for i=1:nbed
    newbedge(i,1) = permut(bedge(i,1));
    newbedge(i,2) = permut(bedge(i,2));
end
```

El algoritmo anterior nos permite obtener una malla como la que se muestra de manera gráfica en la figura 1.11.

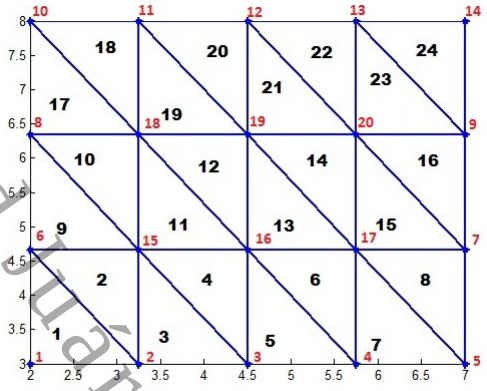


Figura 1.11: Malla con numeración de elementos y renumeración de nodos.

Capítulo 2
Refinamiento de Mallas Triangulares

2.1. Datos necesarios para refinar una malla

Una vez que se ha obtenido una malla podemos obtener datos adicionales que nos ayudarán a refinarla. Algunas aplicaciones del refinamiento de una malla son: para mejorar una solución numérica de una EDP o para resolver problemas en donde simultáneamente es necesario aproximar un tipo de funciones en una malla y otro tipo de funciones en una malla doblemente fina.

Las estructuras de datos ayudan de una manera fundamental a resolver este tipo de problemas. Éstas permiten el acceso rápido y la manipulación de la información, permitiendo simplificaciones significativas en la metodología y la estructura del código. También pueden reducir drásticamente los requerimientos de la unidad central de proceso en nuestros equipos de cómputo.

Como hemos visto en el capítulo anterior para una malla tenemos dos arreglos principales para almacenar los datos de ésta,

1. Arreglo con las coordenadas de los nodos.

2. Arreglo con los nodos que definen a cada elemento.

En la matriz DCOOR están las coordenadas de cada nodo y en la matriz elementos encontramos las triadas de nodos que definen a los triángulos de la malla. Como estamos usando mallas triangulares, la matriz elementos está dada de la siguiente forma:

elementos(1:nelem,1:3)

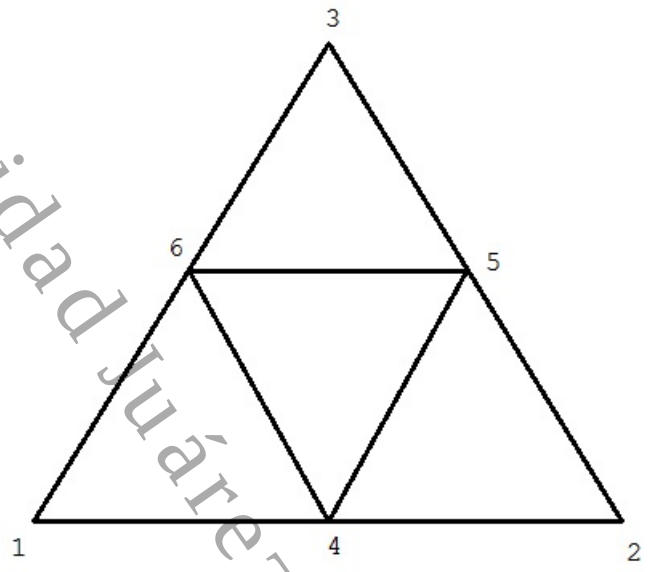


Figura 2.1: En cada arista se buscan los puntos medios y éstos serán los nuevos nodos de la malla fina.

El objetivo de este capítulo es refinar la malla original dividiendo cada triángulo en 4 subtriángulos como se muestra en la figura 2.1. Para esto necesitamos conocer algunos datos más, como son: elementos rodeando un punto, puntos rodeando puntos, elementos rodeando elementos y elementos rodeando aristas. En las siguientes secciones se mostrarán las ideas principales y el algoritmo de los programas que calculan dichos datos. Se describen los programas en el orden en que son ejecutados con el fin de obtener el refinamiento de la malla gruesa:

1. esp (Elementos rodeando puntos)
2. psp (Puntos rodeando puntos)
3. ese (Elementos rodeando elementos)
4. edges1 (Todas las aristas de la malla)
5. esed (Elementos rodeando aristas)
6. subgra (Subdivide la malla)

2.1.1. Elementos Rodeando Puntos

La forma más eficiente de guardar datos variables es a través de listas enlazadas. En lugar de arreglos largos, en la subrutina *esp* (elements surrounding points) tenemos dos arreglos, esup2, de dimensión $nn + 1$, que básicamente guarda el número de elementos que rodean a cada nodo, en forma individual en una primera asignación ($esup2(i) = k$ significa que al nodo $i - 1$ lo rodean k elementos), en forma acumulada después (los elementos que rodean al punto i están almacenados en las posiciones esup2(i)+1,...,esup2(i+1) del arreglo esup1) y finalmente se recorren los datos una posición a la izquierda para fines de cálculos posteriores.

Con esto, en esup1 se almacenarán qué elementos están rodeando a cada nodo, y se identificarán con la ayuda de esup2, ya que además de contener el número de elementos que contiene a cada nodo, nos dá el inicio y el final de las ubicaciones de los datos contenidos en esup1. Para contabilizar cuantos elementos rodean a cada nodo, se recorrerá elemento por elemento siguiendo la numeración de éstos, una vez que se está en un elemento éste rodea a cada uno de sus vértices, entonces para cada elemento debe aumentarse en uno el número de elementos que rodean a cada nodo global asociado con el elemento en turno. Siguiendo este procedimiento hasta terminar con los elementos se tendrá completa la información en el arreglo esup2. Para identificar cuales son los elementos que rodean a cada nodo.

Para la malla mostrada en la figura 1.11, se obtuvieron los siguientes datos en los vectores esup1 y esup2. Centraremos nuestra atención en el nodo 5 al cual lo rodean los elementos 7 y 8, por eso se destacan los números 7 y 8 del vector esup1, que están en las entradas 11 y 12, respectivamente y en el arreglo esup2 se destaca el valor 11 en la entrada 5. Notemos que el vector esup2 nos indica el número de entrada en el que inician los elementos que están rodeando a cada nodo, y en el arreglo esup1 encontraremos los elementos que lo están rodeando.

esup1 =

1 1 2 3 3 4 5 5 6 7 **7** **8** 1 2 9 8 15 16 9 10 17 16 23 24 17 18 18 19 20 20 21 22 22 23 24 24 2 3 4 9 10 11 4 5 6 11 12 13 6 7 8 13 14 15 10 11 12 17 18 19 12 13 14 19 20 21 14 15 16 21 22 23

esup2 =

1 2 5 8 **11** 13 16 19 22 25 27 30 33 36 37 43 49 55 61 67 72

2.1.2. Puntos rodeando puntos (psp)

Para encontrar el número de puntos que rodean a un punto y saber qué lo está rodeando, utilizaremos los datos de los vectores esup1 y esup2. En la subrutina *psp* (points surrounding points) se obtendrán los vectores psup1 y psup2, que contendrán cuáles y cuántos puntos rodean un punto, respectivamente. A éstos se les denotará como sigue:

psup1(1:mpsup), psup2(1:npoin+1),

donde npoin es el número de nodos globales de la malla, mpsup es el tamaño del arreglo psup1. El arreglo psup2 guarda las posiciones en psup1 donde están almacenados los puntos que rodean a cada punto, similar a las listas de la subrutina *esp*, esup1 y esup2.

Sólo los puntos de los elementos vecinos pueden ser puntos que rodean a determinado punto. Conforme vayamos construyendo los arreglos psup1 y psup2, las entradas de éstos se guardan en un arreglo lpoin de tamaño npoin. Nos moveremos nodo por nodo, determinando los nodos que lo rodean de la siguiente manera: tomando a los elementos rodeando al nodo (información en *esup1* y *esup2*), note que si el elemento está rodeando al nodo significa que el nodo en turno pertenece a él, por lo que se contabilizan y almacenan únicamente los nodos diferentes a éste. Se inicia la contabilización a partir del nodo 1 local y hasta el nodo local del elemento que rodea al nodo, en el orden establecido en las listas esup1 y esup2.

En el arreglo *psup1* se encuentran todos los nodos (puntos) rodeando al nodo (punto) y en psup2 se nos indica el comienzo y el final de los datos dados en psup1, de manera similar como lo hacen esup1 y esup2.

Para ilustrar cómo funciona el programa *psp.m* seguiremos con el rectángulo de la figura 1.11. Al ejecutar el programa *mallacuadrada.m* se obtienen los siguientes datos en los vectores psup2 y psup1, enseguida de los vectores esup1 y esup2 encontrados por la subrutina *esp.m*.

psup1 =

6 2 6 1 15 3 15 2 16 4 16 3 17 5 17 4 7 1 2 15 8 5 17 20 9 6 15 18 10 7 20 13 14 8 18 11 **18 10 19 12** 19 11 20 13 20 12 9 14 9 13 2 6 3 16 8 18 3 15 4 17 18 19 4 16 5 7 19 20 15 8 16 19 10 11 16 18 17 20 11 12 17 19 7 9 12 13

psup2 =

0 3 7 11 15 18 22 26 30 34 **37** 41 45 49 51 57 63 69 75 81 87

En el texto anterior están los datos de los vectores psup1 y psup2, obtenidos al ejecutar el programa *mallacuadrada.m*, donde psup1 tiene qué puntos (nodos) están rodeando a cada nodo y el vector psup2 nos dice en que entrada inician los nodos rodeando a cada nodo, por ejemplo, el número 37 nos indica que los puntos que están rodeando al nodo 11 están almacenados a partir de la entrada 37 del vector psup1 y terminan en la 40, ya que en la entrada 41 inician los puntos que rodean al nodo 12, y así sucesivamente.

Para construir los arreglos psup2 y psup1 se ejecuta la subrutina *psp.m*.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%FUNCTION PSP FINDS THE NODES THAT SORROUND EACH POINT, The program psp %%
%%has the following INPUT ARGUMENTS:elements.-elements matrix, n.- number%%
%%of nodes of the mesh; esup1.-elements surroinding points of the mesh;  %%
%%esup2.-it stores the locations of esup1 where it starts and ends the   %%
%%nodes that sorround each point. This function returns three OUTPUT     %%
%%ARGUMENTS, psup1 stores the points, and the ordering is such that the  %%
%%points surrounding point ipoin are stored in locations psup2(ipoin)+1  %%
%%to psup2(ipoin+1). The maximun number of nodes sorrounding a point is  %%
%%stored in "major".                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [psup2,psup1,major]=psp(esup1,esup2,elements,n)
major=0;
lpoin(1:n)=0;      %%Initialization to zero of the help array lpoin.
psup2(1)=0;        %%Initializing location 1 of psup2 to 0 (zero).
istor=0;           %%Initializing the storage counter to 0 (zero).
for ipoin=1:n
    %%Loop over the elements surrounding the point
    for iesup=esup2(ipoin):esup2(ipoin+1)-1
        ielem=esup1(iesup);            %%Element number
        for inode=1:3                  %%Loop over nodes of the element
            jpoin=elements(ielem,inode);  %%Point number
            if(jpoin ~= ipoin & lpoin(jpoin)~=ipoin)
                %%Update the storage counter, store ahead, and mark lpoin
                istor=istor+1;
                psup1(istor)=jpoin;
                lpoin(jpoin)=ipoin;
            end
        end
    end
    %%Update the storage counter
    psup2(ipoin+1)=istor+1;
end
psup1 % es el vector donde encontramos todos los puntos que rodean a cada nodo,
      %dados en el orden de los elementos que lo est\'an rodeando.
psup2(1)=1;
```

```
for k=1:n-1
    diff=psup2(k+1)-psup2(k);
    if diff >= major
       major=diff;
    end
end
diff
psup2
```

2.1.3. Elementos rodeando elementos (ese)

Una vez obtenidos los elementos y los puntos que rodean un punto y con las subrutinas *esp.m* y *psp.m*, usaremos esta información para encontrar a todos los elementos que rodean a cada elemento dado de la malla, haciendo uso nuevamente de los vectores ya calculados con *esp.m* y *psp.m*. Conocer estos datos son de suma importancia para refinar la malla, puesto que esto consiste en subdividir cada elemento en cuatro elementos.

Para identificar a todos los elementos rodeando a un elemento, evaluaremos cada cara (arista) que lo compone y localizaremos el número de elemento que lo rodea, estos datos lo almacenaremos en un arreglo llamado esuel que se define como:

esuel(1:nelem, 1:nfael)

Donde nfael es el número de caras por elementos y nelem es el número de elementos de la malla: como en nuestro caso sólo estamos utilizando mallas triangulares, nfael=3. Para construir esta estructura de datos, debemos combinar los puntos que comprenden cualquiera de las caras de un elemento con aquellas caras de los elementos vecinos. Esta información la podemos obtener haciendo uso de los arreglos de elementos, esup1, esup2 calculados anteriormente y el arreglo de ayuda *lhelp*, en el cual se van marcando los elementos que rodean los puntos del triángulo que se está evaluando. Si un elemento es marcado dos veces quiere decir que ese triángulo tiene una arista en común con el elemento en evaluación, pues si dos triángulos comparten una arista entonces son vecinos, por lo que es almacenado en el arreglo *esuel*.

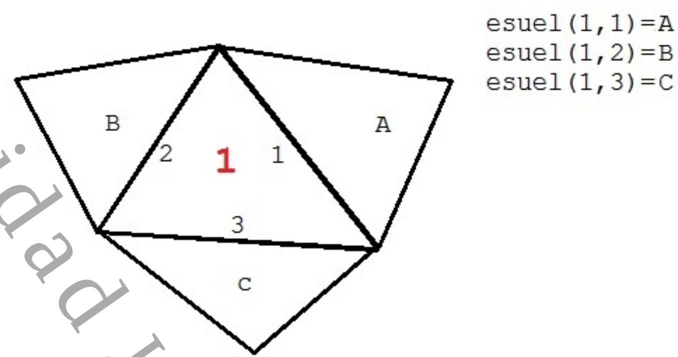


Figura 2.2: El arreglo esuel nos muestra a los elementos que están rodeando a un elemento asociando uno a cada arista, en el orden establecido para los elementos.

Entonces, el algoritmo para construir el arreglo esuel está en la subrutina *ese.m*, notemos que al inicio se solicitan lo datos almacenados en los vectores, elements, esup2, esup1.

```
function [esuel]=ese(e,elements,esup2,esup1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%FUNCTION ese2 HELPS TO GET ELEMENTS SORROUNDIG ELEMENTS, it has 4 input%%
%%arguments: e.-number of elements in the mesh; elements.-connectivity   %%
%%matrix of the elements;esup1.-it stores the elements; and the ordering %%
%%is such that the elements surrounding point ipoin are stored in        %%
%%location esup2(ipoin)+1 to esup2(ipoin+1). It also has an output       %%
%%argument: esuel.- it is the matrix were we find the elements           %%
%%sorrounding elements                                                   %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
esuel(1:3,e)=0;     %% Initialize the whole matrix to zero
lhelp(1:e)=0;       %% Help array to mark the elements that
                    %% sorround each point of one element
x= size(esup2)-1;
for ielem=1:e
    z=1;
    node=1;
        for ientry=1:3
            nnode=elements(ielem,ientry);
            location1=esup2(nnode);
            if (nnode==x(1,2))              %
                location2=esup2(nnode+1);
            else
```

```
                location2=esup2(nnode+1)-1;
            end
              % We look for the elements sorrounding each point of the
              % element. We mark that element in lhelp.
            for nelem=location1:location2
              if(lhelp(esup1(nelem))==0)
              lhelp(esup1(nelem))=lhelp(esup1(nelem))+1;
              flag=1;
              end
              if(lhelp(esup1(nelem))==1 & flag~=1 & esup1(nelem) ~= ielem)
                                                    %% If the element
              esuel(z,ielem)=esup1(nelem);          %% is marked twice,
              z=z+1;                                %% it means we found
              end                                   %% an element.
              flag=0;
            end
        end
    lhelp(1:e)=0;
end
esuel
esuel
```

2.1.4. Numeración de aristas

En el capítulo anterior vimos cómo se han numerado las aristas de la frontera. En esta sección analizaremos el algoritmo para la numeración de las aristas de toda la malla. Para ello se ha creado un arreglo llamado lpoin(1:nn, 1:major) donde nn es el número de nodos de la malla y major es el número máximo de puntos que pueden rodear a un punto. En este arreglo auxiliar se almacenarán los puntos que generan a cada arista de la malla.

En el programa *psp.m* se almacenaron todos los puntos que están rodeando a un punto, ahora se pueden generar las aristas frontera y las aristas interiores. Hemos dividido la generación y numeración de aristas en dos secciones: numeración de aristas frontera y numeración de aristas interiores.

Aristas frontera

Las aristas que se generan primero son las frontera, para después agregar las interiores. Lo primero que hacemos en la subrutina *edges1.m* es identificar a los nodos de la frontera, una vez que se los tiene identificados se usa el arreglo lpoin, obtenido en el programa *psp.m*. Se inicia la generación de aris-

tas frontera uniendo al nodo número uno con el siguiente nodo que lo rodea siempre y cuando tenga la etiqueta de ser frontera en sentido antihorario. Ver figura 2.3. Recuerde que los nodos de la malla han sido renumerados de tal forma que al principio se encuentran los nodos frontera.

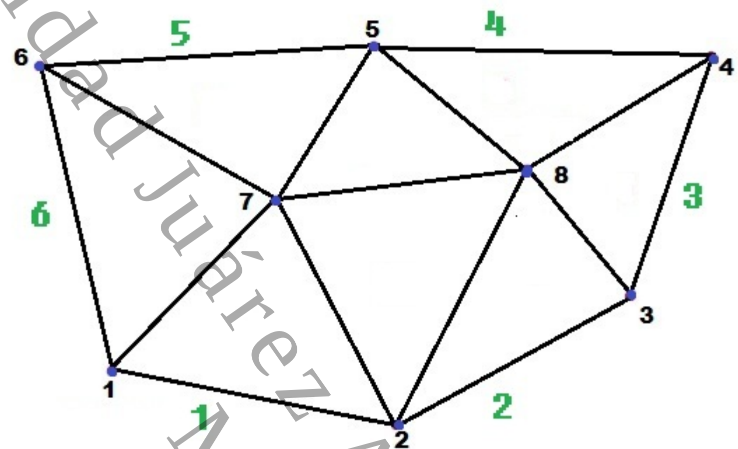


Figura 2.3: Los números de color verde son el orden en que se numeran las aristas frontera y los negros son el número de nodo de la malla triangular.

El algoritmo descrito debajo, muestra que lo primero que se hace con los nodos de la malla es identificar y etiquetar los nodos que se encuentran en la frontera, para después construir las aristas que la forman.

```
function [edges,counter,gedge]=edges1(psup1,psup2,nn,bedges,k,major)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%FUNCTION edges1 GETS EDGES OF A MESH, IT USES THE FOLLOWING INPUT      %%
%%ARGUMENTS: psup2 INDICATES THE POSITION IN psup1 WHERE POINTS          %%
%%SORROUNDING EVERY POINT; nn IS THE NUMBER OF NODES OF THE MESH; lpoin  %%
%%IT IS A HELP ARRAY WHERE WE STORE POINTS, THAT FORM THE EDGES, WITH    %%
%%EVERY POINT; bedges is THE MATRIX OF BOUNDARY EDGES; k IS THE NUMBER OF%%
%%BOUNDARY EDGES; major IS THE MAXIMUN NUMBER OF NODES SORROUNDING EACH  %%
%%NODE. OUTPUT ARGUMENTS ARE: edges IS THE EDGES MATRIX; counter IS THE  %%
%%TOTAL NUMBER OF EDGES; gedge IS AN ARRAY TO INDICATE IF THE EDGE IS A  %%
%%BOUNDARY EDGE OR AN INTERNAL EDGE.                                     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% lpoin is a help matrix where we store every point sorrounding
%% points.
```

```
lpoin(1:nn,1:major)=0;
for i=1:k
    lpoin(bedges(i,1),1)=bedges(i,2);
    lpoin(bedges(i,2),2)=bedges(i,1);
end
% 1. BOUNDARY EDGES
for l = 1:k
    edges(l,:) = bedges(l,:); % The first edges are on the boundary
    gedge(l) = 0;  % Flag for boundary edges
end
```

Teniendo identificados los nodos de la frontera se inicia la generación de de aristas frontera uniendo el primer nodo con el siguiente nodo frontera en el sentido contrario a las manecillas del reloj, la generación de aristas es de manera continua, por ejemplo, para el nodo 1 de la figura 2.4 se tienen tres nodos rodeándolo, sin embargo la primera arista estará definida del nodo 1 al nodo 2 y la dos será del 2 al 3 y así sucesivamente.

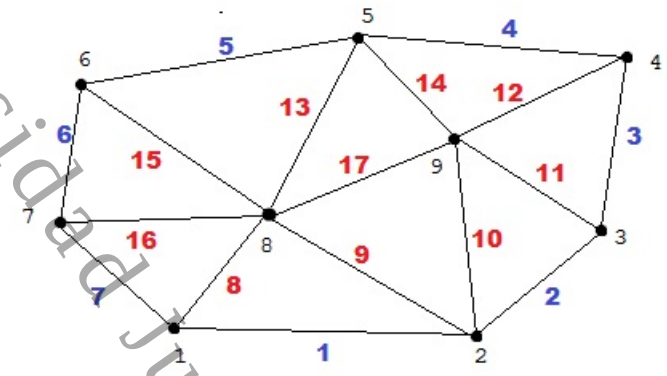
La numeración de las aristas frontera está definida a partir de la generación de éstas.

Aristas interiores

Una vez que ya se tienen a todas las aristas frontera, se seguirá con la generación de aristas interiores. Se inicia nuevamente en el nodo número uno. En los arreglos psup1 y psup2 se tienen a todos los puntos que rodean un punto, lpoin contiene a todos los frontera, ahora conocemos quienes son nodos interiores, por lo que se inicia a unir a cada nodo con los puntos restantes que lo rodean en el orden establecido en los arreglos psup1 y psup2 hasta terminar con el último nodo.

Enseguida se muestra la parte del algoritmo utilizado para generar las aristas interiores de la malla.

```
% 2. INTERNAL EDGES
iedge=k+1;
x=0;  %Flag to indicate if a edge alrady exists.
for inode=1:nn                       %For each node we will look in psup1 for
    location1=psup2(inode);         %the points sorrounding it.
    location2=psup2(inode+1)-1;     %We can find these nodes from location1
    for nnode=location1:location2   %to location2 given by psup2.
        for i=1:major                          %We look for nnode in lpoin,
            if(lpoin(inode,i)==psup1(nnode)) %it means that this edge was
```



psup1=

7 2 8 | 1 3 8 9 | 2 4 9 | 3 5 9 | 4 6 8 9

5 7 8 | 6 1 8 | 1 2 7 9 6 5 | 2 3 8 4 5

psup2=

1 4 8 11 14 18 21 24 30

Figura 2.4: Notemos que las aristas frontera se numeran continuamente a partir del nodo 1.

```
    x=x+1;                              %already built up.
    end
end
if (x == 0)                           %If this point isn't found we
    edges(iedge,1)=inode;             %we generate a new edge, this edge
    edges(iedge,2)=psup1(nnode); %is determinated by two nodes:
    gedge(iedge)=1;                   %inode and nnode.
    iedge=iedge+1;                    %Flag for internal edges
    for j=1:major
        if(lpoin(inode,j)==0)              %Where we find a space in
           lpoin(inode,j)=psup1(nnode);%lpoin(inode,j) we store
           break                           %nnode to indicate that this
        end                                %edge was built up, and
    end                                    %we break the cycle.
    for m=1:major
```

```
                if(lpoin(psup1(nnode),m)==0)   %Where we find a space in
                   lpoin(psup1(nnode),m)=inode;%lpoin((nnode),m) we store
                   break                       %inode to indicate that
                end                            %edge was already built up,
            end                                %and we break the cycle.
        end
        x=0;                                    %Reset of the flag.
        end
end
counter=iedge-1
edges
%Total number of edges
```

La numeración de las aristas frontera e interiores es muy importante, ya que al subdividir la malla original, los nuevos nodos de la malla refinada estarán definidos por los puntos medios de cada arista de la malla original, lo cual será abordado en secciones posteriores.

2.1.5. Elementos rodeando aristas (esed)

Elementos rodeando aristas (esed) nos proporciona la información acerca de qué elementos están rodeando a cada arista de la malla, notemos que a lo más el número de elementos que pueden rodear a una arista es dos. En todas las aristas frontera sólo se tiene un sólo elemento que lo rodea y en las aristas interiores se tienen a dos elementos rodeándo a una arista dada.

Los datos que esed.m necesita para poder funcionar son el número total de las aristas de la malla, qué nodos definen a cada arista, información que está en el arreglo edges, se necesitan los dos arreglos obtenidos en esp.m: esup1 y esup2, el número total de elementos dados por la variable e y la matriz elementos.

Lo primero que *esed.m* calcula es qué elementos están rodeando a cada una de las aristas de la malla usando información que anteriormente ha sido obtenida de los programas esp.m, psp.m y ese.m. Del arreglo edges tenemos identificados a los nodos de cada arista de la malla, entonces se toman los nodos que definen a cada arista en el orden establecido por el arrego edges, por medio de los arreglos esup2 y esup1 conocemos qué elementos están rodeando al nodo uno de la arista, se toma al nodo dos de la arista dada y nuevamente podemos conocer qué elementos lo están rodeando, si hay un elemento que rodea al nodo uno y al nodo dos de la arista al mismo tiempo, se almacena, ya que éste está rodeando a la arista.

Se genera el arreglo triangleedges, en él hallamos a cada elemento definido por sus aristas localmente ordenadas de 1 a 3, es decir, con ayuda de éste se conocen a cada elemento y qué aristas lo están formando. El orden establecido para las aristas, está definido por el orden de los nodos locales de cada elemento, es decir, la arista uno será la que une al nodo local 1 con el nodo local 2, la arista dos será la que une al nodo local 2 con el nodo local 3 y la arista tres la que une al nodo local 3 con el nodo local 1. Figura 2.5.

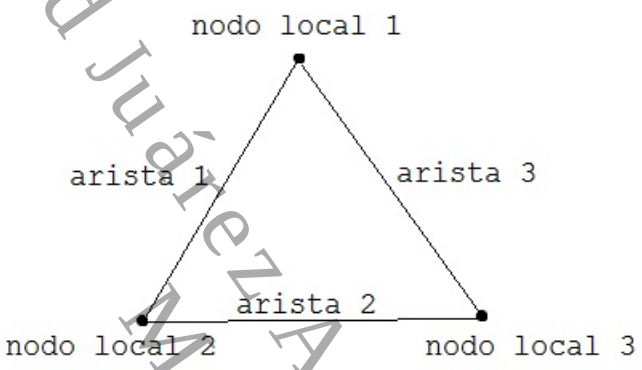


Figura 2.5: La numeración de aristas depende de la numeración local de los nodos.

Antes de realizar el proceso de subdivisión de una malla se ha generado la matriz triangleedges , en la que se almacenan las aristas que contiene cada elemento en forma ordenada. Para esto hemos construido el algoritmo ***esed***, el cual también genera un arreglo con los elementos a los que pertenece la arista dada.

2.2. Refinamiento o subdivisión de una malla

El ***refinamiento*** o subdivisión regular de elementos de una malla consiste en convertir un elemento en cuatro pequeños elementos como se muestra en la Figura 2.6. Como se ha mencionado en el primer capítulo, a una malla la definen sus nodos y elementos (triángulos), por lo que es muy importante conocer las coordenadas de los nodos y qué nodos definen a cada triángulo.

Para subdividir la malla se deben de encontrar los puntos medios de cada una de las aristas en el orden que éstas tienen, iniciando con las aristas frontera y concluyendo en las aristas interiores.

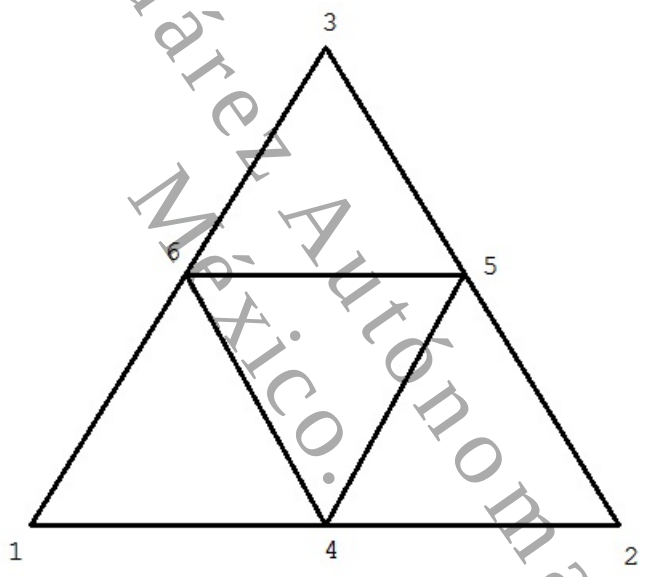


Figura 2.6: En cada arista se buscan los puntos medios y éstos serán los nuevos nodos.

Los nuevos nodos los llamaremos **nodos de velocidad** y a los nodos de la malla gruesa los llamaremos **nodos de presión**.

2.2.1. Numeración de nodos de velocidad

La numeración de los nodos de velocidad será semejante a los de la malla de presión, primero se numeran los nodos de la frontera de la siguiente manera: los nodos frontera de velocidad que ya existían en la malla gruesa quedarán con el mismo número que le correspondía en la malla gruesa, continuando la numeración en los nuevos nodos frontera, es decir en los puntos medios de las aristas frontera. Figura 2.7.

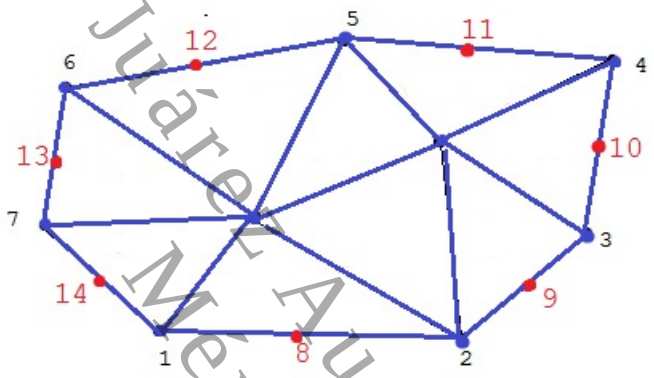


Figura 2.7: Los nodos velocidad se numeran a partir de los nodos de presión.

La numeración de los nodos interiores se hace de la siguiente manera: los primeros en numerarse serán los nodos que estaban en la malla gruesa iniciando en el número nn (número de nodos) - nb (nodos de frontera). Una vez que se han numerado los nodos de la malla gruesa se continúa con los nuevos nodos, esto se hace en el orden que han establecido las aristas, notemos que todos los nodos velocidad en las aristas frontera han sido numerados, los nodos de velocidad que ya eran de presión han sido numerados, y ahora los nodos que están en las aristas interiores deben ser numerados en el orden que éstas tienen con el número siguiente en el que se quedó la numeración. Lo ilustramos en la figura 2.8. El orden que se estableció en las aristas de la figura 2.8 ha sido la misma que se usó anteriormente en la figura 2.4.

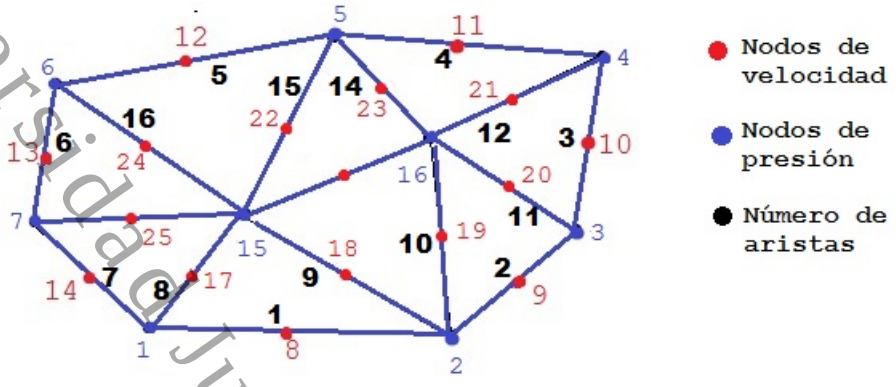


Figura 2.8: Numeración de nodos interiores.

2.2.2. Numeración de elementos de velocidad

Una vez que los nodos de frontera e interiores están numerados entonces se procede a la construcción y numeración de los nuevos elementos. Éstos se construyen a partir de los puntos medios de cada arista de presión, definiendo de cada elemento de la malla gruesa a cuatro elementos nuevos. En el capítulo 1, sección 1.1.1, se estableció que los nodos de cada elemento tienen una numeración global y local, por lo que para los elementos de velocidad también se usarán dichas numeraciones.

Como se ha mencionado de cada elemento de presión se obtienen cuatro elementos de velocidad, los números globales correspondientes a cada elemento nuevo está definido a partir del número correspondiente al elemento que lo contiene, ya que los cuatro elementos nuevos tienen los números $4\overline{e}+i$ para $i = \overline{1,4}$, donde $\overline{e} = e - 1$ y $e =$ número de elemento de presión. Los elementos se numeran a partir del primer triángulo de la malla gruesa o de presión iniciando con sus cuatro elementos, y el orden en que se numeran es: el primer elemento de velocidad es el que está en el vértice uno del elemento de presión, el siguiente será el que está en el vértice dos, el tercero el que está en el vértice tres del elemento de presión y el número cuatro será el que queda definido por las nuevas aristas. Figura 2.9.

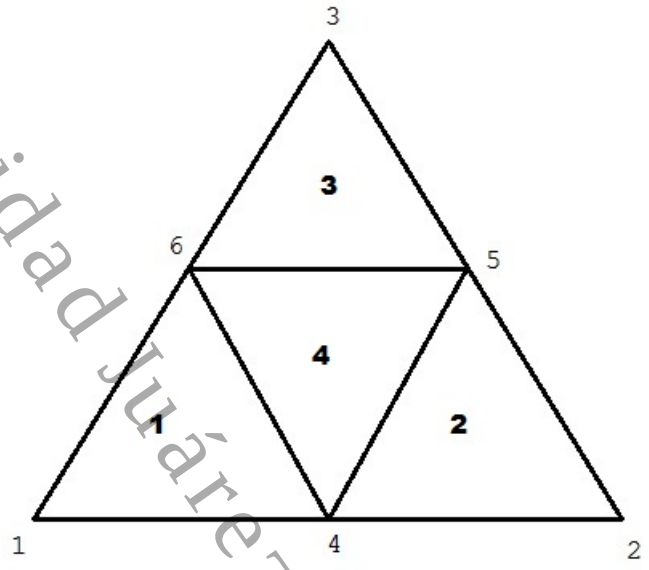
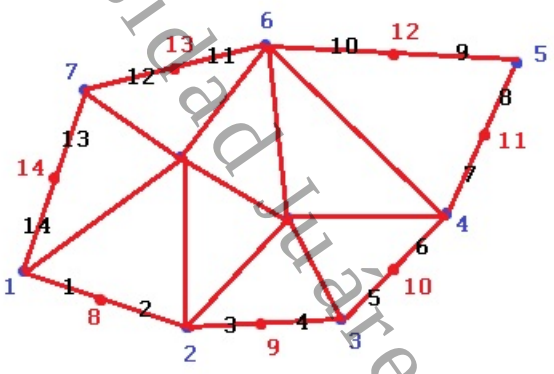


Figura 2.9: El orden de la numeración de los elementos depende de la numeración local de los vértices del elemento de la malla gruesa

2.2.3. Numeración de aristas de frontera

En algunas aplicaciones la numeración de aristas frontera de velocidad son necesarias es por ello que necesitamos determinar su numeración.

La numeración de aristas de frontera es semejante que en la malla gruesa, es decir se numeran continuamente a partir del nodo uno que está unido con el siguiente nodo de frontera en sentido contrario de la manecillas del reloj. En nuestro caso, los nodos frontera se han duplicado pues ya hemos obtenido los nuevos nodos de la malla fina. Figura 2.10.



arista	nodo1	nodo2
1	1	8
2	8	2

Figura 2.10: Los números de color negro nos muestran la numeración de las aristas de la malla fina.

2.3. Graficación del refinamiento de la malla

Para graficar la malla original y la malla refinada utilizamos la subrutina *subgra.m*. Se necesitan tener los arreglos de nodos y elementos, además de tener identificadas las aristas de frontera, tener el arreglo edges donde hallamos todos los datos de las aristas de la malla gruesa, counter que es el número total de aristas y tener identificado a cada elemento mediante sus aristas.

El la subrutina *subgra.m* dibuja en azul la malla gruesa y en rojo la malla fina. Figura 2.11.

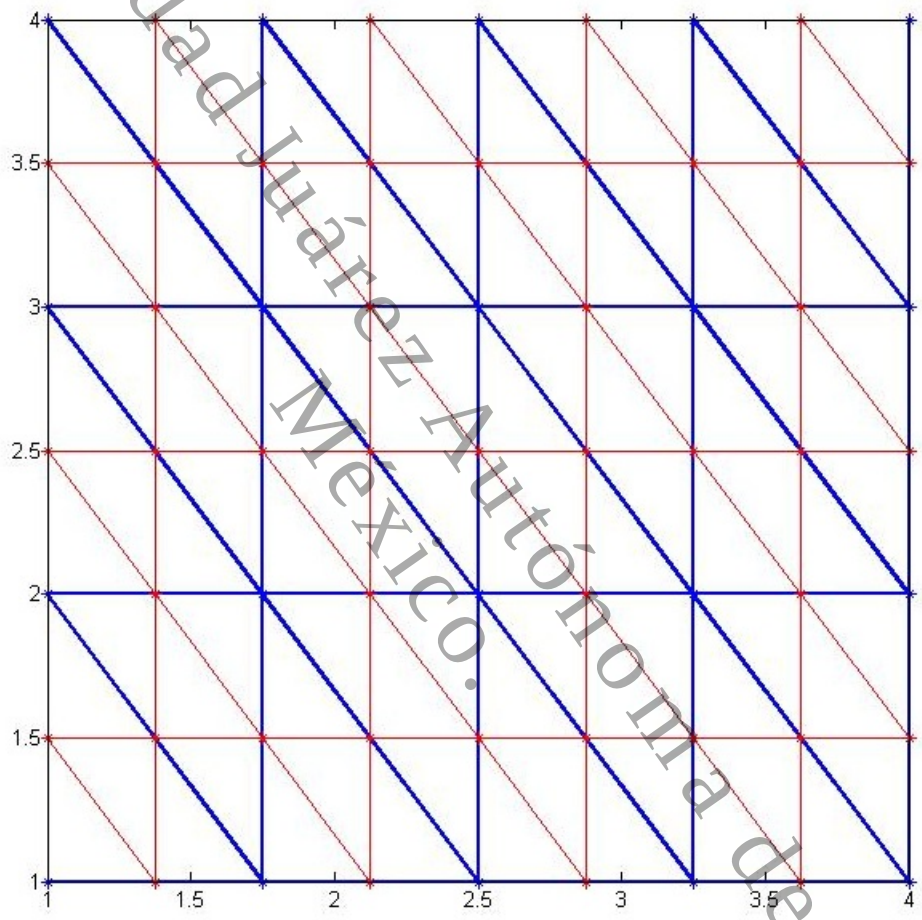


Figura 2.11: La malla fina de color rojo y la malla gruesa es la de color azul.

Capítulo 3
Aplicaciones del mallado triangular y refinamiento

En este capítulo se darán dos aplicaciones del mallado y refinamiento de mallas de dominios en 2D: la convergencia del método de elemento finito y la solución de un problema de ecuaciones diferenciales parciales con mallas mixtas.

En la convergencia del método de elemento finito, se utiliza el refinamiento de una malla para solucionar numéricamente un problema elíptico con condiciones de frontera mixtas. Comenzaremos con una malla triangular que llamaremos malla gruesa, la cuál refinaremos varias veces. Para cada una de estas mallas se aproxima la solución del problema elíptico, comparando el tamaño de la malla con el error de la aproximación.

La segunda aplicación del refinamiento de mallas se utiliza para la solución de un problema de ecuaciones diferenciales parciales con mallas mixtas. El problema considerado es un problema de Tipo Stokes y es un ejemplo de problemas donde es necesario usar dos mallas simultáneamente para aproximar correctamente la solución. Sólo se describirá el problema sin mostrar resultados numéricos, trabajo sustentado en [1].

3.1. Convergencia del método de elemento finito

En las siguientes secciones se describe el procedimiento para usar el método de elemento finito. Se obtendrá la solución numérica de un problema

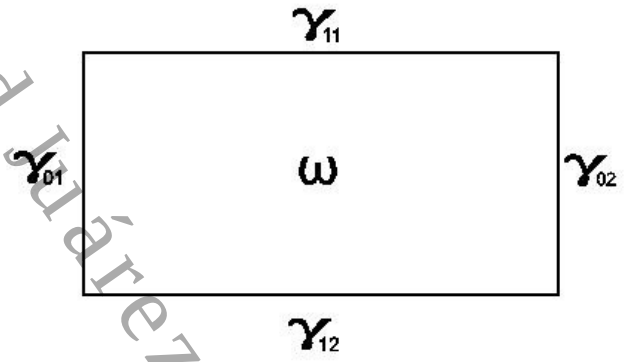


Figura 3.1: Un dominio ω rectangular.

elíptico con condiciones de frontera mixtas de la forma

$$-\nu\triangle u + \alpha u = f \text{ , en } \omega \subset \mathbb{R}^2, \tag{3.1}$$

$$u = g_0 \text{ sobre } \gamma_0, \tag{3.2}$$

$$\frac{\partial u}{\partial n} = g_1 \text{ sobre } \gamma_1, \tag{3.3}$$

donde $\nu > 0$, $\alpha \geq 0$, ω es un rectángulo abierto cuya frontera $\gamma = \partial\omega$ es la unión disjunta de $\gamma_0 = \gamma_{01} \cup \gamma_{02}$ y $\gamma_1 = \gamma_{11} \cup \gamma_{12}$. Ver figura 3.1.

3.1.1. Formulación variacional

Para aplicar el método de elemento finito al problema elíptico (3.1)-(3.3), es necesario primero obtener lo que se denomina la formulación variacional del problema, la cual se obtiene utilizando los siguientes conjuntos definidos como sigue:

$$V = \left\{ v \in H^1(\omega) : v = g_0 \text{ sobre } \gamma_0 \right\}$$

y

$$W = \left\{ v \in H^1(\omega) : v = 0 \text{ sobre } \gamma_0 \right\},$$

En lugar de resolver el problema diferencial descrito por (3.1)-(3.3), resolvemos el problema integral que consiste en encontrar $u \in V$ tal que

$$\nu \int_\omega \nabla u \cdot \nabla v dxdy + \alpha \int_\omega uvdxdy = \int_\omega f\ vdxdy\ + \int_{\gamma_1} g_1\ vd\gamma,\ \forall v \in W \quad (3.4)$$

A (3.4) se le conoce como la formulación débil de (3.1)-(3.3) y una función u que satisface (3.4) se llama una solución débil de (3.1)-(3.3). Obsérvese que se resuelve un problema integral en lugar de un problema diferencial.

Si $f \in L^2(\omega)$, se aplica entonces el teorema de Stampachia (ver [5], pp. 176) con $H = H^1(\omega)$ y se concluye que existe una única solución de (3.4) y que

$$u = \min_{v \in V} \left\{ \frac{1}{2}(\nu \int_\omega |\nabla v|^2 + \alpha \int_\omega v^2 - 2\int_\omega fv - 2\int_{\gamma_1} g_1\ v) \right\}. \square \quad (3.5)$$

A la ecuación (3.5) se le conoce como la formulación variacional de (3.1)-(3.3). Así que existe una única solución débil para el problema (3.1)-(3.3) y se tiene que si f, g_0 y g_1 son suficientemente suaves (continuas, por ejemplo), entonces la solución débil también es la solución clásica.

3.1.2. Espacios de elemento finito

En esta sección nos restringiremos al caso de que ω sea un rectángulo como el descrito en el capítulo uno. Una vez obtenida la formulación variacional para el problema elíptico, para obtener una aproximación por elemento finito es necesario tomar espacios V_h y W_h contenidos en los espacios V y W que intervienen en la formulación variacional. Un espacio V_h se forma de la siguiente manera:

i) Se hace una discretización del dominio ω por medio de triángulos K satisfaciendo (ver figura 3.2)

1) $\overline{\omega} = \cup K$.

2) $\overset{\circ}{K}_i \cap \overset{\circ}{K}_j = \phi$ para $i \neq j$. ($\overset{\circ}{K}$= interior de K).

3) Si K_i y K_j son adyacentes dos de sus vértices deben coincidir, es decir, no se permite que un vértice de K_i sea punto interior de una arista de K_j.

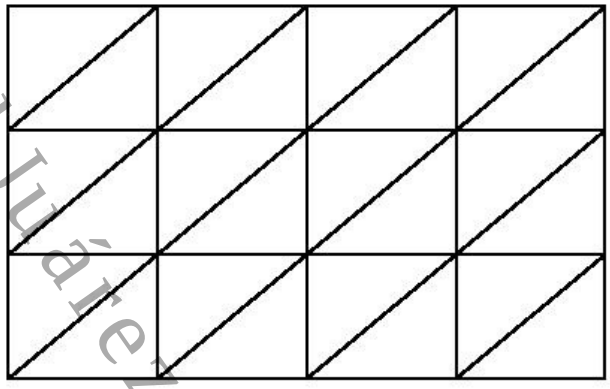


Figura 3.2: Una discretización de ω.

ii) Se define un espacio V_h y los elementos que serán base para tal espacio. Con la triangulación anterior de ω la cual se denota por τ_h se asocian la siguiente definición y los siguientes espacios de dimensión finita:

Definición.-Sea τ_h la colección de triángulos K que forman la partición de ω. Si $K \in \tau_h$ entonces se define a h_K como el diámetro del triángulo K, esto es, la longitud del lado más largo de K. Igualmente se define ρ_k como el diámetro del círculo inscrito en K. Se define

$$h = \max_{K \in \tau_h} h_K.$$

*Definición.*Para $K \in \tau_h$ y $\omega \subset \mathbb{R}^2$ se define
$P_1(K) = \{p : K \to \mathbb{R} \,|p(x,y) = a + bx + cy\}$,
$H_h(\omega) = \{v \in C^o(\omega) : v \,|_K \in P_1(K),\ \forall K \in \tau_h\}$,
$V_h(\omega) = \{v \in H_h(\omega) : u_h = g_0 \text{ sobre } \gamma_0\}$,
$W_h(\omega) = \{v \in H_h(\omega) : v(n_j) = 0,\ \forall n_j \in \gamma_0\}$.

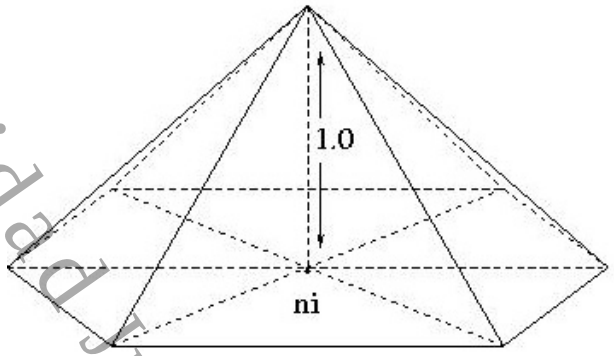


Figura 3.3: Función base local v_i^h.

La base de H_h son las funciones v_i^h que están en H_h y que toman los siguientes valores:

$$v_i^h(n_j) = \delta_{ij}$$

para n_j un vértice o nodo de la triangulación τ_h. Si se etiquetan los vértices de 1 a N, entonces la base de H_h consta de tantos elementos como nodos tenga la triangulación. La base de W_h se obtiene al quitar de la base de H_h las funciones v_i^h asociadas con los nodos que pertenecen a γ_0. Es fácil ver que las v_i^h son linealmente independientes y que cualquier función $u \in H_h$ se puede escribir como una combinación lineal de ellas, esto es, $\left\{v_i^h\right\}_{i=1}^N$ es una base para H_h. En la figura 3.3 se ilustra una función base v_i^h.

Al espacio H_h que se forma con los pasos anteriores se le llama un espacio de elemento finito. Una vez hecho lo anterior se sustituye V_h por V y W_h por W en la formulación variacional (3.4) para obtener el problema discreto asociado:

$$\begin{cases} \text{Encontrar } u_h \in V_h \text{ tal que} \\ \quad a(u_h, v_h) =< f, v_h > \,, \forall v_h \in W_h, \end{cases} \tag{3.6}$$

donde

$$a(u_h, v_h) = \nu \int_\omega \nabla u_h \cdot \nabla v_h + \alpha \int_\omega u_h v_h$$

y

$$< f, v_h >= \int_\omega f v_h + \int_{\gamma_1} g_1 v_h d\gamma.$$

Asumiendo que $\{v_1^h, .., v_n^h, v_{n+1}^h, .., v_N^h\}$ es la base para H_h y $\{v_1^h, .., v_n^h\}$ es la base para W_h se tiene que si $u = (u_1, .., u_N)$ es el vector de aproximación para la solución u:

$$u_j = u_h(n_j),\ j = 1, ..., N.$$

Para n_j el nodo j de la discretización de ω entonces $u_j = g_0(n_j)$ para $j = n+1, ..., N$,

$$u_h = \sum_{j=1}^{N} u_j v_j^h\ ,$$

y que las u_j deben satisfacer

$$\sum_{j=1}^{n} u_j a(v_i^h, v_j^h) =< f, v_i^h > - \sum_{j=n+1}^{N} u_j a(v_i^h, v_j^h)\ ,\ i = 1, ..., n, \tag{3.7}$$

o equivalentemente

$$Au = b, \tag{3.8}$$

donde $a_{ij} = a(v_i^h, v_j^h)$ y $b_i =< f, v_i^h > - \sum_{j=n+1}^{N} u_j a(v_i^h, v_j^h)$. La matriz A es simétrica y definida positiva, por lo cual el vector solución u existe y es único, así que la aproximación u_h a la solución u existe y es única. Para el cálculo de los elementos de la matriz A se hace uso de la propiedad de aditividad de la integral y de la forma en que se discretizó el dominio ω:

$$a_{ij} = \sum_{K \in \tau_h} \int_K (\nu \nabla v_i^h \cdot \nabla v_j^h + \alpha v_i^h v_j^h).$$

3.1.3. Ejemplo Numérico

Se considera el dominio ω en forma de L invertida definido por los puntos $A = (-1/2, 1/2)$, $B = (1/2, 1/2)$ y $C = (0, 0)$, $\alpha = 1$, $\nu = 1$, $g_0 = 0$, $g_1 = 0$ y $f(x, y) = (\alpha + 2\nu\pi^2) sin(\pi y)$. (Ver figura 3.4). Para los datos anteriores se conoce la solución exacta del problema (3.1) -(3.3), la cual es $u(x, y) = sen(\pi x) sen(\pi y)$.

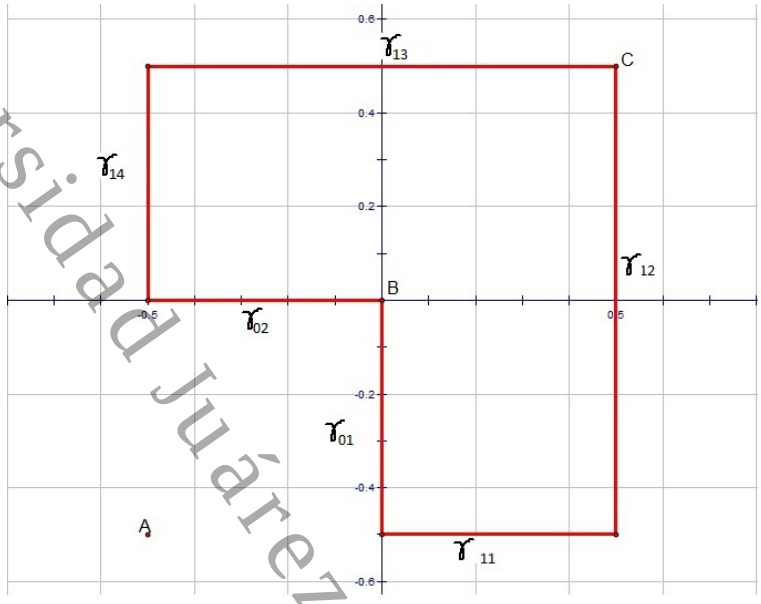


Figura 3.4: Dominio ω definido por los puntos A, B y C.

Para este ejemplo, se obtendrán varias mallas para aproximar la solución al problema usando el método de elemento finito, cada malla debe cumplir con ser más fina que la anterior. En cada caso se debe verificar que el error obtenido en cada refinamiento se aproxime cada vez más a cero, esto es, se debe verificar la convergencia del método de elemento finito.

La relación entre el error y el parámetro de discretización está dado por:

$$e \leq Ch^2, \text{para alguna } C > 0, \tag{3.9}$$

esto es equivalente a la relación

$$ln\ e \leq 2ln\ h + ln\ C, \text{ para alguna } C > 0. \tag{3.10}$$

Siguiendo las ideas descritas en el capítulo 1 en las secciones 1.2 y 1.3, definimos el número de divisiones en la base y altura del dominio, en la figura 3.5 mostramos de manera gráfica la primer malla utilizada con $nx = 5$ y $ny = 5$, donde las bases serán las fronteras γ_{11} y γ_{14}, las alturas serán definidas por las fronteras γ_{01} y γ_{02}; posteriormente se muestran los resultados

al refinarla cuatro veces haciendo que las siguientes mallas tuvieran el doble de divisiones en nx y ny.

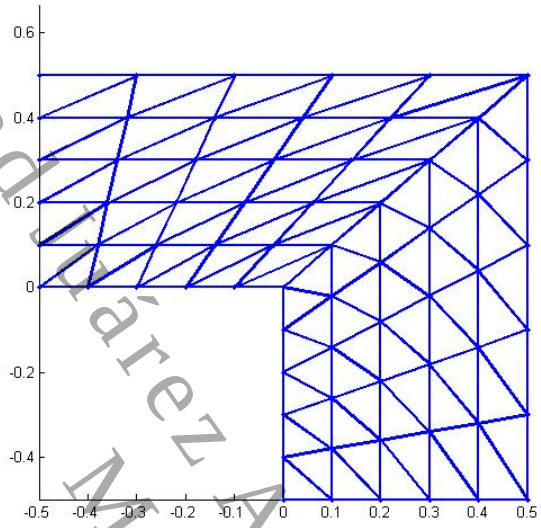


Figura 3.5: Malla triangular para el ejemplo de la L invertida.

La numeración de nodos, elementos y aristas es de la forma que se describió en los capítulos uno y dos. Recordemos que los nodos se dividen en interiores y frontera, donde en nuestras mallas los primeros en numerarse son los nodos frontera, los cuales para este caso se podría pensar que son aquellos que están en la orilladel dominio, sin embargo no es así, solo son los que están sobre γ_1, en la figura 3.6 mostramos cómo se numeran los nodos, no están numerados completamente, el lector puede continuar la numeración continuando como se ha mostrado en el ejemplo hasta llegar al nodo 66.

Para mostrar la convergencia del método de elemento finito hemos obtenido cinco mallas, cada una siempre más fina que la anterior. Se han calculado el valor de la arista más grande de la malla obtenida h, el error y sus respectivos logarítmos naturales. Para calcular el error resolvemos la ecuación (3.11).

$$e = \text{Máx}|U(n_i) - U_i| \tag{3.11}$$

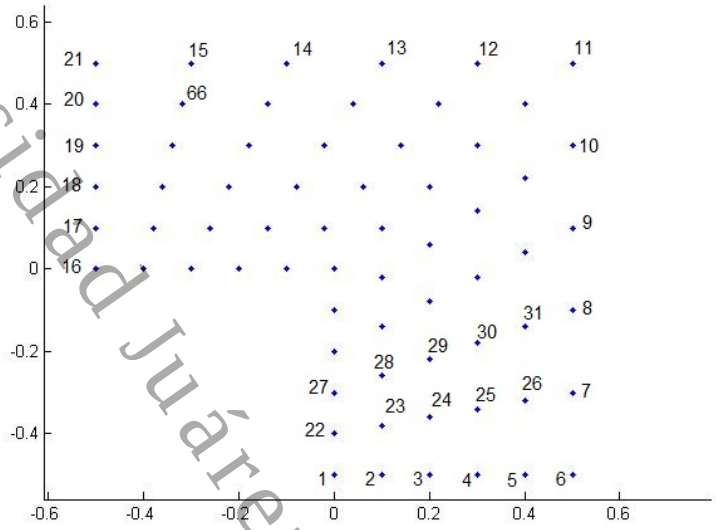


Figura 3.6: Numeración de algunos nodos en el dominio L invertida, el nodo 66 es el último.

Cómo se mencionó anteriormente la malla se refinó cuatro veces y se obtuvieron los resultados de la tabla de la figura 3.7.

nx	h(nx)	Error= e	ln h(nx)	ln e
5	0.3882	1.32E-01	-0.946234608	-2.024953356
10	0.2084	4.91E-02	-1.568295969	-3.013896244
20	0.1079	2.70E-02	-2.226550407	-3.611918413
40	0.0549	1.07E-02	-2.90224193	-4.537511538
80	0.0277	3.70E-03	-3.586322866	-5.599422459

Figura 3.7: Cálculo de las aristas más grandes y el error asociado a éstas para las divisiones nx elegidas, con sus respectivos logarítmos.

Los datos de la tabla, muestran que el error es siempre más pequeño cuando la malla utilizada es más fina que la anterior. Además que al graficar $h(nx)$ contra el error podemos observar siempre se cumple que los puntos $\big(ln(h), ln(e)\big)$ están siempre por debajo de la recta $l = 2\ ln\ h + ln\ C$, es decir, la pendiente de la recta que interpola los puntos: primero y último de los datos obtenidos, es menor que 2. Esto nos dice que nuestros cálculos efectivamente son correctos, el error está cada vez más cerca de ser cero, figura 3.8.

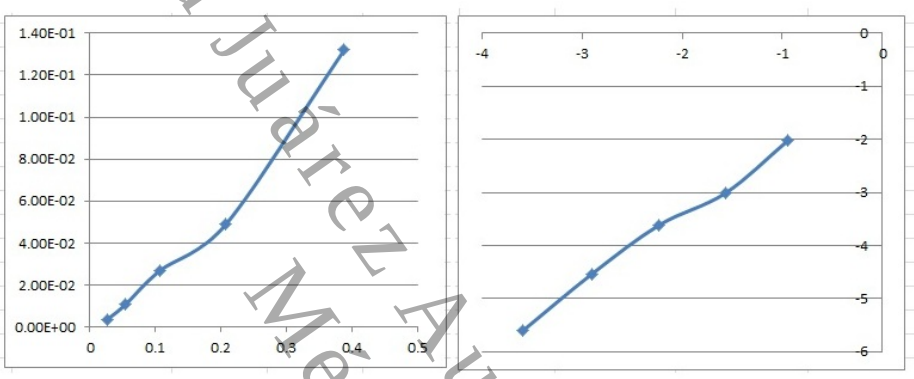


Figura 3.8: Si graficamos los puntos de la tabla, siempre están por debajo de la recta con pendiente $m = 2$.

En la figura 3.9 se muestran las gráficas de las soluciones exacta y aproximada, resultando que ambas son muy similares, sin embargo no son iguales.

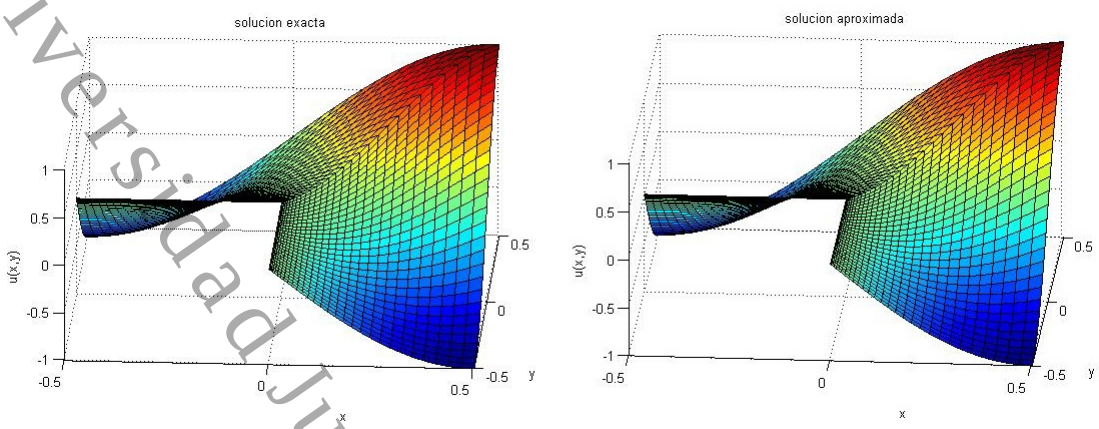


Figura 3.9: La solución aproximada es muy parecida a la exacta en la mayoría de sus puntos.

Si graficamos la diferencia de la solución exacta y la solución aproximada, observamos que en la mayoría de los nodos la diferencia es muy pequeña, sin embargo el error se incrementa a medida que nos acercamos a los nodos que se encuentran cerca del punto (0.5,0.5), figura 3.10.

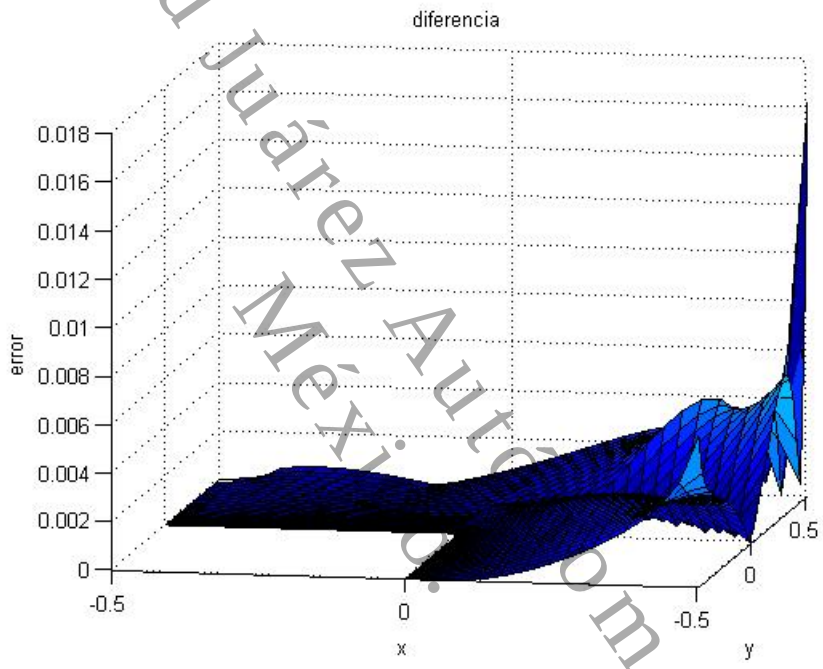


Figura 3.10: Observe que el error aumenta cuando se está cerca del punto (0.5, 0.5).

3.2. Resolviendo un problema tipo Stokes degenerado.

En esta parte describiremos un ejemplo en donde se requiere utilizar simultáneamente el uso de una malla y su refinamiento, tal como lo mencionamos en el capítulo uno, es decir, es un problema de mallas mixtas, donde se deben determinar dos variables dependientes y una de ellas se aproxima en la malla gruesa y la otra en la malla fina.

Usaremos un modelo del tipo de masa consistente para ajuste de campos de velocidades en dos dimensiones, a partir de datos discretos que representan medidas sobre un dominio Ω, esto es, el problema diferencial dado un campo inicial $\mathbf{u}^I : \Omega \longrightarrow \mathbb{R}^2$ se debe encontrar un campo ajustado $\mathbf{u} : \Omega \longrightarrow \mathbb{R}^2$ tan cercano como sea posible al inicial y $\lambda : \Omega \longrightarrow \mathbb{R}$ tales que:

$$\begin{aligned} S(\mathbf{u} - \mathbf{u}^I) &= \nabla\lambda \text{ en } \Omega \ , \\ \nabla\mathbf{u} &= 0 \text{ en } \Omega \ , \\ \mathbf{u} \cdot \overrightarrow{n} &= 0 \text{ en } \Gamma_N. \end{aligned} \tag{3.12}$$

El conjunto Ω se considera una región en $\mathbb{R}^2$ con frontera $\partial\Omega$, la cual está dada por $\partial\Omega = \Gamma_N \bigcup \Gamma_V \bigcup \Gamma_D$, en el caso de campos de viento, y para el caso de agua $\partial\Omega = \Gamma_N \bigcup \Gamma_V$, donde Γ_N es la frontera natural, y Γ_V, Γ_D es el resto de la frontera la cual es una frontera artificial, como se muestra en la figura (3.11).

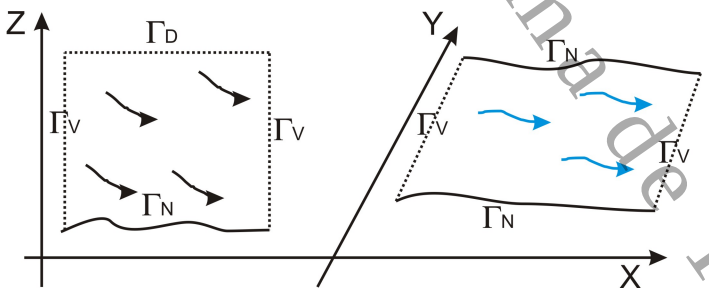


Figura 3.11: Regiones 2D para campos de velocidad

Resolver el problema diferencial descrito por (3.12) es equivalente a resolver el problema integral: dado $\mathbf{u}^I : \Omega \longrightarrow \mathbb{R}^2$ se debe encontrar $(\mathbf{u}, \lambda)$ tal que:

$$\int_\Omega S\mathbf{u} \cdot \mathbf{v} dx + \int_\Omega \lambda \nabla \cdot \mathbf{v} dx = \int_\Omega S\mathbf{u}^I \cdot \mathbf{v} dx, \quad \forall\ \mathbf{v} \in \mathbf{V_N}. \tag{3.13}$$

Para aproximar la solución $(\mathbf{u}, \lambda)$ del problema(3.13), donde $\mathbf{u} : \mathbb{R}^2 \to \mathbb{R}^2$, pero $\lambda : \mathbb{R}^2 \to \mathbb{R}$, se puede usar el método de gradiente conjugado (como se puede ver en [1]), lo cual implica resolver por el método de elemento finito problemas con la misma estructura que (3.13), pero para valores de λ conocidos, lo cual se sabe resolver por ser problemas de tipo elíptico. Para la aproximación numérica de este tipo de problemas se utilizan mallas mixtas, donde λ se aproxima en la malla gruesa, mientras $\mathbf{u}$ se aproxima en la malla fina.

Capítulo 4
Conclusiones

En esta tesis se abordó el problema de mallar con triángulos una región irregular plana. Se trabajó un algoritmo para hacer el mallado, partiendo de un algoritmo para mallar una región rectangular. Con el fin de poder usar estas mallas computacionalmente, el mallado se resume en numerar los nodos y triángulos de la malla y construir dos arreglos, uno donde se guarden las coordenadas de los nodos y otro donde se establezca qué nodos forman cada triángulo. Para esto es necesario establecer también una numeración local de los vértices de cada triángulo.

Asociado con el problema de mallado con triángulos de una región, está el problema de refinar una malla, el cual también se consideró en esta tesis, y que consiste en que dada una malla, obtener otra más fina dividiendo cada triángulo en cuatro subtriángulos. La nueva malla obtenida se conoce como un refinamiento de la original. Una característica que se le ha pedido a este refinamiento es que considere la relación entre triángulos originales y nuevos, es decir dado un triángulo original saber qué subtriángulos contiene y a la inversa, dado un triángulo de la malla fina saber a qué triángulo original pertenece. Esto es necesario para algunas aplicaciones donde se deben resolver problemas que requieren usar simultáneamente una malla y su refinamiento.

En este trabajo se muestran dos aplicaciones del uso de mallas y sus refinamientos. En la primera aplicación se verifica la convergencia del método de elemento finito en la solución de un problema elíptico particular. En este caso es necesario resolver varias veces el mismo problema elíptico pero en diferentes mallas, siendo estas refinamientos sucesivos de una malla original. Aquí cada vez que se resuelve el problema elíptico se utiliza solo una malla.

La segunda aplicación consiste en mostrar un problema tipo Stokes degenerado donde es necesario utilizar mallas mixtas, es decir, simultńeamente una malla y un refinamiento ya que en el problema intervienen dos funciones desconocidas que podemos interpretar como velocidad y presión y para que el algoritmo numérico usado sea estable, la presión debe aproximarse en la malla original mientras que la velocidad debe aproximarse en la malla refinada.

Bibliografía

[1] Arias, Palacios R.; Tesis de Maestría: *Ajuste de campos de velocidad vía gradiente conjugado.* UJAT 2012.

[2] Bello Galeana, C.E.; (2006) *Mallas Triangulares.*

[3] Delgado J., Juárez Valencia L.H., Saavedra P., Sandoval M.L.; (2010) *First Symposium on Inverse Problems and its Aplicca-tions.* Ixtapa, México.

[4] Díaz M.A.; (2000) *Métodos de mallado y algoritmos adaptativos en dos y tres dimensiones para la resolución de problemas electromágneticos cerrados mediante el método de los elementos finitos.* PhD thesis, Universidad Politécnica de Valencia, España.

[5] H. Bresiz, *Análisis funcional. Teoria y aplicaciones*, Alianza Editorial, 1978.

[6] Juárez Valencia L. H.; *Apuntes para la subdivisión de mallas bidimensionales.* Universidad Autónoma Metropolitana. México, D.F.

[7] Juárez Valencia L.H.; *Numerical simulation of a viscous pump for MEMS applications.* Preliminary Report, UAM-Iztapalapa. México, D.F.

[8] *MATLAB, The Language of Technical Computing.* Getting Started with MATLAB.

[9] Rivera Loaiza, C.; (2000) *Utilización de Redes Petri para la Elaboración de una Interfaz de Usuario.* PhD thesis. Morelia,México.

[10] Rodríguez P. A.; (2011) *Mallas Geométricas, sus aplicaciones y la Paralelización de Algoritmos de Refinamiento de Mallas.*

[11] Suárez Rivero J.P.; *Sobre Algoritmos de Refinamiento de Mallas y Estructuras de Datos Relacionados.* Dpto. Matemática Aplicada, Universidad de las Palmas de Gran Canaria España, España.

[12] Tinoco-Ruiz, J. G.; (1997) *Funcionales discretos para la generación de mallas suaves y convexas sobre regiones planas irregulares.*. PhD thesis, CIMAT, Guanajuato, Gto.

[13] Torres, C.L y Wilgenhoff, Cristhian P.; (2009) *Trabajo final de anaálisis y diseño de algoritmos Generador de Mallas Triangulares.* Argentina.