UNIVERSIDAD NACIONAL DE CÓRDOBA
FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES

Trabajo Final de Grado

Ingeniería Aeronáutica

Desarrollo y Evaluación de Algoritmos para Combinar Mallas de Elementos Finitos con Grillas del Método de Red de Vórtices Inestacionario con Topología Arbitraria

Pérez Segura, Martín Eduardo

2014

UNIVERSIDAD NACIONAL DE CÓRDOBA
FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES

Trabajo Final de Grado

Ingeniería Aeronáutica

Desarrollo y Evaluación de Algoritmos para Combinar Mallas de Elementos Finitos con Grillas del Método de Red de Vórtices Inestacionario con Topología Arbitraria

Pérez Segura, Martín Eduardo

Dirección: Dr. Ing. Sergio Preidikman
Co-Dirección: Mag. Ing. Mauro S. Maza

2014

CONTENIDOS

Capítulo I: Notas Introductorias

I.1. Contexto General

La aeroelasticidad es una rama de las ciencias de ingeniería que ha cobrado preponderancia a lo largo del siglo XX en conjunción con el desarrollo de la industria aeroespacial. Conceptualmente representa la mixtura entre tres campos de la física que se relacionan activamente: La Dinámica, La Elasticidad y La Aerodinámica. Su principal objeto de estudio concierne a la interacción entre un cuerpo, o estructura elástica, inmerso en una corriente fluida y las acciones aerodinámicas resultantes, pasando por el análisis de deformaciones, alteraciones del campo de movimiento y efectos inerciales.

Debido a su naturaleza intrincada, la aplicación práctica de los conceptos aeroelásticos ha sido relegada casi exclusivamente al tratamiento computacional, al igual que otros campos del análisis estructural y la aerodinámica. De este proceso surge una nueva área de estudio denominada sucintamente "Aeroelasticidad Computacional" o "CAE" (de aquí en adelante por sus siglas en inglés). Una tarea de gran importancia dentro del campo de la aeroelasticidad computacional (CAE) es la simulación numérica de problemas de interacción fluido-estructura o "FSI" (por sus siglas en inglés). La principal dificultad en este tipo de problemas radica en que las acciones aerodinámicas sobre un cuerpo flexible inmerso en un fluido, con movimiento relativo entre ellos, dependen de la forma, velocidad, y aceleración del cuerpo, mientras que estas tres, a su vez, dependen de las cargas aerodinámicas que el fluido ejerce sobre el cuerpo flexible.

Lógicamente la concurrencia de estos dos fenómenos de naturaleza disímil, la dinámica del cuerpo y las acciones aerodinámicas, debe ser representada de forma fehaciente para que los procesos de simulación también lo sean. En este sentido y de manera general, la respuesta de la estructura a las cargas aerodinámicas se calcula utilizando el método de elementos finitos (FEM), mientras que para las cargas aerodinámicas sobre la estructura se utilizan técnicas de la dinámica de los fluidos computacional (CFD). Para atacar el problema de FSI el dominio es discretizado, emergiendo dos mallas o grillas. La primera de ellas es una malla de elementos finitos que surge sobre la estructura, denominada malla estructural (ME). En segundo lugar y cuando la técnica utilizada para calcular las cargas aerodinámicas es el Método de Red

de Vórtices (UVLM), sólo es necesario discretizar la interface o frontera definida por el contorno del cuerpo dentro del dominio fluido dando origen así a una nueva malla denominada grilla aerodinámica (GA).

En general las grillas provenientes del UVLM (GA) y las mallas provenientes del FEM (ME) tienen topologías muy diferentes. La topología de las GA depende de consideraciones puramente aerodinámicas y es independiente de la discretización de la estructura, generalmente se compone de elementos cuadriláteros, con un nodo en cada vértice, y un nodo adicional, denominado punto de control, en el centro geométrico de cada elemento. Mientras que, una malla de elementos finitos, en general, obedece únicamente a consideraciones de la dinámica estructural.

Durante el paso de información entre la malla estructural y su contraparte aerodinámica es necesario que se conserven las cargas totales y la energía total. Además, esta interacción usualmente involucra dos tópicos principales:

- la transferencia de desplazamientos, velocidades, y aceleraciones desde los nodos de la ME hacia los puntos nodales de la GA; y
- la transferencia de fuerzas/presiones desde los llamados puntos de control de la grilla del UVLM hacia los nodos de la malla del FEM.

Por lo tanto, esta transferencia de datos entres grillas y mallas es de capital importancia en el campo de la aeroelasticidad computacional, donde los métodos de interpolación entre las mismas pueden fácilmente transformarse en el factor que controla la precisión de la simulación aeroelástica.

Un campo relativamente reciente de aplicación de la CAE, en el cual la interrelación entre GA y ME ha mostrado ser sumamente preponderante, es el análisis de alas batientes. Esta área de estudio está inspirada en la biología por cuanto trata el complejo movimiento que se produce durante de vuelo de diversas criaturas (insectos principalmente) con el fin de utilizar sus resultados en el diseño de dispositivos de utilidad práctica. En este caso, la malla estructural del FEM procura representar las alas de estos animales consideradas como estructuras elásticas y la grilla aerodinámica simula el flujo de aire alrededor de las mismas, dando lugar a un problema de FSI sumamente complejo que deberá considerar el batimiento y todos los fenómenos asociados a éste.

Tanto como para el análisis de alas batientes como para cualquier otro caso de FSI, la interacción entre mallas y grillas constituye un punto clave. Luego, es precisamente aquí donde encuentra sitio el presente trabajo que busca contribuir con medios computacionales para hacer más eficaz y eficiente el pre-procesamiento de datos que requiere dicha interacción.

I.2. OBJETIVO

La transferencia de datos y parámetros entre mallas estructurales y grillas aerodinámicas es un proceso que se repite constantemente a medida que avanzan los pasos de cálculo de la resolución de un problema de FSI. Esta interacción puede encararse de diversos modos que dependerán del método de resolución que se

aplique. No obstante y sin importar cuál sea el procedimiento utilizado, es imprescindible generar un vínculo entre nodos y/o puntos de control, y elementos, según sea el caso, antes de proceder al intercambio de información.

A modo de ejemplo, supóngase un punto de la grilla aerodinámica que posee un valor de presión que debe transferirse a la estructura como una carga aplicada. Dado que la estructura se encuentra discretizada con una malla de FEM, la carga aerodinámica deberá aplicarse en uno de sus elementos, el cual deberá ser identificado. Ahora bien, si se tiene en cuenta que una GA cuenta con algunos decenas de miles de nodos y una ME posee una cantidad similar de elementos que se encuentran en constante movimiento y deformación, se pone de manifiesto la importancia de la eficiencia en el desarrollo de este proceso. A modo aclaratorio, en la figura siguiente se esquematiza la interrelación entre un punto de la grilla aerodinámica y un elemento de la malla estructural.

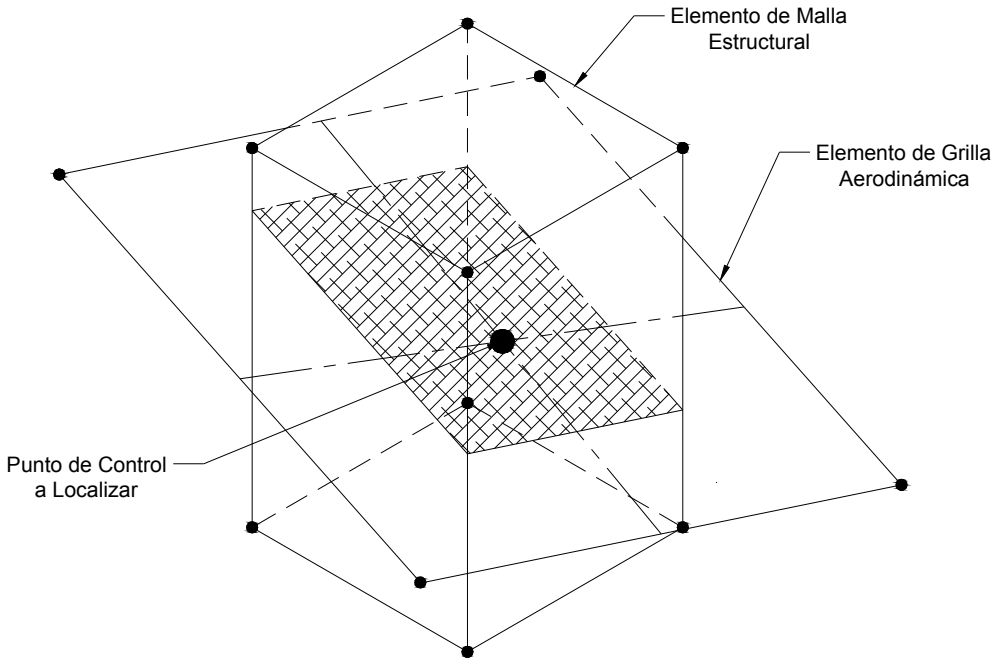


Figura I.1: Esquema de interacción entre un punto de la GA y un elemento de la ME

Habiendo explicado lo anterior, puede decirse de forma general que el objetivo de este trabajo final es desarrollar e implementar algoritmos eficientes y robustos que permitan transferir datos entre grillas aerodinámicas y mallas estructurales, ambas, con topología arbitraria e independiente; y, de forma particular, consiste en la localización topológica de los nodos y puntos de control de la malla de red de vórtices dentro de los elementos estructurales provenientes de la aplicación del Método de Elementos Finitos. Todo esto orientado a la aplicación práctica del análisis de alas batientes.

Para alcanzar el objetivo, debe tenerse en cuenta que la implementación de los algoritmos desarrollados se realiza dentro de una herramienta computacional. La cual

comprende un código de programa denominado, de aquí en adelante Programa de Vinculación, cuyo propósito es ejecutar las tareas para las que los algoritmos fueron diseñados. Además, es necesario incluir sistemas de medición de performance para la evaluación y comparación de los distintos algoritmos a fines de conocer su eficiencia propia y comparativa.

I.3. ORGANIZACIÓN

Teniendo presente el contexto en el que se sitúa el presente trabajo y estando su objetivo definido en las líneas precedentes, se da paso a exponer de manera resumida la disposición y la relación que existe entre los temas que se tratarán a continuación.

Luego de este breve título introductorio necesario para ambientar al lector en la materia, se da lugar a un capítulo netamente de contenidos teóricos que brindan nociones preliminares al desarrollo subsiguiente. Inicialmente, el Capítulo II, trata las generalidades del Método de los Elementos Finitos: se reseña su origen, se mencionan sus aplicaciones, y se establece su relación con los Problemas de Valores en la Frontera y los métodos de aproximación. Seguidamente, el tópico se traslada a los elementos y sus características. Se describen las funciones de forma como el parámetro principal que define a un elemento, para luego entrar en el método del Dominio de Origen. Respecto a éste se presentan dos casos, quizás los más generalmente utilizados, el Cuadrilátero Bilineal y el Hexaedro Trilineal, para elementos planos y espaciales respectivamente. Adicionalmente, se trata el método de degeneración para construir elementos alternativos y el modo sistemático de utilizar todos los conceptos anteriores para elementos de orden superior (no lineales). Por último, se expone el concepto de elemento isoparamétrico y las ventajas que su utilización implica.

Establecidas las nociones teóricas generales, se da paso al análisis del Programa de Vinculación en sí. El Capítulo III contiene una descripción conceptual del código que conforma al Programa de Vinculación, presentando sus funciones principales y división interna. Respecto a esta última, se propone una desmembración del código en tres bloques o segmentos bien definidos, los módulos. Estos módulos poseen objetivos parciales específicos y diferenciados pero se acoplan durante la ejecución para obtener finalmente la Matriz de Localización. En este cuarto capítulo se tratan también, el contexto de implementación del software, las características del mallador y el modo en que éste proporciona los datos de las mallas. Seguidamente se presenta cada módulo por separado para concluir, finalmente, con una reseña sobre su acoplamiento.

Luego de la recorrida conceptual por los módulos del Programa de Vinculación, los capítulos IV, V y VI tratan distintas soluciones aplicables a cada uno de ellos y sus implementaciones. El Capítulo IV, refiere al primero de los módulos, el Módulo de Ingreso, y expone algunos de sus conceptos más importantes referidos al procesamiento de mallas. Muestra la importancia del Ordenamiento Nodal, define y describe la confección de Matrices de Procesamiento de Datos y la utilización de

Bibliotecas de Parámetros. En todos los casos se introducen los mecanismos de implementación describiendo textual y esquemáticamente los procesos, concluyendo con una sección referida al funcionamiento general del módulo.

El siguiente capítulo comprende el Módulo de Selección y describe seis algoritmos diferentes aplicables a éste. Inicialmente, el Capítulo V expone el objetivo general del segundo módulo para luego dar paso a las descripciones de los métodos implementados. El primero de los sistemas es el Algoritmo de Lista, que cuenta con tres variantes que se presentan comparativamente. Luego, se detalla el Algoritmo de Centros de Masa que hace uso de una propiedad físico-geométrica para realizar la selección. Otro método lo constituye el Algoritmo ESUP que aporta el sistema de almacenamiento dinámico conocido como Listas Vinculadas. Por último, se describe el desarrollo de un procedimiento combinado entre los dos anteriores, denominado Algoritmo ESUP Ordenado por Centros de Masa. En el cual se pone de manifiesto la fuerte influencia del orden del conjunto de elementos seleccionados.

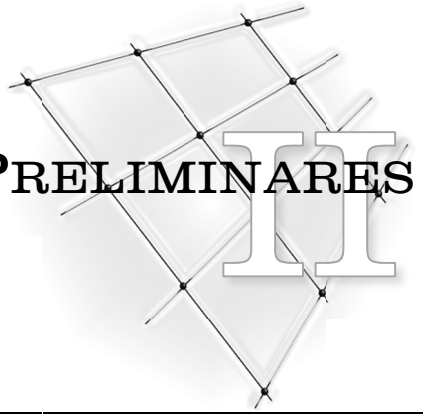
El último capítulo sobre algoritmos e implementación, el Capítulo VI, está destinado al Módulo de Localización. Luego de precisar los objetivos del mismo, se proponen dos sistemas, uno de ellos constituido por un proceso iterativo de búsqueda que se basa en el clásico método numérico de Newton-Raphson. En adición, se plantea una posible mejora denominada Reducción de Elementos, pretendiendo optimizar el algoritmo de localización. El segundo sistema utiliza el concepto de representación por coordenadas de área para desarrollar a partir de éste un algoritmo de evaluación. Aquí se integran conceptos del álgebra y la geometría analítica en el desarrollo de un método con un enfoque particular.

Habiendo detallado las posibles soluciones a implementar para la confección del Programa de Vinculación, se da paso a las evaluaciones y mediciones de desempeño. El Capítulo VII, engloba todo lo referente a sistemas de testeo de algoritmos, toma de datos e interpretación de valores. A continuación de establecer algunos conceptos generales, se clasifican las pruebas a realizar según si se aplican a un módulo en particular o al código general como conjunto. El apartado siguiente comprende la implementación de mecanismos para la medición de desempeño dentro del código, haciendo referencia a las herramientas utilizadas a tal fin. Como es de esperarse siempre que sea necesario efectuar una prueba debe generarse un contexto controlable en el que cada uno de los algoritmos pueda ser evaluado, es por esto que también se destina una sección a la descripción de las mallas de prueba que se emplearán en los testeos. Luego de los desarrollos anteriores se arriba al segmento más representativo del capítulo: la exposición e interpretación de resultados. Aquí se muestran los valores obtenidos en las distintas etapas de medición de performance y se dan conclusiones sobre los mismos.

El capítulo final de este trabajo se dedica al análisis reflexivo del material. El Capítulo VIII se desdobla entre tres tópicos principales con una marcada influencia de lo expuesto en las primeras secciones de éstas notas introductorias. Observaciones sobre el grado de cumplimiento del objetivo planteado, recomendaciones basadas en las experiencias intrínsecas del desarrollo del trabajo y propuestas de mejoras y opciones para trabajos futuros, se incluyen como comentarios finales.

Luego de los ocho capítulos de desarrollo de material, se adicionan cuatro anexos que complementan los temas tratados, tres en la versión impresa y uno en soporte digital (CD-ROM). El primero de ellos presenta las expresiones formales de las funciones de forma para los elementos de mayor difusión. El segundo, proporciona esquemas sobre los sistemas de ordenamiento nodal y su vinculación. Finalmente, el tercero, da un listado de los códigos de programas desarrollados. En el anexo digital se encuentra un archivo (en formato ".pdf") con el presente trabajo, junto con una carpeta, identificada como "Códigos", donde pueden encontrarse todos los algoritmos desarrollados en concordancia con el listado del Anexo III.

Capítulo II: Nociones Preliminares



II.1. El método de los elementos finitos

La observación y explicación de fenómenos naturales a través del método científico, que acompañó el desarrollo del hombre a lo largo de la historia ha sufrido indudablemente un cambio de paradigma en la segunda mitad del siglo XX. A comienzos de la década de 1960, el surgimiento de la computación como una herramienta de uso sistemático y efectivo permitió la recuperación e implementación de los métodos de análisis numérico surgidos en el pasado. Estos métodos resultaron vastamente útiles en las distintas ramas de las ciencias y, en especial, en la ingeniería.

Resulta dificultoso precisar el momento de la invención o surgimiento del Método de los Elementos Finitos (o "FEM", del inglés Finite Element Method). Históricamente tiene sus raíces en la necesidad de solucionar problemas de elasticidad y análisis estructural para estructuras livianas en el campo aeroespacial. Sin embargo, a través de los años y como consecuencia de su perfeccionamiento se hizo extensivo a casi todas las ramas de la ingeniería.

En términos generales, el FEM es un procedimiento para aproximar numéricamente la solución de Problemas de Valores en la Frontera (o "BVP", por sus siglas en inglés Boundary Value Problem), cuyas soluciones exactas no pueden obtenerse por los métodos analíticos clásicos, a excepción de un puñado de casos simples cuya aplicación se encuentra significativamente acotada. Cabe mencionar que una "solución numérica" constituye una aproximación al valor exacto que se obtendría con un tratamiento analítico. No obstante, la correcta implementación de métodos numéricos hará de dicha aproximación un valor plausible para aplicaciones prácticas.

Las aplicaciones y variantes del FEM son numerosas y exceden el objetivo de este trabajo, por lo que sólo es menester mencionar un principio que tienen en común muchos de ellos: la discretización. La discretización es el proceso por el cual se particiona el dominio continuo en un conjunto discreto de sub-dominios, comúnmente llamados elementos.

II.2. PROBLEMA DE VALORES EN LA FRONTERA Y EL FEM

Un BVP está compuesto por una ecuación diferencial en derivadas parciales, o un sistema de ellas, cuya solución cumple con determinadas condiciones propuestas sobre los límites del dominio sobre el que se plantea, denominadas "condiciones de borde". En general existen diversos tipos de ecuaciones diferenciales y de condiciones a las cuales puede exigirse se sometan. Ésta diversidad da lugar a meticulosas clasificaciones que tienen en cuenta la morfología del álgebra involucrada, mas no es necesario ahondar en estos conceptos aquí puesto que son prescindibles a la hora de vincular los BVP con el FEM.

De todos los casos que revisten interés práctico en las diversas áreas de la ingeniería, sólo algunos pocos tienen la particularidad de ser discretos por naturaleza. Los restantes, que representan la gran mayoría, poseen un carácter continuo que dificulta su tratamiento. Como se mencionó, la resolución de las ecuaciones diferenciales que aplican a estos casos es costosa y muchas veces impracticable. Por lo tanto, la implementación del FEM como método de cálculo constituye la alternativa al complejo tratamiento de sistemas continuos, y para ello son necesarios los Métodos de Aproximación.

II.2.1. EL ENFOQUE DE LA APROXIMACIÓN

Existen diversos Métodos de Aproximación que pueden aplicarse a una formulación de BVP, pero en todos ellos es imprescindible discretizar el dominio, lo cual se consigue a través de la generación de una "malla de elementos finitos". Además, la discretización comprende una etapa de pre-procesamiento en la que se pretenden dominios sencillos sobre los cuales sean válidas soluciones simples, de tipo algebraicas, para luego ensamblarlas en la solución general buscada.

La malla de elementos finitos está compuesta por subdominios definidos a partir de un conjunto de nodos conectados entre sí por curvas que representan su frontera. Como premisa general, una malla deberá no sólo respetar la geometría del dominio, sino también ser adecuada para las características del mismo. De ahí que se requiere un análisis profundo y una interpretación adecuada del problema para generar una malla de elementos.

Por otro lado, la transformación un procedimiento de aproximación, como ser el método de Galerkin (ver [1]), en una aplicación del FEM se logra a través de la selección de Funciones de Forma que cumplan determinadas propiedades, que serán definidas más adelante. A este respecto pueden diferenciarse dos puntos de vista: uno global, que considera todo el dominio del BVP, en el cual las Funciones de Forma, N_A, están definidas; y uno local o del elemento, que se centra en cada subdominio definiendo sus propiedades, incluidas las Funciones de Forma.

En otras palabras, el punto de vista global define parámetros referidos a una función de interpolación u^h que, estando definida globalmente, se restringe al

dominio del elemento. Tomando como ejemplo un dominio lineal para la definición del BVP, es decir, $\overline{\Omega} = x \in [0,1]$, se tendrá para cada elemento los siguientes parámetros globales:

- Dominio: $[\mathrm{x_A}, \mathrm{x_{A+1}}] \in [0,1]$;
- Nodos: $\{\mathrm{x_A}, \mathrm{x_{A+1}}\}$;
- Grados de Libertad: $\{\mathrm{d_A}, \mathrm{d_{A+1}}\}$;
- Funciones de Forma: $\{\mathrm{N_A}, \mathrm{N_{A+1}}\}$;
- Función de Interpolación: $\mathrm{u^h} = \mathrm{d_A N_A(x)} + \mathrm{d_{A+1} N_{n+1}(x)}$

Sin embargo, es necesario definir los parámetros locales de cada elemento ya que son éstos los que permiten la estandarización de los cálculos para la implementación computacional. Se tiene entonces para el punto de vista local:

- Dominio: $[\xi_1, \xi_2]$;
- Nodos: $\{\xi_1, \xi_2\}$;
- Grados de Libertad: $\{\mathrm{d_1}, \mathrm{d_2}\}$;
- Funciones de Forma: $\{\mathrm{N_1}, \mathrm{N_2}\}$;
- Función de Interpolación: $\mathrm{u^h}(\xi) = \mathrm{d_1 N_1}(\xi) + \mathrm{d_2 N_2}(\xi)$

La relación entre los parámetros globales y locales se realiza a través de una transformación afín, entendida como una función que vincula los dominios involucrados, es decir,

$$\xi(x): [x_A, x_{A+1}] \rightarrow [\xi_1, \xi_2]. \qquad \text{(II.2.1.1)}$$

Donde $\xi(x)$ es un mapeo y x es un punto e, inversamente,

$$x(\xi): [\xi_1, \xi_2] \rightarrow [x_A, x_{A+1}]. \qquad \text{(II.2.1.2)}$$

Donde $x(\xi)$ es un mapeo y ξ es un punto. En ambos casos deberá verificarse la coincidencia entre nodos, esto es

$$\xi(x_A) = \xi_1, \qquad \xi(x_{A+1}) = \xi_2 \qquad \text{(II.2.1.3)}$$

$$x(\xi_1) = x_A, \qquad x(\xi_2) = x_{A+1}. \qquad \text{(II.2.1.4)}$$

Con los conceptos anteriores queda establecida de manera somera la relación entre los BVP y el FEM, a través de los métodos de aproximación. Como conclusión debe destacarse que en cada caso particular con el que se trate, existirán conjuntos de parámetros locales y globales con propiedades dadas por la naturaleza del BVP en cuestión. No obstante, el enfoque local permite ahondar en el análisis de elementos y funciones de forma por sí mismos, e independientemente de la aplicación práctica que pueda dárseles en el FEM. De este modo, una vez definidas globalmente las propiedades matemáticas del problema se podrá recurrir a un extenso arsenal de elementos de distinta naturaleza para la implementación del FEM, cuya selección deberá ser gobernada por un criterio apropiado que evalúe la precisión y eficiencia de los procesos de cálculo.

II.3. ELEMENTOS

Como se señaló oportunamente, el enfoque local de los elementos permite realizar un análisis sobre éstos independizándolos del caso práctico del que provengan o en el cual se pretendan utilizar. Así, será posible profundizar sobre ciertos conceptos y características de los elementos que serán imprescindibles para acometer al objetivo del presente trabajo.

A modo de definición, un elemento puede entenderse como una porción reducida de un continuo. En consecuencia, la geometría del continuo, que en definitiva representa el dominio sobre el cual se planteará el problema, será considerada como el ensamble de pequeños subdominios de geometría simple que no se intersecan: los elementos. De ahí se expresa que una malla de elementos discretiza al continuo. Por otro lado, el vocablo "finito" merece especial mención puesto que distingue la teoría aplicada en el FEM de aquella relacionada al cálculo diferencial, luego, debe señalarse que un elemento finito nada tiene que ver con un elemento infinitesimal.

De acuerdo a las características topológicas del continuo que se desea discretizar podrán utilizarse elementos como cuadriláteros o triángulos en dos dimensiones, o bien, elementos como hexaedros, tetraedros o pirámides en tres dimensiones. También es posible encontrar elementos que no poseen aristas o bordes rectos, sino que se constituyan por curvas polinómicas de segundo o tercer orden. En este sentido, el dominio de un elemento queda determinado por los nodos que lo conforman, es decir, por los puntos de coordenadas conocidas que definen sus bordes o aristas. Finalmente, la variación espacial de los parámetros del problema se expresa dentro de cada elemento como una expansión polinómica, generalmente en términos de las coordenadas de sus nodos, que representa una aproximación a la variación analítica exacta.

II.4. FUNCIONES DE FORMA

Las funciones de forma son quizás el parámetro más representativo de los elementos, naturalmente luego de sus nodos y geometría que son indispensables para definirlos. Es deseable definir las funciones de forma N_A, de manera tal que, a medida que se refine la malla de elementos, el método de aproximación converja a la solución exacta. En referencia, existen tres condiciones suficientes para que este fenómeno ocurra, las cuales serán mencionadas aquí sin mayor profundidad puesto que exceden el enfoque del trabajo, a saber:

- Las funciones deben ser suaves en el interior del elemento (Ω): no se acepta la existencia de discontinuidades y debe existir al menos una derivada única en todo punto.
- Las funciones deben ser continuas a través de los límites de los elementos (Γ): esto garantiza, en conjunción con la condición anterior, que en el peor de los casos, las derivadas de las funciones de forma poseen discontinuidades finitas en la interface entre elementos.
- Las funciones deben ser completas: la completitud implica que la función de interpolación del elemento es capaz de representar exactamente un polinomio cuyos coeficientes son acordes a los grados de libertad

nodales, esto es, mientras la malla sea refinada, los valores solución exactos y sus derivadas se aproximan a valores constantes en el dominio del elemento y éstos deben ser representables.

Estas condiciones son propias de las funciones de forma más comunes utilizadas en los distintos tipos de elementos (que pueden encontrarse en el Anexo I), y proporcionan un modo sencillo de evaluar la convergencia del FEM en una gran gama de problemas. Sin embargo, no son condiciones necesarias ya que es posible construir elementos con funciones de forma que no las respeten y que, incluso así, sean convergentes a la solución exacta a medida que se refine la malla.

II.5. DOMINIO DE ORIGEN

Un método para la definición de elementos que resulta simple, sistemático y de fácil implementación computacional es el uso del dominio de origen. Este mecanismo consiste en utilizar un dominio auxiliar que será de naturaleza similar a aquel en el cual se pretende el elemento (dominio del elemento). Una vez definido el dominio de origen, se utiliza una transformación afín o mapeo que lo vincula con el dominio del elemento. De esta manera, a partir de un elemento simple, de geometría sencilla, puede definirse otro acorde a las necesidades del problema.

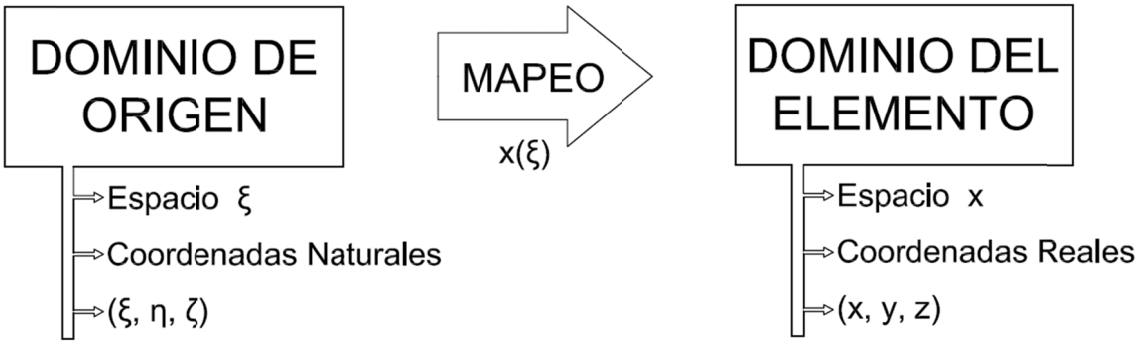


Figura II.1: Método del Dominio de Origen

Existen dos casos clásicos de dominios de origen que engloban casi la totalidad de los elementos comúnmente utilizados en el FEM. Éstos son el "Cuadrilátero Bilineal" y el "Hexaedro Trilineal" que son utilizados para la formación de elementos en dos y tres dimensiones respectivamente. Si bien, y como su nombre lo indica, ambos elementos son de primer orden (lineales) su tratamiento es fácilmente extensible a órdenes superiores y su simplicidad es oportuna a la hora de esclarecer conceptos.

II.5.1. CUADRILÁTERO BILINEAL

Se define el dominio de origen como un cuadrilátero regular de bordes rectos de dos unidades de longitud determinados por cuatro nodos identificados con las coordenadas naturales "ξ" y "η", en el plano $\mathbb{R}^2$ (denominado "espacio ξ"), según,

$$\boldsymbol{\xi_a} = \begin{Bmatrix} \xi_a \\ \eta_a \end{Bmatrix}, \qquad a = 1,2,3,4. \tag{II.5.1.1}$$

Siendo,

$$\boldsymbol{\xi_1} = \begin{Bmatrix} -1 \\ -1 \end{Bmatrix}, \boldsymbol{\xi_2} = \begin{Bmatrix} 1 \\ -1 \end{Bmatrix}, \boldsymbol{\xi_3} = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}, \boldsymbol{\xi_4} = \begin{Bmatrix} -1 \\ 1 \end{Bmatrix}. \tag{II.5.1.2}$$

Se pretende un cambio de coordenadas a través de un mapeo que transforme el cuadrilátero del "espacio ξ" en un elemento real en el plano $\mathbb{R}^2$ (denominado "espacio x") pero en coordenadas "x" e "y", definido por cuatro nodos dados por

$$\boldsymbol{x_a} = \begin{Bmatrix} x_a \\ y_a \end{Bmatrix}, \qquad a = 1,2,3,4. \tag{II.5.1.3}$$

Se propone por conveniencia expresar el mapeo de forma tal que cada punto del "espacio x" responda a

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{a=1}^{4} N_a(\boldsymbol{\xi})\, \boldsymbol{x_a}. \tag{II.5.1.4}$$

O bien, en término de sus coordenadas

$$x(\xi,\eta) = \sum_{a=1}^{4} N_a(\xi,\eta)\, x_a \tag{II.5.1.5}$$

$$y(\xi,\eta) = \sum_{a=1}^{4} N_a(\xi,\eta)\, y_a. \tag{II.5.1.6}$$

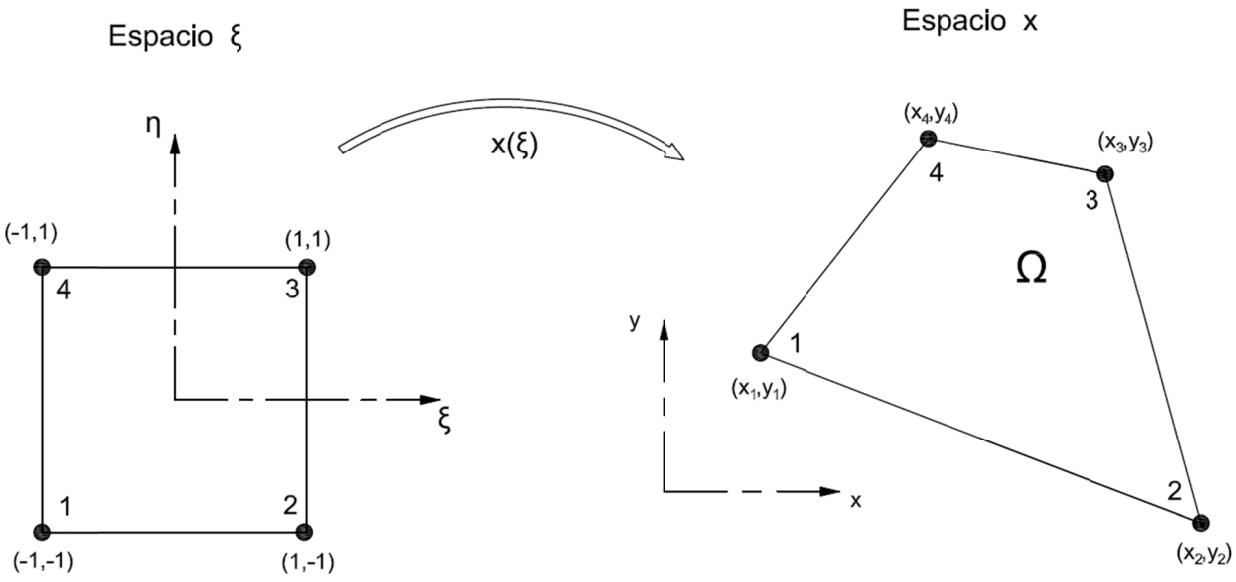


Figura II.2: Cuadrilátero Bilineal

Las funciones N_a se determinan a partir de asumir que las coordenadas del "espacio x" pueden expresarse como expansiones bilineales de las coordenadas del "espacio ξ" lo que coincide con la denominación del dominio de origen. Esto es

$$x(\xi,\eta) = \alpha_0 + \alpha_1\xi + \alpha_2\eta + \alpha_3\xi\eta \qquad \text{(II.5.1.7)}$$

$$y(\xi,\eta) = \beta_0 + \beta_1\xi + \beta_2\eta + \beta_3\xi\eta. \qquad \text{(II.5.1.8)}$$

Donde los valores de α_i y β_i, con $i = 0, \dots, 3$, deben determinarse con la relación que expresa que los nodos de ambos elementos deben coincidir, es decir,

$$\boldsymbol{x(\xi_a) = x_a}, \qquad a = 1,2,3,4. \qquad \text{(II.5.1.9)}$$

Una implicancia de la (II.5.1.9) que merece ser destacada se escribe:

$$N_a(\boldsymbol{\xi_b}) = \delta_{ab} \qquad \text{(II.5.1.10)}$$

Donde δ_{ab} representa el "Delta de Kronecker".

La combinación de las expresiones (II.5.1.7), (II.5.1.8) y (II.5.1.9) conduce a un sistema de ecuaciones que permite obtener los coeficientes α_i y β_i, con $i = 0, \dots, 3$. Reemplazando éstos en la ecuación (II.5.1.4) se obtiene la expresión para las funciones N_a, que toman la forma

$$N_a(\xi,\eta) = \frac{1}{4}(1 + \xi_a\xi)(1 + \eta_a\eta). \qquad \text{(II.5.1.11)}$$

El mapeo (II.5.1.4) con las funciones N_a expresadas de esta forma (II.5.1.11), presenta la particularidad de conservar rectas las líneas de coordenadas constantes en el "espacio ξ" como líneas rectas en el "espacio x" y, además, los bordes del dominio de origen son mapeados como bordes del elemento real.

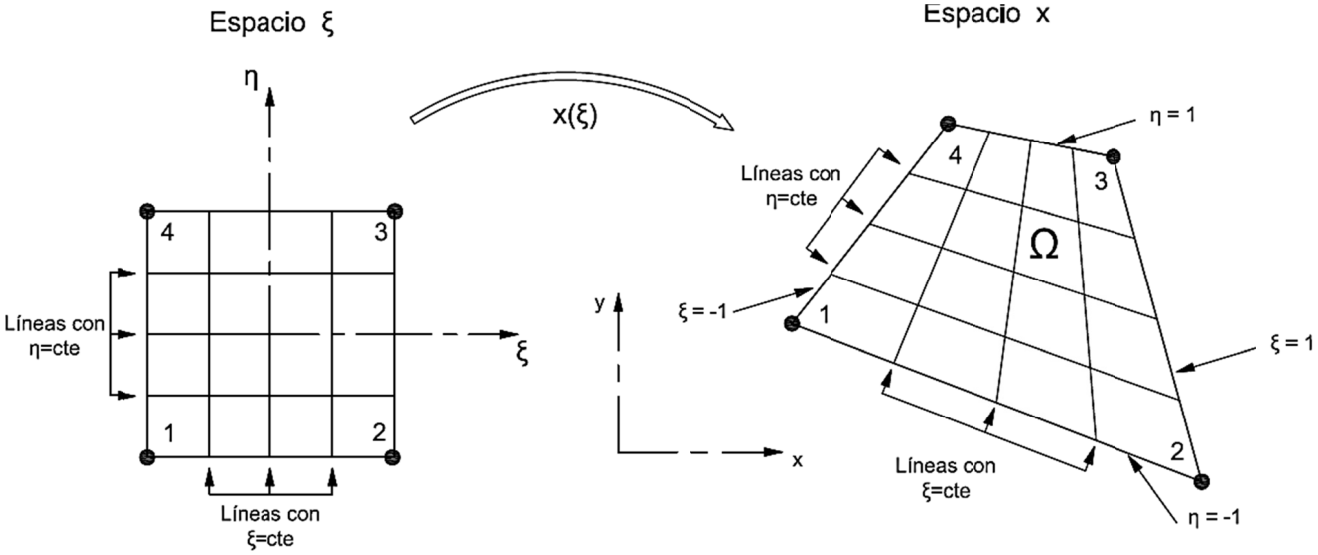


Figura II.3: Correspondencias entre líneas y puntos en el Cuadrilátero Bilineal

Por conveniencia práctica, las funciones de forma N_a suelen agruparse en un vector columna que permite, a través del producto matricial, generar el mapeo

(II.5.1.4) de manera sencilla, teniendo en cuenta que las coordenadas x_a pueden expresarse como un vector conforme. Es decir,

$$\boldsymbol{x}(\boldsymbol{\xi}) = [N_a]_{Ax1} \times [\boldsymbol{x_a}]_{nxA}, \qquad A = 4. \tag{II.5.1.12}$$

Donde "n" representa las dimensiones del espacio en el que se trabaja, en este caso n=2.

II.5.2. HEXAEDRO TRILINEAL

El hexaedro trilineal es una generalización del cuadrilátero bilineal y se desarrolla con un procedimiento análogo. No obstante, en este caso se trabaja en tres dimensiones y constituye la base de virtualmente todos los elementos de este tipo.

El dominio de origen se define como un hexaedro regular (cubo) con aristas que poseen dos unidades de longitud, y en cuyo centro geométrico se encuentra el centro de coordenadas del "espacio ξ". Naturalmente, se tendrán ocho nodos dados por las siguientes coordenadas naturales:

$$\boldsymbol{\xi_a} = \begin{Bmatrix} \xi_a \\ \eta_a \\ \zeta_a \end{Bmatrix}, \qquad a = 1,2,3,4,5,6,7,8. \tag{II.5.2.1}$$

Siendo,

$$\begin{aligned} \boldsymbol{\xi_1} = \begin{Bmatrix} -1 \\ -1 \\ -1 \end{Bmatrix}, \boldsymbol{\xi_2} = \begin{Bmatrix} 1 \\ -1 \\ -1 \end{Bmatrix}, \boldsymbol{\xi_3} = \begin{Bmatrix} 1 \\ 1 \\ -1 \end{Bmatrix}, \boldsymbol{\xi_4} = \begin{Bmatrix} -1 \\ 1 \\ -1 \end{Bmatrix}, \boldsymbol{\xi_5} = \begin{Bmatrix} -1 \\ -1 \\ 1 \end{Bmatrix}, \boldsymbol{\xi_6} \\ = \begin{Bmatrix} 1 \\ -1 \\ 1 \end{Bmatrix}, \boldsymbol{\xi_7} = \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix}, \boldsymbol{\xi_8} = \begin{Bmatrix} -1 \\ 1 \\ 1 \end{Bmatrix}. \end{aligned} \tag{II.5.2.2}$$

Nuevamente se requiere un mapeo que transforme el dominio de origen en un elemento real en el "espacio x", el cual se definirá por los nodos en coordenadas "x", "y", "z",

$$\boldsymbol{x_a} = \begin{Bmatrix} x_a \\ y_a \\ z_a \end{Bmatrix}, \qquad a = 1,2,3,4,5,6,7,8. \tag{II.5.2.3}$$

Del tipo

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{a=1}^{8} N_a(\boldsymbol{\xi})\, \boldsymbol{x_a}. \tag{II.5.2.4}$$

O bien, en término de sus coordenadas,

$$x(\xi, \eta, \zeta) = \sum_{a=1}^{8} N_a(\xi, \eta, \zeta)\, x_a \tag{II.5.2.5}$$

$$y(\xi,\eta,\zeta) = \sum_{a=1}^{8} N_a(\xi,\eta,\zeta)\, y_a \qquad \text{(II.5.2.6)}$$

$$z(\xi,\eta,\zeta) = \sum_{a=1}^{8} N_a(\xi,\eta,\zeta)\, z_a \qquad \text{(II.5.2.7)}$$

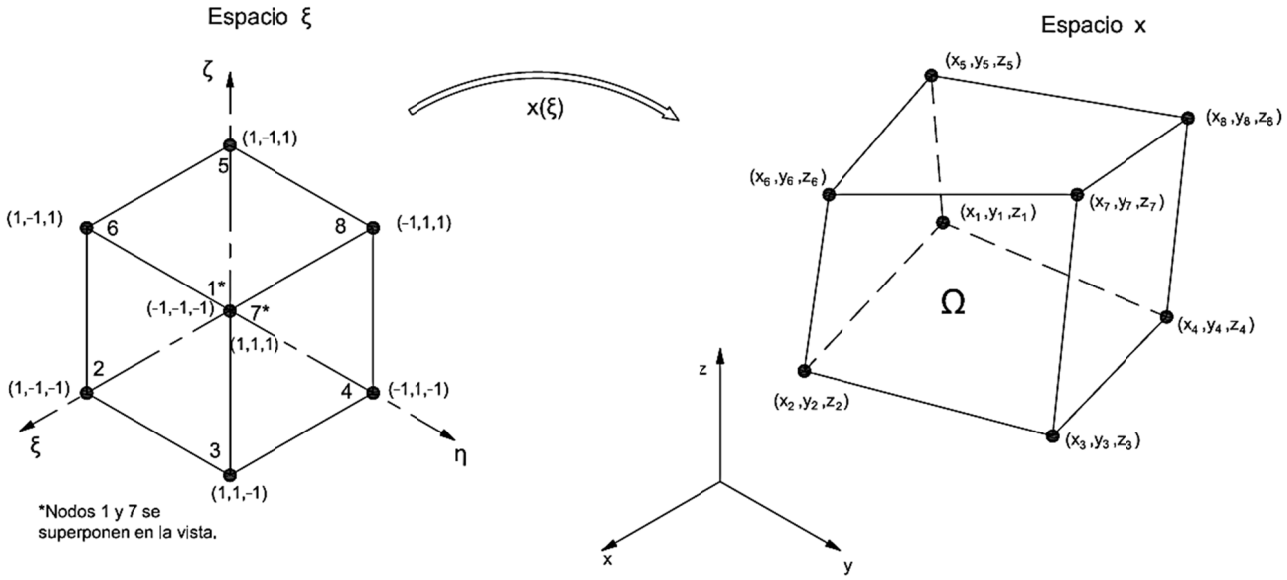


Figura II.4: Hexaedro Trilineal

También en este caso se propone que las coordenadas del "espacio x" sean combinaciones trilineales de las coordenadas del "espacio ξ", lo que deriva en:

$$x(\xi,\eta,\zeta) = \alpha_0 + \alpha_1\xi + \alpha_2\eta + \alpha_3\zeta + \alpha_4\xi\eta + \alpha_5\eta\zeta + \alpha_6\xi\zeta + \alpha_7\xi\eta\zeta \qquad \text{(II.5.2.5)}$$

$$y(\xi,\eta,\zeta) = \beta_0 + \beta_1\xi + \beta_2\eta + \beta_3\zeta + \beta_4\xi\eta + \beta_5\eta\zeta + \beta_6\xi\zeta + \beta_7\xi\eta\zeta \qquad \text{(II.5.2.6)}$$

$$z(\xi,\eta,\zeta) = \gamma_0 + \gamma_1\xi + \gamma_2\eta + \gamma_3\zeta + \gamma_4\xi\eta + \gamma_5\eta\zeta + \gamma_6\xi\zeta + \gamma_7\xi\eta\zeta \qquad \text{(II.5.2.7)}$$

Donde los coeficientes α_i, β_i y γ_i, con $i = 0, \ldots, 7$, se determinan con la condición de coincidencia de nodos, esto es,

$$\boldsymbol{x}(\boldsymbol{\xi_a}) = \boldsymbol{x_a}, \qquad a = 1,2,3,4,5,6,7,8. \qquad \text{(II.5.2.8)}$$

Obtenidos los coeficientes, reemplazando en las expresiones (II.5.2.5), (II.5.2.6) y (II.5.2.7), y despejando las funciones N_a, resulta

$$N_a(\xi,\eta,\zeta) = \frac{1}{8}(1+\xi_a\xi)(1+\eta_a\eta)(1+\zeta_a\zeta). \qquad \text{(II.5.2.9)}$$

El mapeo de la expresión (II.5.2.4) conformado por las funciones N_a antes expuestas conserva las mismas propiedades que en el caso bilineal, es decir, conserva rectas las líneas de coordenadas constantes en el "espacio ξ" como líneas rectas en el

"espacio x" y, además, los bordes del dominio de origen son mapeados como bordes del elemento real.

También es posible aplicar en este caso la expresión matricial del mapeo con la salvedad de recordad que son tres las dimensiones en las cuales se definen los dominios, a saber,

$$\boldsymbol{x}(\boldsymbol{\xi}) = [N_a]_{Ax1} \times [\boldsymbol{x_a}]_{nxA}, \qquad A = 8;\ n = 3. \tag{II.5.2.10}$$

II.6. ELEMENTOS ALTERNATIVOS

El cuadrilátero bilineal y el hexaedro trilineal, son quizás los dominios de origen más comunes y utilizados en el FEM, por su relación directa con los cuadriláteros y los elementos tipo bloque (hexaedros) que tienen gran difusión y utilidad. Sin embargo, existen otros elementos que son igualmente requeridos para diversas aplicaciones prácticas y que, por lo tanto, no pueden ser dejados de lado, a saber: triángulos, tetraedros, prismas y pirámides. La construcción de estos tipos de elementos puede realizarse a partir de dos procedimientos que se comentan a continuación.

El primero consiste en utilizar un dominio de origen conforme al elemento que se pretende. Esto es, si se desea construir, por ejemplo, un elemento triangular se definirá en el "espacio ξ" un triángulo de geometría sencilla y se obtendrá el mapeo correspondiente, de acuerdo al procedimiento descrito en la sección anterior. De igual modo ocurrirá para un tetraedro o un prisma en un espacio tridimensional. Este método, si bien reviste sencillez, presenta la desventaja de requerir el uso de tantos dominios de origen como tipos de elementos se requieran incluir, por lo que no será utilizado aquí.

La alternativa a lo antes mencionado y el segundo procedimiento de construcción de estos elementos (triángulos, tetraedros, pirámides y prismas) es la implementación del método de degeneración. Este método permite obtener un elemento alternativo a partir de un dominio de origen que no es conforme con él. De este modo, si se trabaja en dos dimensiones se utilizará siempre como dominio de origen el cuadrilátero bilineal, y análogamente con el hexaedro trilineal en el espacio.

II.6.1. EL MÉTODO DE DEGENERACIÓN

El método de degeneración consiste en la coalescencia de los nodos del elemento mediante la modificación de las funciones de forma, N_a, que conforman el mapeo. Esto permite "suprimir" aristas o bordes para alterar la geometría y lograr un elemento de naturaleza diferente a aquel que le dio origen.

A modo de ejemplo, supóngase que se requiere la construcción de un elemento tipo cuña (prisma triangular). Por tratarse de un elemento tridimensional, se recurre al hexaedro trilineal como dominio de origen y se superponen dos pares de nodos para eliminar una de las caras.

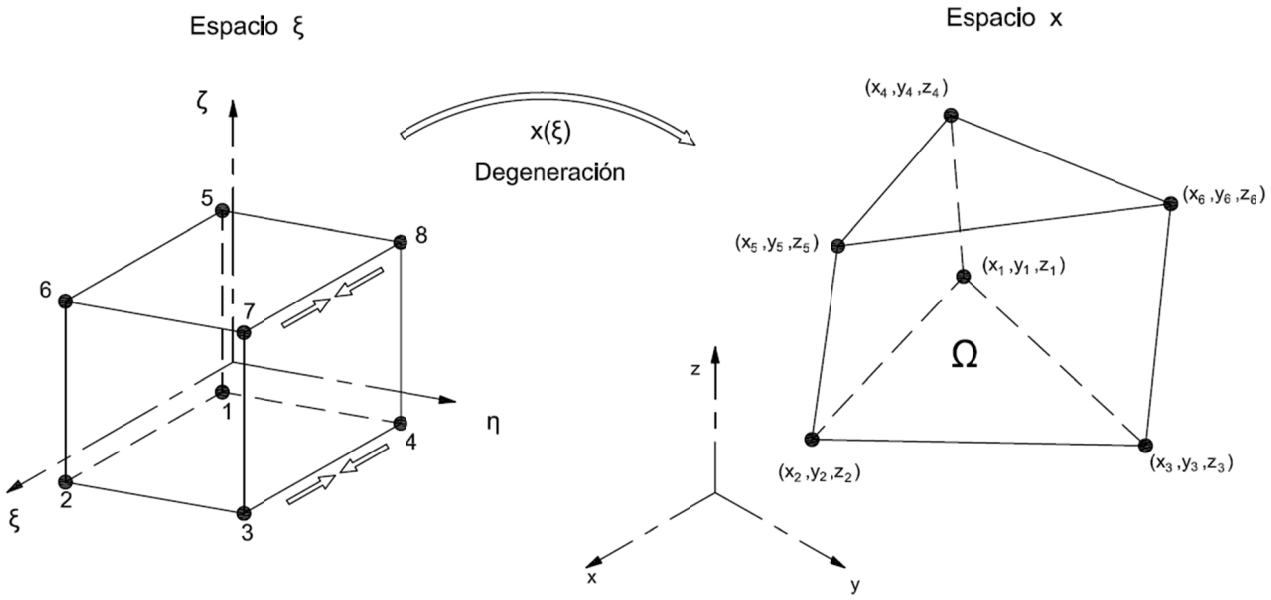


Figura II.5: Método de Degeneración, construcción de prisma triangular

Siguiendo la figura el nodo 7 coincidirá con el nodo 8 y, el 3 con el 4.

Tomando el mapeo dado por la (II.5.2.4) y expandiendo la sumatoria resulta:

$$\boldsymbol{x}(\boldsymbol{\xi}) = N_1(\boldsymbol{\xi})\boldsymbol{x_1} + N_2(\boldsymbol{\xi})\boldsymbol{x_2} + N_3(\boldsymbol{\xi})\boldsymbol{x_3} + N_4(\boldsymbol{\xi})\boldsymbol{x_4} + N_5(\boldsymbol{\xi})\boldsymbol{x_5} + N_6(\boldsymbol{\xi})\boldsymbol{x_6} + N_7(\boldsymbol{\xi})\boldsymbol{x_7} + N_8(\boldsymbol{\xi})\boldsymbol{x_8}. \quad \text{(II.6.1.1)}$$

Por la coincidencia entre los nodos antes mencionada, deberá ser

$$\boldsymbol{x_3} = \boldsymbol{x_4}$$
$$\boldsymbol{x_7} = \boldsymbol{x_8} \quad \text{(II.6.1.2)}$$

Luego, y obviando los argumentos de las funciones,

$$\boldsymbol{x} = N_1\boldsymbol{x_1} + N_2\boldsymbol{x_2} + (N_3 + N_4)\boldsymbol{x_3} + N_5\boldsymbol{x_5} + N_6\boldsymbol{x_6} + (N_7 + N_8)\boldsymbol{x_7}. \quad \text{(II.6.1.3)}$$

Reagrupando las sumatorias

$$\boldsymbol{x} = \sum_{a=1}^{3} N'_a\boldsymbol{x_a} + \sum_{a=5}^{7} N'_a\boldsymbol{x_a}. \quad \text{(II.6.1.4)}$$

Donde,

$$N'_a = \begin{cases} N_a & a = 1,2,5,6 \\ N_a + N_{a+1} & a = 3,7 \end{cases}. \quad \text{(II.6.1.5)}$$

De esta manera se obtienen las funciones de forma para el mapeo que genera un prisma triangular a partir de un hexaedro. Siguiendo el mismo razonamiento es posible construir los elementos restantes sin mayor dificultad.

II.7. ELEMENTOS DE ORDEN SUPERIOR

En algunos casos el uso exclusivo de funciones lineales en el desarrollo de elementos no satisface íntegramente los requerimientos del problema, y es aquí donde se debe dejar de lado el primer orden. Los elementos de orden superior presentan dos ventajas claras, en primer lugar, permiten aproximaciones más precisas a las variaciones espaciales exactas de los parámetros del problema y, por otro lado, son capaces de representar dominios con geometrías más complejas de manera más fehaciente. No obstante, su construcción y utilización es más costosa y puede prolongar considerablemente los tiempos de ejecución del FEM. Por lo tanto, la selección apropiada del tipo de elemento será una solución de compromiso que variará según el caso práctico que se evalúe.

La clave para la construcción de estos elementos yace en las expresiones que vinculan las coordenadas del "espacio ξ" con las del "espacio x": ya no se impondrá una expansión lineal (bilineal o trilineal) sino una del orden deseado para el elemento. Naturalmente, el incremento del orden de dichas relaciones, expresiones (II.5.1.7), (II.5.1.8), (II.5.2.5), (II.5.2.6) y (II.5.2.7), conllevará al incremento de la cantidad de nodos necesarios para que el sistema de ecuaciones resultante se encuentre determinado. Por ejemplo un cuadrilátero completo de segundo orden tendrá nueve nodos y un hexaedro de la misma característica tendrá veintisiete. A modo de aclaración, decir que un elemento es "completo" implica que todas sus funciones de forma poseen el mismo orden. Pudiendo existir el caso en el que se combinen, en un mismo elemento, distintos órdenes resultando en lo que se denomina elementos incompletos o de transición.

Como es de esperarse, la aplicación del dominio de origen es perfectamente válida para generar elementos de orden superior siempre que se tenga en cuenta que el elemento auxiliar deberá ser del orden deseado. Tómese por ejemplo el caso plano de un cuadrilátero de segundo orden, el dominio de origen en este caso será un cuadrado de dos unidades de lado y con nueve nodos: cuatro en los vértices, cuatro en el punto medio de cada lado y uno en el centro.

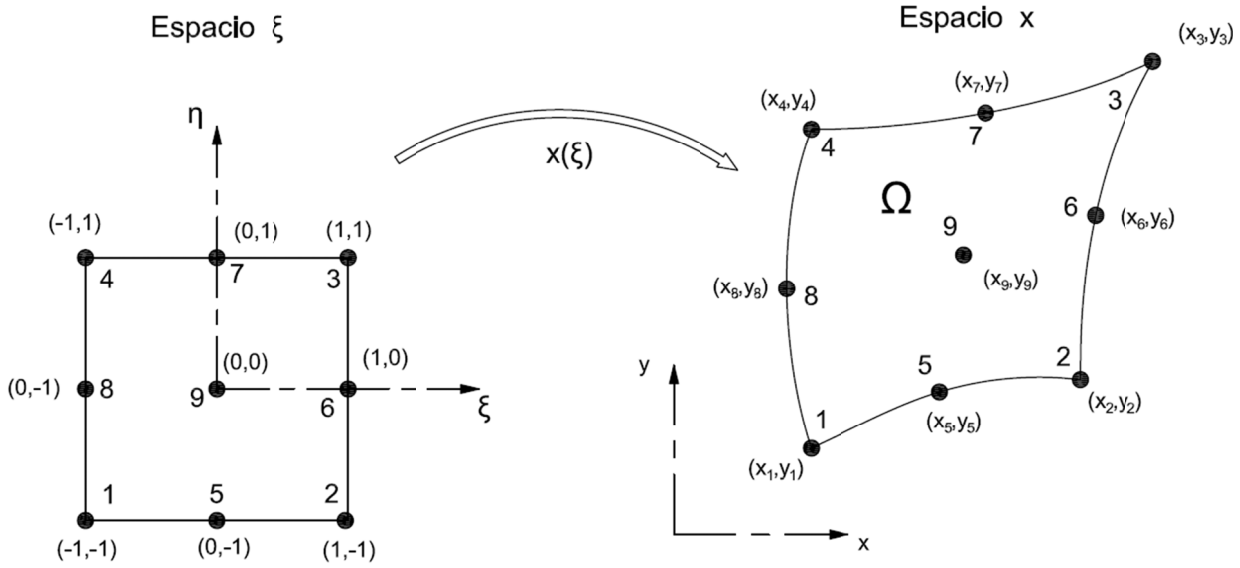


Figura II.6: Dominio de Origen para un Cuadrilátero de Segundo Orden

El mapeo propuesto será similar al caso lineal (II.5.1.4), del tipo

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{a=1}^{9} N_a(\boldsymbol{\xi})\, \boldsymbol{x_a}. \tag{II.7.1}$$

Pero las relaciones entre las coordenadas $\boldsymbol{\xi}$ y $\boldsymbol{x}$ serán de segundo orden, según lo esperado. Esto es,

$$x(\xi,\eta) = \alpha_0 + \alpha_1\xi + \alpha_2\eta + \alpha_3\xi\eta + \alpha_4\xi^2 + \alpha_5\eta^2 + \alpha_6\xi\eta^2 + \alpha_7\xi^2\eta + \alpha_8\xi^2\eta^2 \tag{II.7.2}$$

$$y(\xi,\eta) = \beta_0 + \beta_1\xi + \beta_2\eta + \beta_3\xi\eta + \beta_4\xi^2 + \beta_5\eta^2 + \beta_6\xi\eta^2 + \beta_7\xi^2\eta + \beta_8\xi^2\eta^2 \tag{II.7.3}$$

Se impone nuevamente la condición de coincidencia de nodos,

$$\boldsymbol{x}(\boldsymbol{\xi_a}) = \boldsymbol{x_a}, \qquad a = 1,2,3,4,5,6,7,8,9. \tag{II.7.4}$$

Así, se tiene un sistema de ecuaciones en los coeficientes α_i y β_i, con $i = 0, \dots, 8$, que, luego de resolverse, permite mediante el reemplazo en la expresión (II.7.1) la determinación de las funciones N_a, y, por ende, del mapeo.

Este proceso es aplicable a cualquier otro elemento y a cualquier otro orden según sea necesario. También es posible recurrir a la degeneración para construir elementos alternativos con órdenes mayores que uno.

Existe un modo sistemático de derivar las funciones de forma de ciertos elementos que resulta apropiado para la implementación computacional. Éste se vale del resultado publicado por Joseph-Louis de Lagrange en el año 1795 (luego de los trabajos de Waring y Euler) consistente en una base monómica combinable para

interpolar dispersiones de puntos: lo que hoy se conoce como Polinomios de Lagrange.

Los polinomios de Lagrange se definen como:

$$l_n^m(\xi) = \frac{\prod_{\substack{i=1 \\ i\neq n}}^{m}(\xi - \xi_i)}{\prod_{\substack{i=1 \\ i\neq n}}^{m}(\xi_i - \xi_n)} \tag{II.7.5}$$

Y su combinación apropiada deriva en las funciones de forma de los elementos requeridas.

La utilización de los polinomios de Lagrange variará según la dimensión del espacio donde se construye el elemento, el orden del elemento que se desea construir y el nodo al que la función de forma hace referencia. Es decir, la dimensión del espacio indicará cuántos polinomios deberán multiplicarse para construir la función de forma, uno para cada coordenada. Luego, el orden del elemento estará indicado por el superíndice "m". Finalmente, el subíndice "n" indicará la relación entre los índices de los nodos. Para esclarecer lo anterior, considérese el siguiente ejemplo. La función de forma genérica para el elemento cuadrilátero de segundo orden de la Figura II.6 es,

$$N_a(\xi,\eta) = l_n^m(\xi) l_n^m(\eta) = l_b^2(\xi) l_c^2(\eta). \tag{II.7.6}$$

Siguiendo la Figura II.7 la combinación de índices determina los valores de "b" y "c" dando, a partir del reemplazo correspondiente, las funciones de forma que, para citar algunas, serán

$$N_1(\xi,\eta) = l_1^2(\xi) l_1^2(\eta) = \frac{1}{4}\xi\eta(\xi - 1)(\eta - 1) \tag{II.7.7}$$

$$N_5(\xi,\eta) = l_2^2(\xi) l_1^2(\eta) = \frac{1}{2}\eta(1 - \xi^2)(\eta - 1) \tag{II.7.8}$$

$$N_9(\xi,\eta) = l_2^2(\xi) l_2^2(\eta) = (1 - \xi^2)(1 - \eta^2) \tag{II.7.9}$$

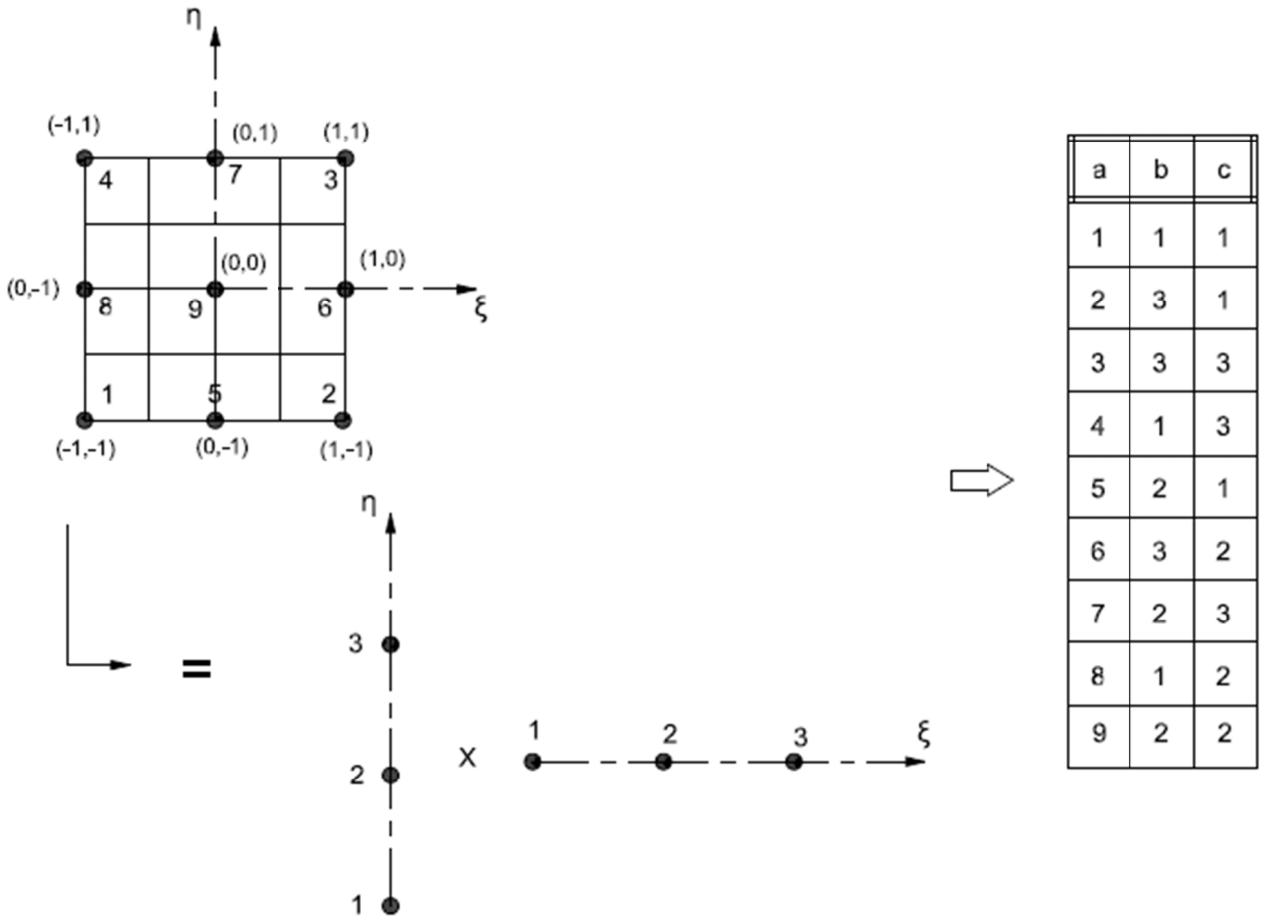


Figura II.7: Construcción de listado de índices para polinomios de Lagrange

II.8. ELEMENTOS ISOPARAMÉTRICOS

Un elemento isoparamétrico será aquel que posea idénticas funciones N_a para definir el mapeo que lo construye y para definir sus funciones de interpolación. De ahí que se utiliza "el mismo parámetro" en la generación del elemento y en los cálculos del FEM: las funciones de forma N_a.

De forma analítica, sea $\overline{\Omega}_o$ el dominio de origen en el "espacio ξ" de un elemento arbitrario, y sea el mapeo que lo transforma al "espacio x" en el dominio $\overline{\Omega}_e$

$$\boldsymbol{x}: \overline{\Omega}_o \rightarrow \overline{\Omega}_e \tag{II.8.1}$$

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{a=1}^{n_{en}} N_a(\boldsymbol{\xi})\, \boldsymbol{x_a}. \tag{II.8.2}$$

Donde n_{en} indica el número de nodos que tiene el elemento.

Luego, el elemento será isoparamétrico si su función de interpolación, puede escribirse como

$$\boldsymbol{u^h}(\boldsymbol{\xi}) = \sum_{a=1}^{n_{en}} N_a(\boldsymbol{\xi})\, \boldsymbol{d_a}. \qquad \text{(II.8.3)}$$

La importancia del concepto de elemento isoparamétrico radica en dos aspectos. Por un lado, las condiciones suficientes de convergencia se satisfacen de manera automática, ahorrando cualquier tipo de verificación posterior en este sentido. Por otro, que habiendo generado las funciones N_a para construir el elemento éstas ya se encuentran disponibles para ser utilizadas en los cálculos de procesamiento de FEM, lo que evita el proceso de definición de funciones de interpolación en la implementación computacional. Además, los elementos derivados del Cuadrilátero Bilineal y del Hexaedro Tilineal tienen la característica de ser isoparamétricos.

Capítulo III: El Programa de Vinculación

III.1. Nociones Generales

Luego del Capítulo II donde se introdujeron los conceptos teóricos que sustentan el desarrollo del tema, se busca, en este tercer capítulo, presentar de manera conceptual la estructura y funcionamiento del que será el programa objetivo. Siguiendo con lo dicho en el capítulo introductorio se pretende un código que tome como input las mallas del FEM y las grillas del UVLM, y genere una matriz de localización que indique en qué elemento de la ME se encuentra cada punto de control y cada nodo de la GA.

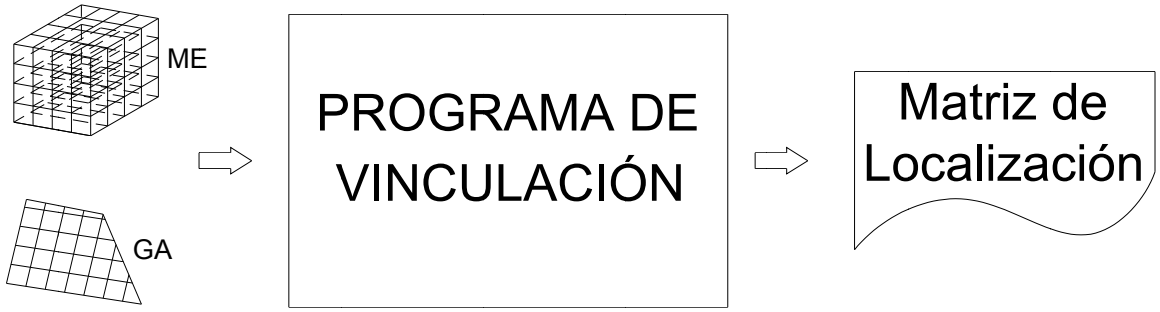


Figura III.1: Diagrama Genérico del Programa de Vinculación

A pesar de la metodología y los procesos que pueden aplicarse para desarrollar el software, existen funciones generales que éste debe cumplir y que definen su estructura interna. Estas funciones pueden clasificarse en tres grandes grupos, a saber: las destinadas a la recolección de datos provenientes de las mallas y grillas, las que seleccionan y organizan en grupos los elementos de la ME para su posterior evaluación, y las que consisten en la localización propiamente dicha. Conceptos que serán ampliados en secciones ulteriores.

La organización en grupos de las funciones del programa permite, como se dijo, establecer su estructura interna, la cual se divide en tres módulos bien definidos e

identificables con las tareas asignadas a los conjuntos de funciones: El Módulo de Ingreso, El Módulo de Selección y El Módulo de Localización. De este modo, si existe más de una alternativa para ejecutar los distintos módulos, éstas pueden combinarse de manera simple para obtener la performance buscada.

En una descripción sintetizada, el funcionamiento del Programa de Vinculación se condensa en la operativa de cada módulo. En primer lugar, el Módulo de Ingreso traduce los datos de las mallas (nodos, elementos y puntos de control), dentro del código principal del Programa de Vinculación. Seguidamente, el Módulo de Selección asocia a cada punto de la GA un conjunto de elementos de la ME donde sería probable encontrarlo. Finalmente, el Módulo de Localización analiza cada elemento del conjunto para confirmar la localización. En el correr del capítulo se tratará cada módulo de manera individual.

Adicionalmente, se incluyen sistemas de medida de performance para poder evaluar la eficiencia de los códigos. También, se utiliza un software mallador que permite construir mallas con diversos formatos (tanto para las ME como para las GA) que serán los datos de entrada del programa. Con estos dos últimos factores el diagrama se completa según la Figura III.2.

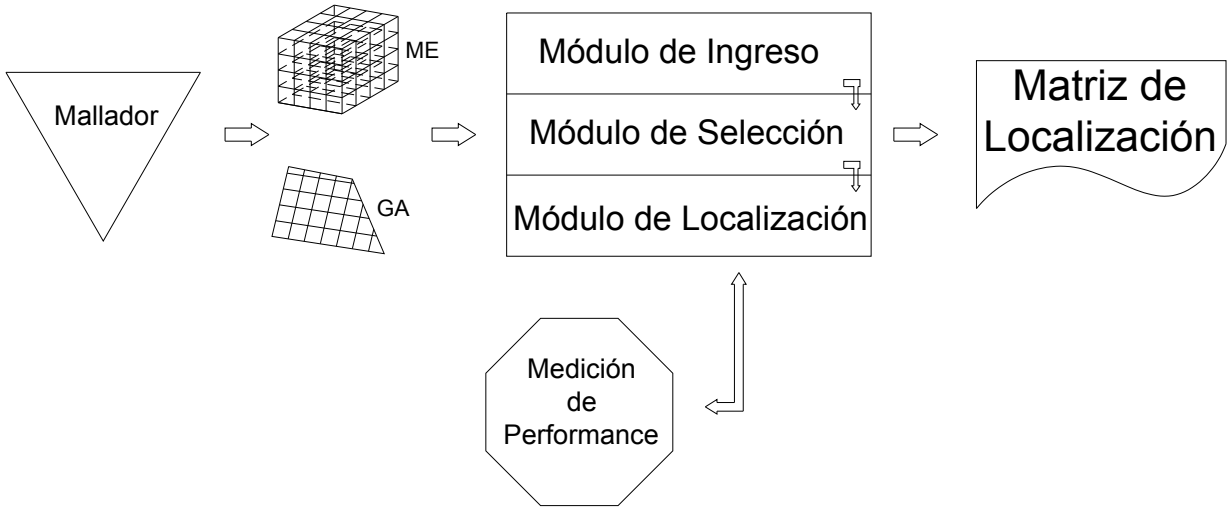


Figura III.2: Diagrama Completo del Programa de Vinculación

Comprendiendo el funcionamiento del Programa de Vinculación será posible confeccionar algoritmos que se adapten a los objetivos de cada módulo. Del mismo modo, es necesario realizar consideraciones especiales acerca del proceso de implementación, para poder dar paso al análisis de los métodos de procesamiento y ejecución de datos que serán finalmente aplicados y evaluados.

III.2. IMPLEMENTACIÓN

La implementación del Programa de Vinculación se realiza a través del entorno programación MATLAB®. Dado que provee sistemas para el cálculo numérico a través

de un lenguaje de alto nivel, constituye una opción simple a la hora de programar algoritmos. Por otro lado, cuenta con una extensa biblioteca de funciones matemáticas, con sistemas de ejecución rápida para el manejo de vectores y matrices, y con mecanismos de evaluación y corrección continua de códigos.

En este sentido, el programa final será un script o una función ejecutable desde la ventana de comandos de MATLAB, que tomará por argumentos los archivos de texto emitidos por el mallador y dará como variable de salida la Matriz de Localización correspondiente. Tendrá también bibliotecas de datos y funciones específicas programadas de forma externa que serán ejecutadas dentro del código principal.

III.3. EL MALLADOR Y SUS ARCHIVOS DE SALIDA

El generador de mallas elegido para dar origen a las mallas y grillas es el desarrollado por Christophe Geuzaine y Jean-Francois Remacle, denominado GMSH® [6]. Gmsh se trata de un mallador de tres dimensiones con un soporte para Diseño Asistido por Computadora (CAD), creado para el desarrollo eficiente de mallas con datos paramétricos y capacidades avanzadas de visualización. Cuenta con cuatro módulos de trabajo: Geometría, Malla, Solver y Post-Procesamiento, y permite tanto el uso de una interface gráfica como la edición de archivos de texto para su manipulación.

Adicionalmente, el mallador permite la selección de diversos algoritmos de discretización, con soporte para numerosos tipos de elementos y funciones de refinamiento. Por otro lado, es posible construir geometrías relativamente complejas sobre las cuales generar las mallas, o bien, importar archivos en los formatos de CAD clásicos.

La principal característica a analizar de Gmsh son los detalles de los archivos de salida para el almacenamiento, ya sea de geometrías construidas, mallas generadas, o ambas, de manera superpuesta. Este rasgo cobra importancia debido a que serán estos archivos los que ingresarán como entrada al Programa de Vinculación, y su comprensión es imprescindible para recolectar los datos necesarios.

Luego de haber construido la geometría, de haber establecido sobre ésta los parámetros y características de la malla y, por último, de haber generado la malla en sí, el mallador crea un archivo de texto que contiene todos los datos que la definen. Es posible decidir entre la utilización dos formatos de archivo, uno en código ASCII y otro en código binario. El sistema de implementación que se usará en el Programa de Localización no admite como datos de entrada archivos en código binario, por ende, la atención será puesta en el formato en código ASCII.

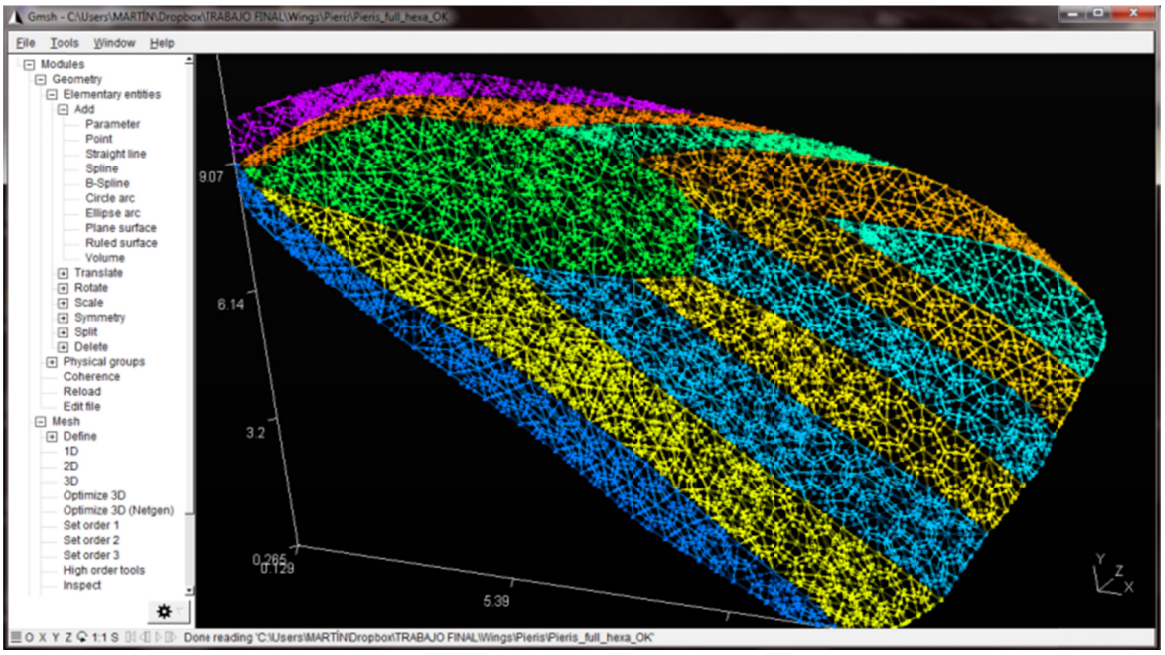


Figura III.3: Vista General del Mallador GMSH

El archivo de texto de salida se nombra con la extensión ".msh", haciendo referencia a que se trata de un archivo de malla. En su interior, se encuentran varios bloques definidos con líneas de título que se identifican con el signo "$" y tratan diversos aspectos de la malla. Sin embargo, para la utilización de estos archivos como datos de entrada en el Programa de Vinculación, sólo son necesarios los bloques concernientes a los nodos y a los elementos, ya que los siguientes sólo hacen referencia a datos necesarios para el post-procesamiento interno de Gmsh.

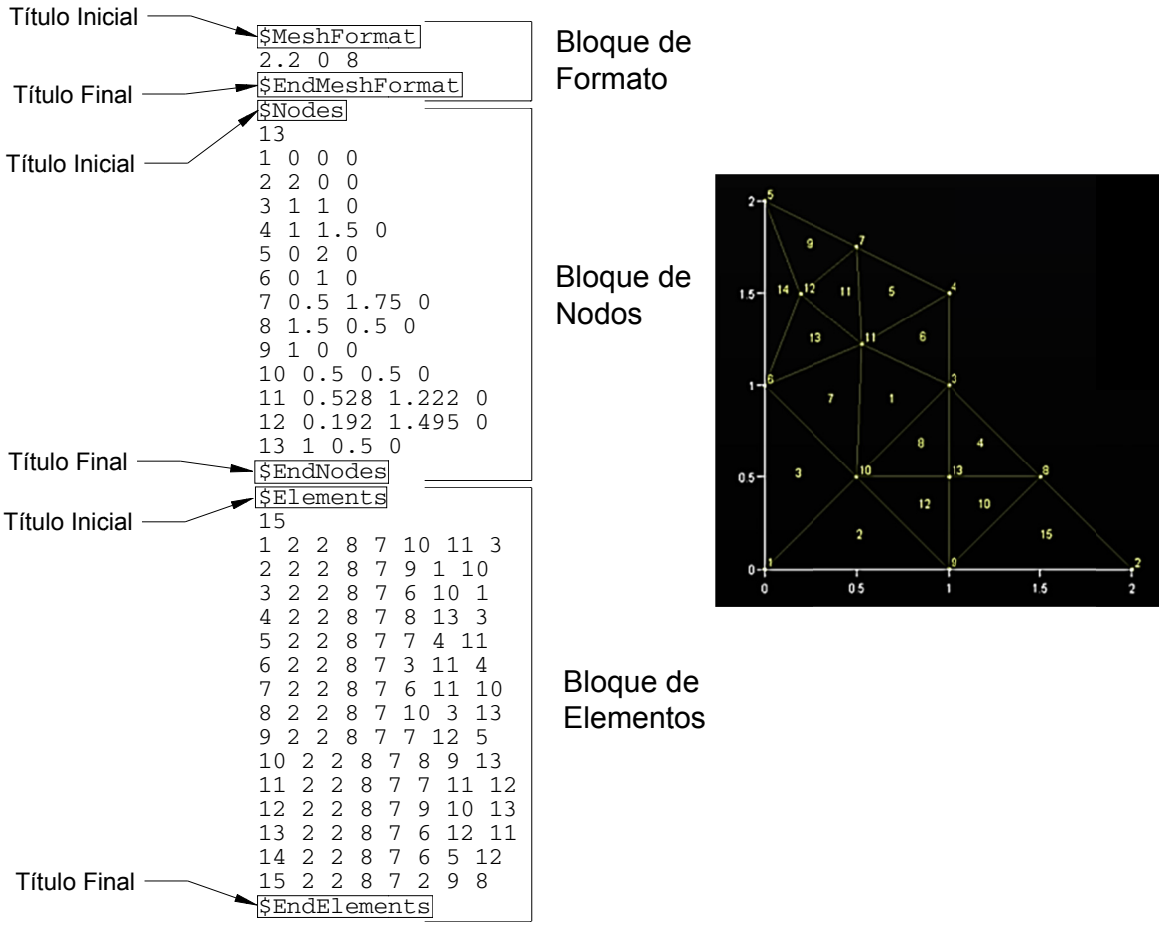


Figura III.4: Archivo de Salida de GMSH y Malla que Representa

El primer bloque o "bloque de formato" cuenta sólo con una línea operativa y dos líneas de título: una de inicio y otra de fin. En él se indica la versión del programa, el tipo de archivo, y el tamaño del mismo. Estos datos no son útiles y deben ser descartados.

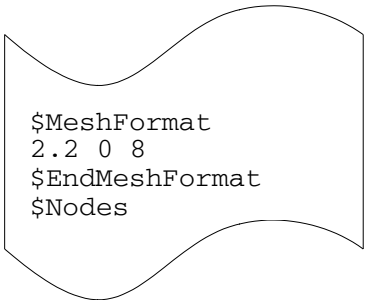


Figura III.5: Bloque de Formato

El segundo bloque o "bloque de nodos" proporciona dos datos importantes, el primero es el número de nodos de la malla, y el segundo un listado de las coordenadas de estos nodos que, por defecto, son coordenadas cartesianas. La lista posee los datos de cada nodo en una línea que inicia con el índice del nodo (número de identificación global), seguido por las tres coordenadas del sistema: "x", "y", y "z". Finalmente y como en todos los casos, se encuentra la línea de título final que cierra el bloque.

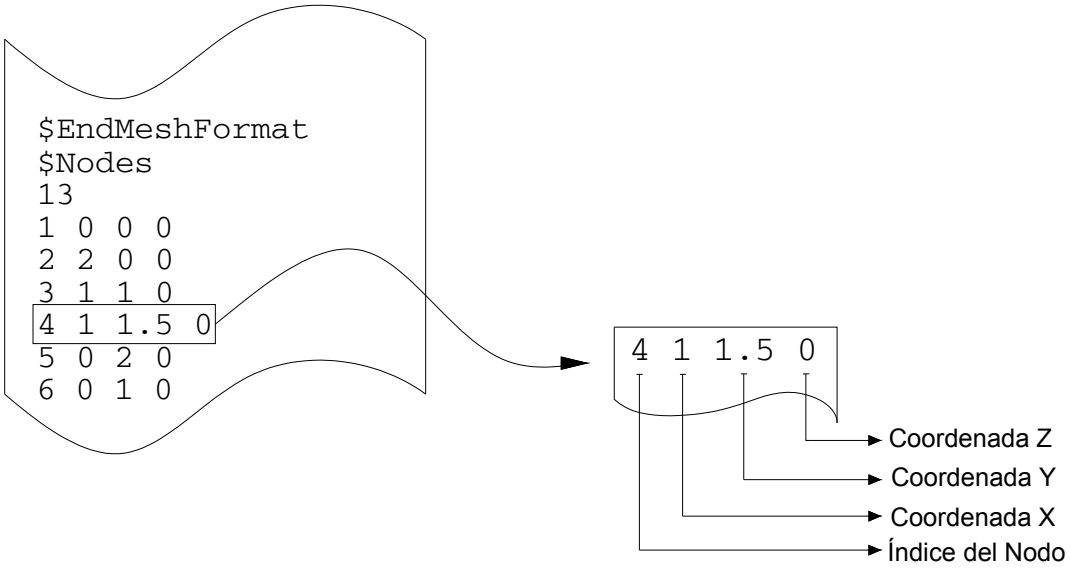


Figura III.6: Bloque de Nodos

El tercer bloque o "bloque de elementos" consiste en información sobre los elementos que componen la malla. Inicia con la línea de título correspondiente y luego indica la cantidad total de elementos de la malla. Las líneas subsiguientes conforman un listado de los elementos y sus características, asignando una línea para cada elemento. Cada línea se compone, a su vez, de una sucesión de números que se relacionan con las particularidades del elemento. En primer lugar, se encuentra el número de elemento, entendido como un índice global de identificación. Seguidamente, se indica con una codificación interna el tipo de elemento del que se trata (ver [6]) algunos de los cuales se muestran a continuación.

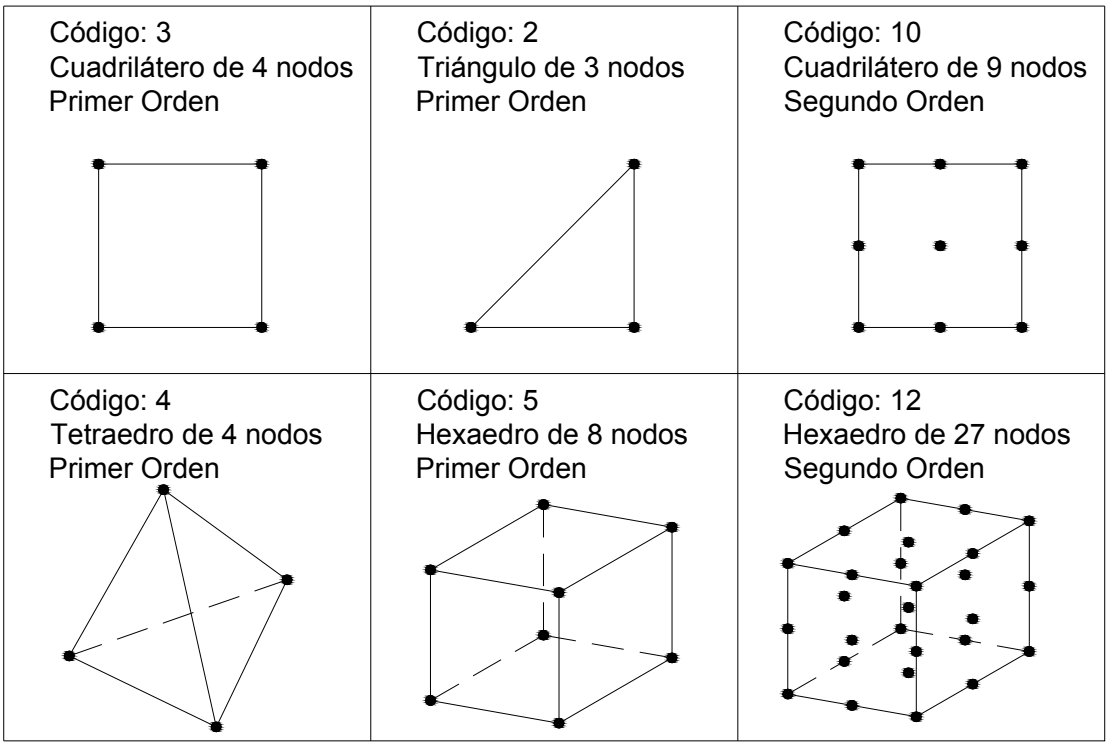


Figura III.7: Codificación Sintetizada de los Tipos de Elementos

En tercer lugar, se indica el número de etiquetas, o "tags", que posee el elemento. Se entiende por etiqueta a un parámetro característico del elemento que es utilizado en los módulos de solver y post-procesamiento de Gmsh, como por ejemplo la entidad geométrica a la que pertenece el elemento. Inmediatamente después del número de etiquetas, se encuentran los "tags" propiamente dichos, tantos como el número anterior lo indique. Estos datos no son útiles para los algoritmos de localización por lo que deben despreciarse. Finalmente se encuentra la lista de nodos que conforman el elemento, ordenados según el sistema propio del mallador e identificados con su índice global.

Como siempre, al final del bloque se encuentra la línea de título de cierre. Esta línea puede asociarse con el final del contenido útil del archivo de datos de la malla, cualquier bloque extra que éste contenga, no será considerado aquí.

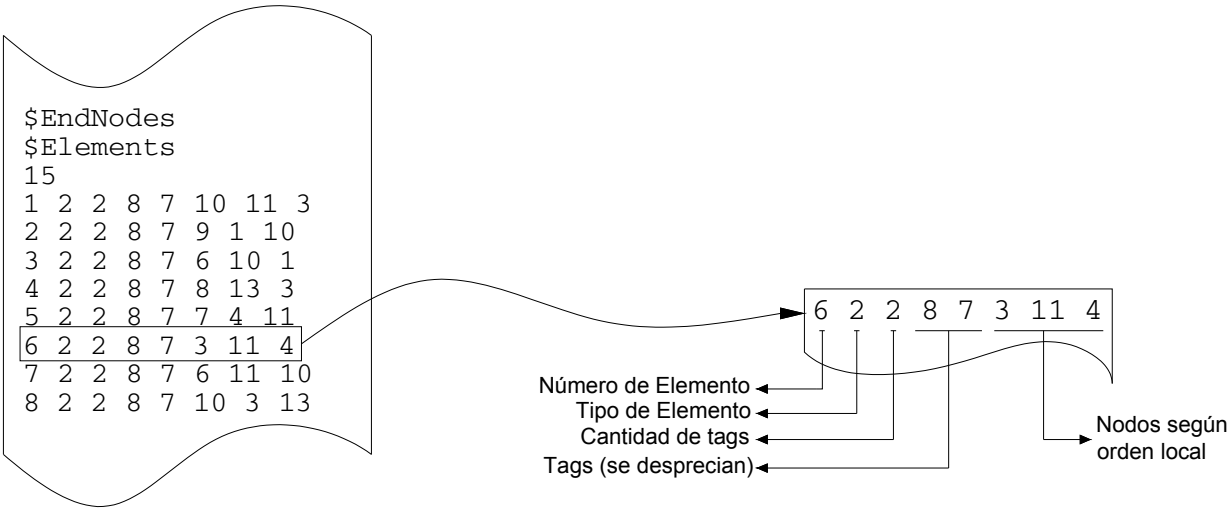


Figura III.8: Bloque de Elementos

III.4. EL MÓDULO DE INGRESO

El primer módulo del Programa de Vinculación, denominado Módulo de Ingreso, tiene por objetivo procesar los datos de entrada provenientes del mallador para convertirlos en estructuras manejables dentro de sí mismo. Por otro lado, este módulo inicial deberá cargar otros parámetros que serán utilizados; como ser, las particularidades de los elementos, sus funciones de forma, matrices jacobianas, entre otros. De este modo, se ponen a disposición de los módulos siguientes las variables internas que serán procesadas oportunamente para alcanzar el objetivo final.

Como se vio en la sección anterior, los archivos de salida del mallador son archivos de texto en código ASCII y son portadores de toda la información necesaria sobre la malla. Luego, el Módulo de Ingreso contará con una función de lectura estandarizada de acuerdo al formato de los archivos ".msh". Dicha función deberá, de manera general, desplazar el cursor a través del archivo de texto diferenciando los caracteres necesarios de aquellos que no aportan información, para generar variables a partir de los primeros y descartar los segundos.

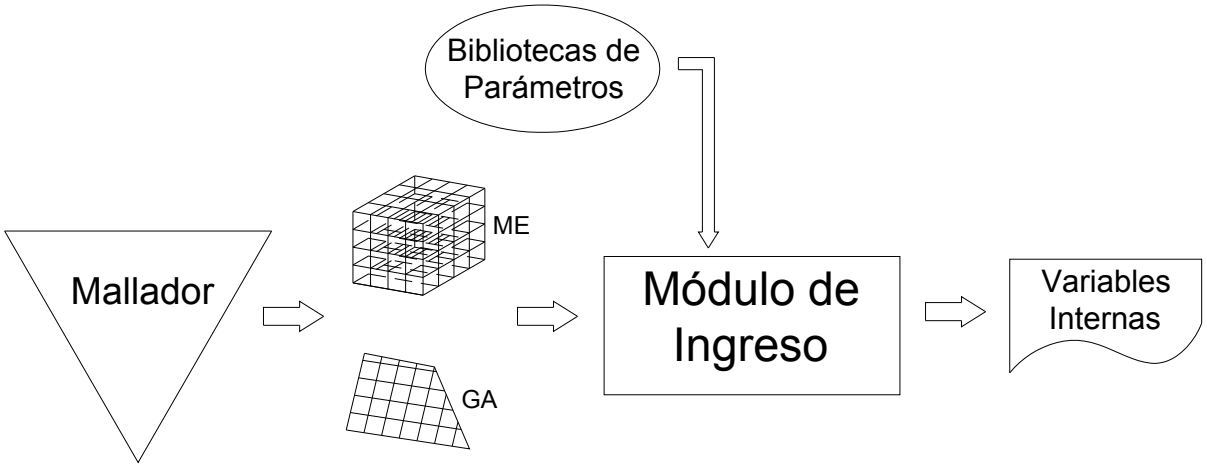


Figura III.9: Diagrama del Módulo de Ingreso

Una diferenciación debe hacerse sobre el procesamiento de las mallas estructurales respecto al de las grillas aerodinámicas, a pesar de que ambas se encuentran en archivos de idéntico formato. Las ME deben almacenarse de manera integral en el programa de localización, esto es, los datos de sus nodos (índices y coordenadas) son igual de importantes que los datos de sus elementos (tipos y nodos que los integran). Para las GA, en cambio, no es necesario ingresar información sobre sus elementos, es decir, una grilla del UVLM puede trabajarse a los fines del Programa de Vinculación como una lista de nodos y puntos de control sin ningún tipo de vínculo adicional. Esta diferencia permite reducir el espacio de almacenamiento de datos y aumentar la eficiencia del primer módulo.

III.5. EL MÓDULO DE SELECCIÓN

El módulo segundo o Módulo de Selección tiene por finalidad realizar una preselección de los elementos donde es probable encontrar cada punto que requiere ser situado. En otras palabras, para cada punto a localizar de la GA, existe un grupo de elementos de la ME en los que las probabilidades de localización son mayores. Determinar de manera eficaz cuáles son estos elementos es el objetivo del segundo módulo.

Consecuentemente, la importancia del Módulo de Selección yace en reducir virtualmente al máximo el número de evaluaciones que realizará el módulo siguiente. Este número fluctuará en un rango que va desde una única evaluación hasta una cantidad igual al número total de elementos de la ME, dependiendo de qué tan precisa sea la selección. No obstante, debe considerarse que los elementos descartados por el módulo 2 no serán, en proncipio, considerados nuevamente, consiguientemente, deberá procurarse garantizar que el elemento que efectivamente posee al punto no sea filtrado.

La metodología de operación de este módulo pretende ser relativamente simple y consiste principalmente en nociones topológicas y geométricas, como ser, cercanía entre puntos, colindancia entre nodos y adyacencia entre elementos. Esto significa que no es posible localizar exactamente un punto sin ejecutar el módulo siguiente.

Si se tiene presente que las evaluaciones de los elementos son sucesivas, cobra importancia otro concepto que se implementa en el Módulo de Selección: el orden de los elementos. Considerando que al último módulo del programa se transfiere un conjunto de elementos para ser evaluados, asignar un grado de prioridad a cada uno puede resultar beneficioso.

Téngase por ejemplo que para un punto dado, los elementos seleccionados son los que cumplen con determinada condición. Si dicha condición puede valorarse cuantitativamente, permitiendo ordenar el conjunto a partir de asignar un nivel a cada elemento, es altamente probable que el módulo siguiente localice al punto antes de finalizar la evaluación de todo el conjunto. Lo que resultará en una reducción del tiempo de ejecución.

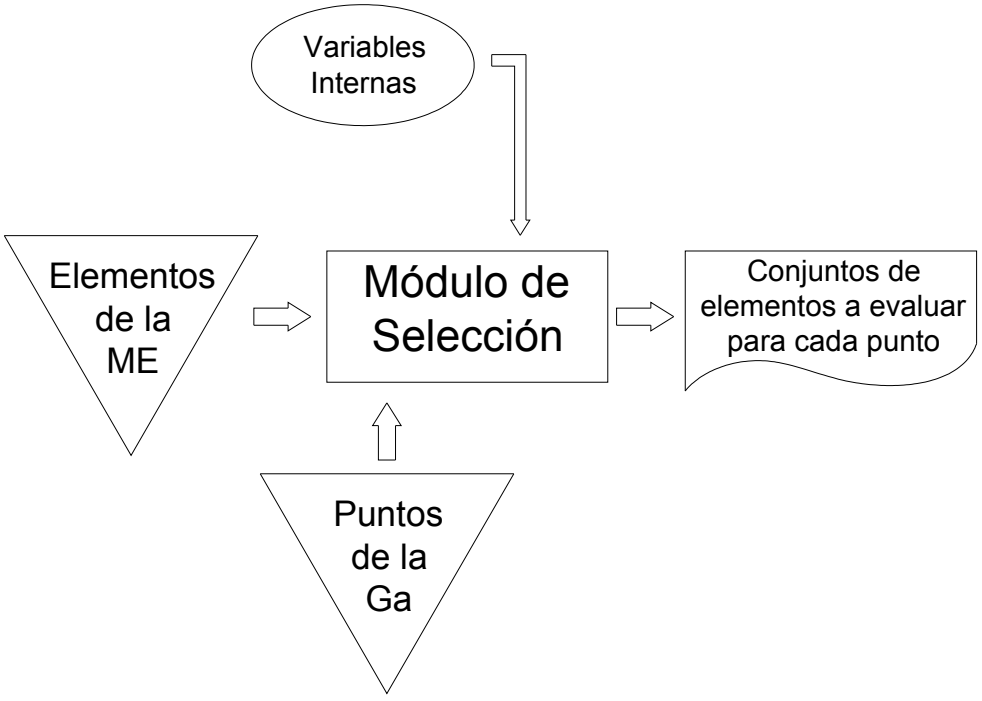


Figura III.10: Diagrama del Módulo de Selección

Sumariando, el Módulo de Selección asociará a cada punto a localizar, un conjunto ordenado de elementos, que probablemente lo contengan, para que sean evaluados en el módulo 3. Esta acción se representa en la Figura III.10.

III.6. EL MÓDULO DE LOCALIZACIÓN

El tercer y último módulo, el Módulo de Localización, cumplimenta finalmente el objetivo de todo el programa. Este módulo tiene por propósito evaluar los elementos que el Módulo de Selección indique y determinar indefectiblemente en cuál de ellos se encuentra el punto en análisis. El resultado de cada ciclo de evaluaciones del tercer módulo debe ser biunívoco, ya que, sin dar lugar a la ambigüedad, cada punto de la GA se corresponderá con uno y sólo un elemento de la ME. Llegado el caso particular en que un punto se posicione en la interface entre dos o más elementos, el Módulo de Localización deberá elegir arbitrariamente uno, y sólo uno, de ellos ya que para los fines de la vinculación es indistinto.

Los métodos utilizados en este módulo son más complejos que aquellos usados en el Módulo de Selección. Aquí se recurre a las propiedades mismas de los elementos, sus sistemas de construcción y otros recursos del análisis numérico.

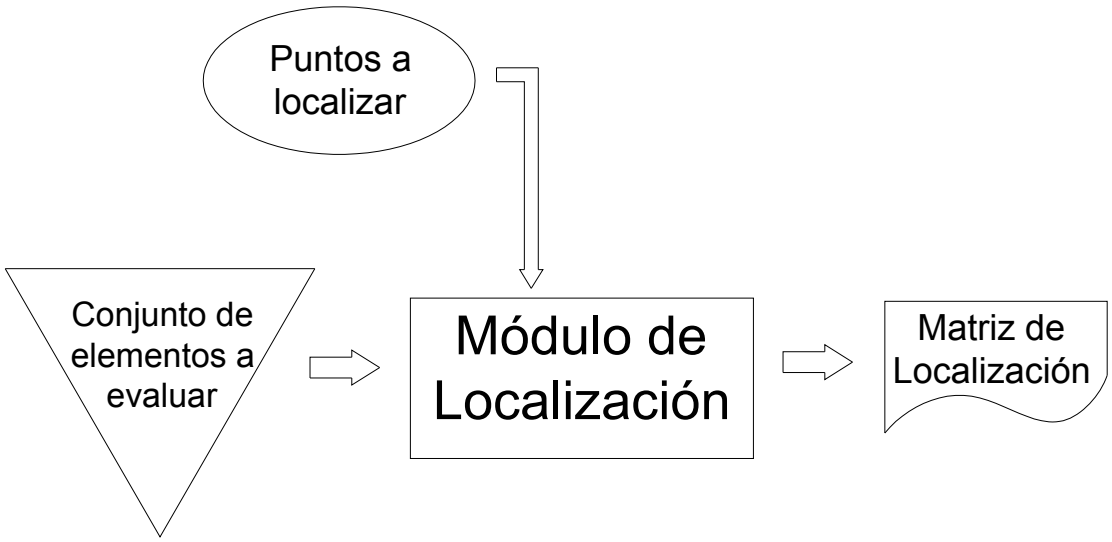


Figura III.11: Diagrama del Módulo de Localización

El Módulo de Localización culminará con la confección de un arreglo con su mismo nombre, la Matriz de Localización. La cual será un cuadro de doble entrada con dos columnas, una para los puntos (nodos y puntos de control de la GA) y otra para los elementos, y tantas filas como puntos se hayan localizado. La individualización de cada punto y elemento se realizará a través del sistema de índices globales correspondiente.

III.7. EL ACOPLAMIENTO ENTRE LOS MÓDULOS

Si bien son numerosos los algoritmos aplicables al Programa de Vinculación, el punto de partida y el objetivo que se pretende no varían. Luego, cada método de ejecución que pueda concebirse para implementar a un Módulo por separado, o al Programa de Vinculación como un conjunto, será válido siempre que arribe al resultado deseado. No obstante, no debe perderse de vista que cada caso práctico

presenta sus propias características y, algunas veces, ciertos sistemas pueden no ser aplicables o resultar ineficientes.

Es por esto que los Módulos que componen el Programa de Vinculación deberán, como se explicó, combinarse de forma tal que el código resultante cumpla con el objetivo lo más eficientemente posible. El acoplamiento entre ellos debe ser sencillo para que cada uno, de ser necesario, pueda reemplazarse como un bloque de programa sin perturbar demasiado el funcionamiento de los restantes. De este modo, cualquier modificación en los algoritmos puede implementarse sin mayores problemas.

Las especificaciones prácticas de los algoritmos serán expuestas en los capítulos siguientes.

Capítulo IV: Módulo de Ingreso: Algoritmos e Implementación

IV.1. Generalidades

En el capítulo anterior se realizó una conceptualización general del funcionamiento y objetivo del primer módulo del Programa de Vinculación. Siguiendo esos lineamientos, se expondrán aquí los mecanismos y procesos que se aplican para alcanzar el objetivo del Módulo de Ingreso.

Se vio anteriormente que las principales funciones que componen el módulo inicial son destinadas a procesar los archivos provenientes del mallador, en una palabra, son funciones de lectura. A este respecto, es notoria la importancia que cobra el entorno de implementación, MATLAB, que proporciona numerosas sentencias simples pero de acción específica que facilitan el tratamiento de archivos. No obstante, también es requerido el acceso a las ya mencionadas bibliotecas de parámetros de los elementos, que deben ser creadas de manera externa y cargadas cuando son requeridas.

En las secciones siguientes se tratarán tanto las funciones que componen al Módulo de Ingreso, como la creación y ejecución de bibliotecas de parámetros. Se presentarán también algunos conceptos de programación y procesamiento de datos que son recurrentes en este tipo de códigos.

IV.2. Ordenamiento Nodal

El ordenamiento nodal es un concepto sumamente importante en las aplicaciones que se tratan en este trabajo. Como su nombre lo indica, hace referencia a la secuencia en la que los nodos de un elemento se nombran o identifican de manera local. La secuencia a utilizar es arbitraria y puede ser definida según la avidez de quién desarrolla un código, pero es recomendable que responda a un sistema lógico, de fácil interpretación y, fundamentalmente, que se respete a lo largo del programa.

Para interpretar lo anterior, considérese el caso siguiente como ejemplo. En la Figura IV.1 se muestra una malla arbitraria compuesta por seis elementos cuadriláteros planos cuatro nodos, de la cual se aisló para el análisis el elemento Nº3.

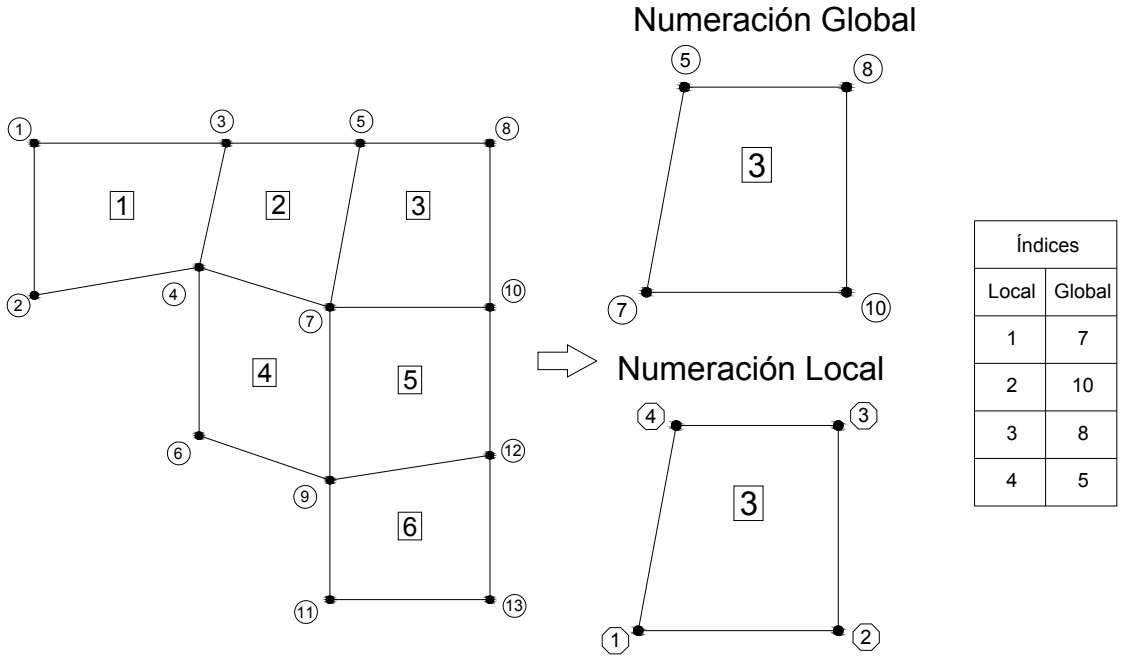


Figura IV.1: Ordenamientos Nodales Local y Global de un Elemento

La numeración global de los nodos dependerá da la aplicación y del tipo de malla con que se trate y, en general, un elemento no tendrá sus nodos con una numeración global sucesiva. Será entonces el ordenamiento local el que permitirá identificar la posición de un nodo específico del elemento y conocer su índice global. Siguiendo con el ejemplo, el elemento Nº3 posee cuatro nodos: el 5, el 8, el 7 y el 10, cuyo orden puede definirse de diversas maneras, siendo preciso escoger una. De aquí en adelante y para todo lo relativo al presente trabajo, se utilizará la numeración propuesta por T. Hughes (dada en [1]) que se corresponde con la indicada en la Figura IV.1. De este modo, se dirá, tomando el ejemplo en particular, que el nodo local 2 del elemento Nº3 posee el índice global 10.

La elección de este ordenamiento nodal proporciona varias ventajas. Entre las cuales se destacan la sencilla lógica de la numeración y la relación entre elementos de igual forma y distinto orden. Esta segunda puede visualizarse en el siguiente caso: los primeros 8 nodos de un hexaedro de segundo orden (que posee 27 nodos) son coincidentes con los 8 nodos de un hexaedro de primer orden.

A pesar de lo ventajoso que pueda parecer optar por una determinada numeración local, existe una desventaja que debe salvarse. Si bien, el ordenamiento será respetado a lo largo de todo el Programa de Vinculación, cuando se trabaja con datos externos, los índices de los nodos deben traducirse. Este es el caso de los archivos proporcionados por el mallador GMSH, el cual posee un sistema de secuenciación propio que difiere del escogido. Consecuentemente, deberá generarse

una función o un conjunto de sentencias que modifiquen dicha secuencia y la compatibilicen con la seleccionada.

La traducción de índices entre distintas tipologías puede realizarse de dos maneras. La primera consiste en identificar la lógica aplicada al ordenamiento externo, en este caso el de GMSH (presentada en [6]) y generar un vínculo aritmético entre ambas. Esto no siempre es posible y difícilmente se aplique idénticamente a todos los tipos de elementos que se utilizan.

El segundo método es generando tablas de compatibilidad entre los ordenamientos, y es el que se utiliza en el Programa de Vinculación. Estas tablas serán utilizadas por funciones especializadas para reordenar los índices. La construcción de las tablas de compatibilidad se lleva a cabo observando paralelamente los elementos de uno y otro sistema, para reasignar los índices convenientemente.

Finalmente, cada vez que se requiera el orden local de los nodos se implementará la tabla de compatibilidad y se realizará un reordenamiento previo. En el Anexo II se presentan las relaciones entre índices y las tablas de compatibilidad para algunos elementos.

IV.3. MATRICES DE PROCESAMIENTO DE DATOS

Las Matrices de Procesamiento de Datos son arreglos que poseen información sobre las mallas, especialmente sobre sus nodos y elementos, dando eficiencia al acceso a los datos. En la bibliografía especializada (por ejemplo [1]) existen numerosos tipos de Matrices de Procesamiento de Datos que cumplen diferentes funciones y son útiles en diversas aplicaciones. No obstante, para el objetivo del Programa de Vinculación sólo se utilizan dos de ellas: la primera es la Matriz de Nodos y Coordenadas" o "Arreglo de Nodos" (o bien "Nodes Array" por su nombre en inglés) y la segunda es "Matriz de Nodos y Elementos" o "Arreglo IEN" (o bien "IEN Array" por su nombre en inglés).

Cada una de las Matrices de Procesamiento de Datos se crea a partir de una función externa al Módulo de Ingreso. Estas funciones se denominan "NodesArray.m" y "IENArray.m", donde la extensión ".m" es propia del entorno de programación. Tanto "NodesArray.m" como "IENArray.m" tendrán por argumento el nombre y ubicación del archivo de la malla que se desea analizar y sus salidas serán las matrices correspondientes y otras variables que dependen de cada caso y se analizarán más adelante.

IV.3.1. EL ARREGLO DE NODOS

Conceptualmente, el Arreglo de Nodos es una matriz que almacena las coordenadas de cada uno de los nodos que componen una malla. Preferentemente se organiza en columnas para facilitar la operatoria posterior, es decir, cada fila de la matriz corresponde a una coordenada y cada columna a un nodo.

La construcción del Arreglo de Nodos se realiza a partir de la lectura directa del bloque de nodos del archivo de datos de la malla que se desea procesar. Sin embargo,

deben tomarse ciertos recaudos respecto al pre-dimensionamiento del arreglo y el escaneo del archivo.

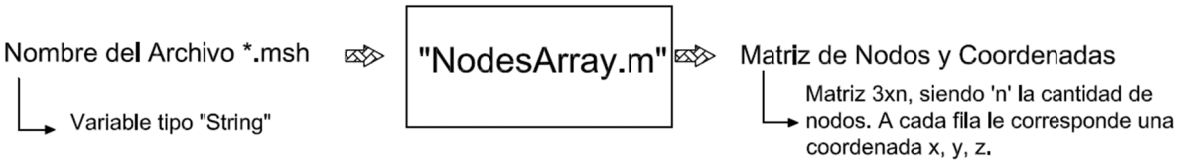


Figura IV.2: Esquema de Funcionamiento de la Función "NodesArray.m"

La función "NodesArray.m" tiene como primera sentencia la apertura en modo de lectura del archivo ".msh" de la malla. El segundo paso es posicionar el cursor en el bloque de nodos del archivo de la malla el cual, como se mostró, se titula con su línea correspondiente, "$Nodes". Una vez posicionado el cursor se procede a la extracción de la primera variable. Ésta se nombra como "NumberOfNodes", corresponde a la cantidad de nodos que posee la malla y proviene de la segunda línea del bloque de nodos del archivo. El conocimiento de la cantidad de nodos de la malla permite el pre-dimensionamiento del Arreglo de Nodos, lo que, como lo explica P. Getreur en su guía [3], incrementa sensiblemente la velocidad de ejecución de la función. Se crea entonces la matriz "NodesArray", en primera instancia, como una matriz de ceros para luego llenarse con los demás datos del archivo, teniendo tantas columnas como nodos posea la malla y una fila por cada coordenada.

El Arreglo de Nodos se completa con la extracción sucesiva de los datos del bloque de nodos. Necesariamente debe hacerse coincidir el número de nodo con el índice de la columna y cada una de las coordenadas, "x", "y", "z", con las filas correspondientes. Esto permite que la extracción de datos pueda implementarse de forma sistemática ya que si se quiere, por ejemplo, la coordenada "y" del nodo número 19 de una malla, deberá recurrirse a la columna 19 y a la fila 2 del Arreglo de Nodos.

Finalmente, es de buena práctica cerrar el archivo una vez finalizada la construcción del arreglo. Con esto se evita la superposición de lecturas y se reduce el tiempo de pre-procesamiento de otras funciones.

Número de Nodo → Cantidad de Nodos

Coordenadas	1	2	3	4	5	6	7	8	9	10	11	12	13
X	0	2	1	1	0	0	0.5	1.5	1	0.5	0.528	1.192	1
Y	0	0	1	1.5	2	1	1.75	0.5	0	0.5	1.222	1.495	0.5
Z	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura IV.3: Arreglo de Nodo para Malla de Figura III.4

IV.3.2. EL ARREGLO IEN

El Arreglo de Nodos y Elementos es una matriz que se construye con los índices globales de cada nodo y permite asociarlos con el ordenamiento local de cada elemento. En este sentido, cobra vital importancia el sistema de numeración de nodos al cual se hizo referencia en la sección anterior.

El Arreglo IEN se organiza de manera tal de asignar una columna a cada elemento de la malla, y en esta columna almacenar, según el ordenamiento local, los índices globales de cada uno de los nodos que componen el elemento. Considérese, a modo de ejemplo, el caso de la Figura IV.4.

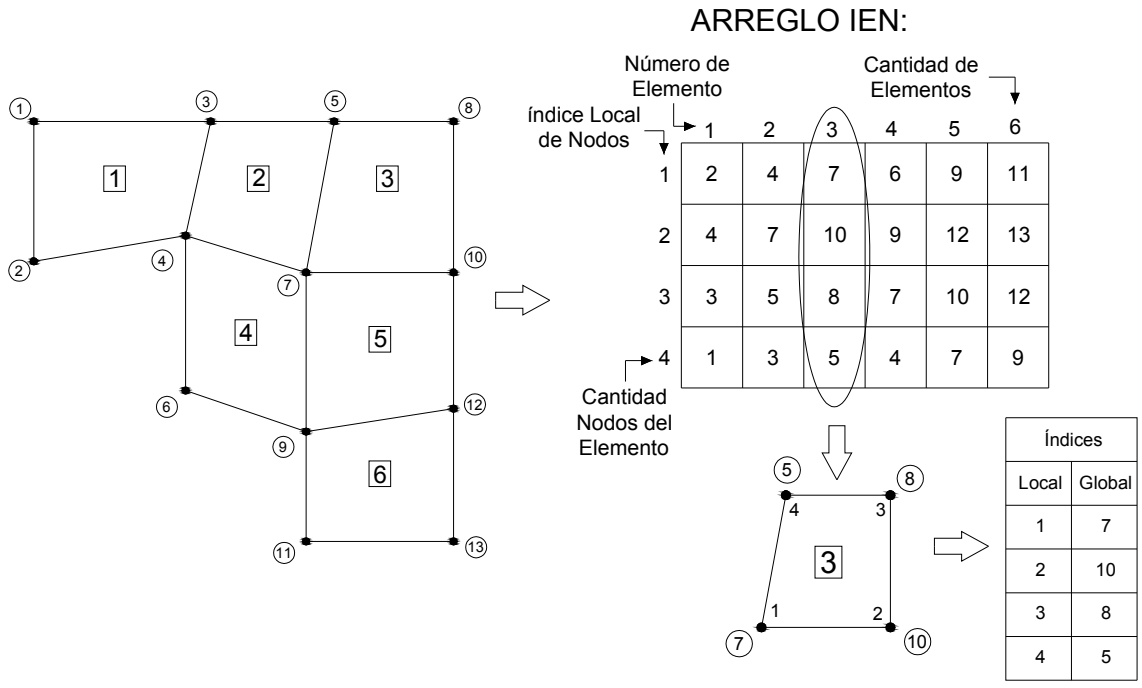


Figura IV.4: Arreglo IEN para una Malla de Ejemplo con Indicación de Ordenamiento Nodal

La función "IENArray.m" encargada de confeccionar el Arreglo de Nodos y Elementos describe un proceso similar a su equivalente para el Arreglo de Nodos. Sin embargo, existen algunas variantes que serán advertidas a continuación.

En primer lugar, se inicia con la apertura del archivo de forma tradicional y se emprende un ciclo de búsqueda de la línea de título. Como se habrá notado, la información necesaria para la construcción del Arreglo IEN se encuentra en el bloque de elementos del archivo ".msh", por lo que la línea de título de interés es la que corresponde a este bloque ("$Elements"). A pesar de esto y con el objetivo de reducir el tiempo de ejecución, el ciclo de búsqueda será idéntico al utilizado en "NodesArray.m", es decir, se identificará la línea "$Nodes". Seguidamente, se extraerá la cantidad de nodos que tiene la malla y se desplazará el cursor tantas líneas como

este número lo indique adicionando una que corresponderá a la línea de título de cierre. Este proceso posicionará finalmente el cursor en el bloque deseado.

El segundo paso es la extracción de la cantidad de elementos de la malla, creando la variable "NumberOfElements". Con este dato, se conoce cuál será el número de columnas que tendrá el Arreglo IEN, restando sólo la determinación del número de filas.

El número de filas del Arreglo IEN será igual a la cantidad de nodos que posean los elementos de la malla. Se entiende que una malla estará compuesta por un solo tipo de elemento, ya que no se incluirán elementos de transición. La lectura de la línea del primer elemento proporcionará al código el tipo de éste y se creará la variable "ElementType". Esta variable será transferida a una biblioteca de parámetros que identificará el tipo de elementos y devolverá su cantidad de nodos, "ElementNodesNumber". Los dos valores obtenidos permitirán dimensionar el Arreglo IEN, cuyo tamaño será: "ElementNodesNumber"x"NumberOfElements".

Conociendo las dimensiones, el arreglo se crea con ceros y se procede a su llenado. Para la extracción y almacenamiento de los índices de los nodos debe considerarse el ordenamiento nodal. Como se explicó, se optó por utilizar una secuencia local para los nodos diferente a la proporcionada por el mallador, luego, a medida que los índices son extraídos deben ser procesados y reordenados según la tabla de compatibilidad correspondiente.

Cuando finalice el ciclo en todas las líneas de los elementos el Arreglo IEN estará completo. Como detalle, se agrega que la función "IENArray.m" no sólo devuelve al código principal la matriz confeccionada, sino que también retorna la variable "ElementType", ya que será requerida posteriormente. Por último y siguiendo con la buena práctica, se cierra el archivo de la malla.

IV.4. BIBLIOTECAS DE PARÁMETROS

Las bibliotecas de parámetros se crean como funciones externas al código principal del programa y su principal función es almacenar datos que son de uso recurrente. El objetivo del uso de estas estructuras es, en primer lugar, proporcionar una fuente ordenada y eficiente de variables y, en segundo lugar, reducir el tamaño de los códigos evitando incluir líneas de programación que sólo almacenan información.

El funcionamiento interno de las bibliotecas se organiza con un algoritmo tipo menú. Una variable proporcionada por el código principal ingresa a la función como argumento y es evaluada comparativamente contra un listado de posibles valores que pueda tomar. Dependiendo de la coincidencia que ocurra, la función automáticamente devuelve al código principal los datos asociados al argumento como una variable de retorno. Esquemáticamente, el proceso se representa en la Figura IV.5.

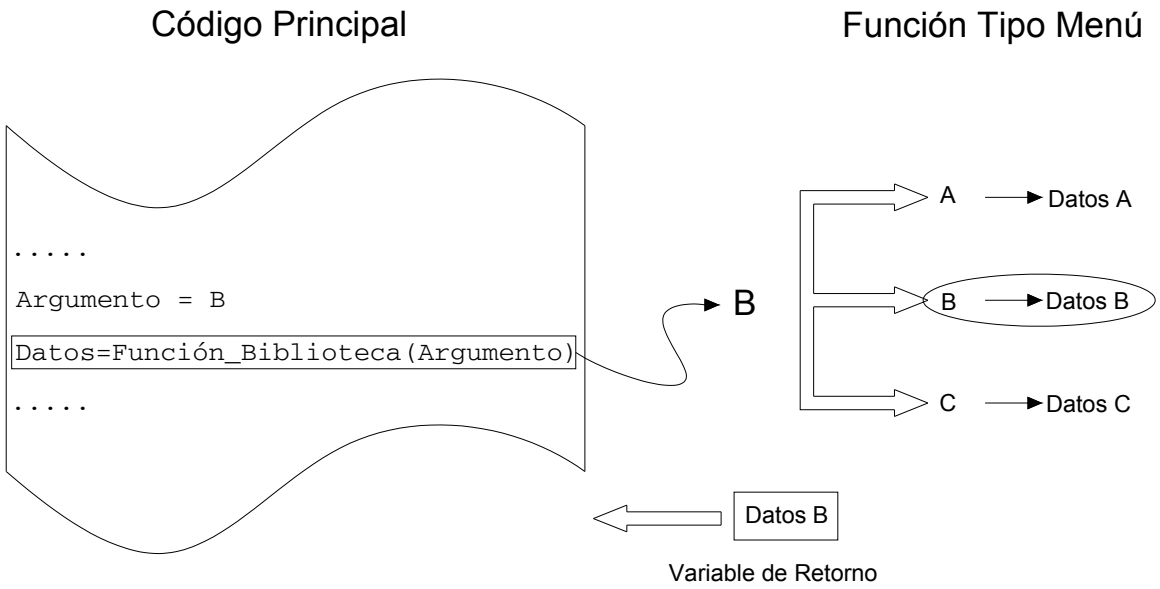


Figura IV.5: Esquema de Funcionamiento de una Biblioteca de Parámetros con una Función Tipo Menú

Las variables de retorno pueden ser de diversa naturaleza, dependiendo del tipo de datos que se maneje en la función. En el Módulo de Ingreso son tres las bibliotecas de parámetros que se utilizan, y proporcionan valores que serán requeridos a lo largo del código principal. Estas son: La Biblioteca de Tipos de Elementos, La Biblioteca de Funciones de Forma y La Biblioteca de Funciones de Forma Derivadas.

IV.4.1. BIBLIOTECA DE TIPOS DE ELEMENTOS

Esta Biblioteca es la encargada de traducir la codificación que utiliza el mallador respecto al tipo de elementos. Es necesaria principalmente en la confección del Arreglo IEN y reviste importancia porque el sistema de tipificación de elementos de GMSH es estrictamente arbitrario y no existe otro mecanismo para reconocer a qué clase se hace referencia.

La función recibe como argumento una variable con el código de identificación (dados en la Figura III.7) y devuelve la cantidad de nodos que posee ese elemento. Internamente trabaja con un menú "switch" y veintiún casos posibles de identificación.

IV.4.2. BIBLIOTECA DE FUNCIONES DE FORMA

La función que almacena la Biblioteca de Funciones de Forma tiene por objetivo proporcionar al código principal un vector como el que se utiliza en la expresión (II.5.1.12). Este vector permite generar el mapeo de los elementos correspondiente de manera sencilla y es necesario a lo largo Programa de Vinculación.

En la sección II.7 se propuso un método de implementación sistemática para la obtención de las funciones de forma, consistente en el uso de los Polinomios de Lagrange. De hecho, programar este proceso en una función externa que se ejecute cuando las funciones de forma sean precisadas, sería una solución viable. Sin embargo, aplicar una función de este tipo demanda un tiempo de ejecución que entorpecería la corrida del código principal y puede ser evitado.

Las funciones de forma de los elementos presentan una característica importante que resulta de suma utilidad y hace propicio el uso de la biblioteca que las contiene: sólo dependen del tipo de elemento con el que se trabaje. En este sentido, sólo basta con conocer con qué elementos tratará el Programa de Vinculación (los provenientes de la ME) para poder calcular el vector de funciones de forma. Así, se recurre a la implementación de los Polinomios de Lagrange para confeccionar, por única vez, un listado (o biblioteca) de los vectores para cada tipo de elemento. Luego, este listado, que se condice con el presentado en el Anexo I, se almacena en una función tipo menú, resultando en la función de la Biblioteca de Funciones de Forma. Internamente la función recibe como argumento el tipo de elemento y, como es de esperarse, retorna el vector de funciones de forma al código principal.

IV.4.3. La Biblioteca de Funciones de Forma Derivadas

Las Funciones de Forma Derivadas son utilizadas en los procesos de localización del Tercer Módulo y son un factor para construir la matriz jacobiana de la transformación afín o mapeo de los elementos. Las Funciones de Forma trabajan en la forma vista en la expresión (II.5.1.12) y, por ende, sus derivadas componen una matriz que surge de aplicar el gradiente al vector $\boldsymbol{N_a}$. Siendo [N′] la matriz de Funciones de Forma Derivadas, se tiene, para el caso de tres dimensiones,

$$[\mathrm{N}'] = \nabla \boldsymbol{N_a} = \begin{bmatrix} \nabla N_1 \\ \vdots \\ \nabla N_a \end{bmatrix} = \begin{bmatrix} \dfrac{\partial N_1}{\partial \xi} & \dfrac{\partial N_1}{\partial \eta} & \dfrac{\partial N_1}{\partial \zeta} \\ \vdots & \vdots & \vdots \\ \dfrac{\partial N_a}{\partial \xi} & \dfrac{\partial N_a}{\partial \eta} & \dfrac{\partial N_a}{\partial \zeta} \end{bmatrix}. \qquad \text{(IV.4.3.1)}$$

Nuevamente, aquí surge la posibilidad de implementar una función externa que efectúe la derivación simbólica de cada función de forma y construya la matriz [N′] cada vez que sea necesario. No obstante, y con el mismo razonamiento aplicado a la Biblioteca de Funciones de Forma, es conveniente, desde el punto de vista del tiempo de ejecución, contar con una función tipo menú que proporcione automáticamente la matriz para cada tipo de elemento.

La construcción de esta función se realiza con un código de derivación simbólica. El proceso consiste en tomar por separado cada función de forma, efectuar las derivaciones parciales respecto de cada coordenada y confeccionar la matriz correspondiente para cada tipo de elemento. Se obtiene así una función que toma por

argumento el tipo de elemento y retorna la matriz de funciones de forma derivadas adecuada.

IV.5. RECEPCIÓN Y PROCESAMIENTO INICIAL DE DATOS

El Programa de Vinculación está pensado como una función autónoma y debe ser capaz de generar por sí misma los datos y variables necesarios para su funcionamiento. El encargado de esta tarea, como se vio hasta aquí, es el Módulo de Ingreso. En este sentido, los únicos argumentos necesarios para la ejecución del código principal serán, de acuerdo con el esquema de la Figura III.9, los archivos de texto de las mallas a procesar. Así, el nombre y la ubicación de los archivos de la ME y la GA se introducen como variables tipo "string", para luego ser tratados en el Módulo de Ingreso.

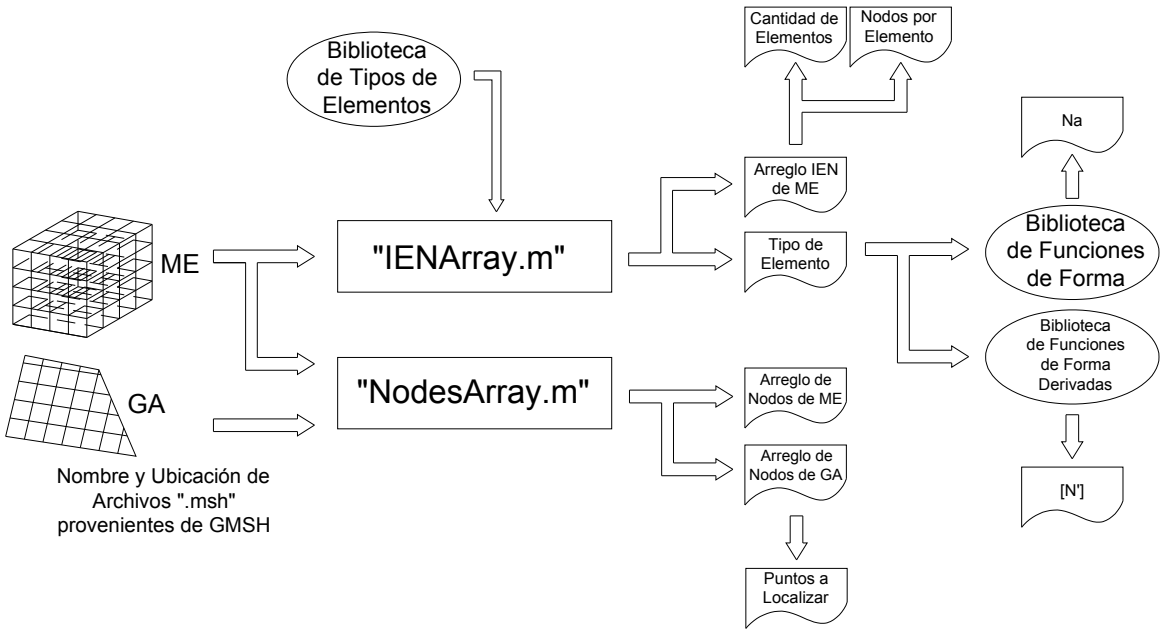


Figura IV.6: Esquema de Funcionamiento General del Módulo de Ingreso

La primera acción que realiza el Módulo de Ingreso es interiorizar los argumentos externos que le son transferidos como variables, asignándoles un nombre que sea sencillo de manejar y reconocer dentro del código. Una vez hecho esto, se da paso a las funciones que construyen las Matrices de Procesamiento de Datos. La ME será argumento tanto de "NodesArray.m" para la Matriz de Nodos y Coordenadas, como de "IENArray.m" para el Arreglo IEN. Ambos arreglos de la malla de FEM son almacenados como variables internas con nombres alusivos, al igual que el tipo de elementos (variable "ElementType") proveniente de la función del Arreglo IEN. Para la GA, en cambio, sólo se ejecuta la función "NodesArray.m" para obtener las

coordenadas de sus puntos. En aquellos casos en los que fuese necesario discriminar entre nodos y puntos de control de la GA, se requerirá un pre-procesamiento adicional.

Inmediatamente después, la variable "ElementType" (tipo de elemento) es utilizada para llamar a las bibliotecas de Funciones de Forma y de Funciones de Forma Derivadas. Estas bibliotecas entregan sus variables de retorno, el vector de Funciones de Forma y la matriz de Funciones de Forma Derivadas respectivamente, que se almacenan convenientemente.

Por último, se extraen algunos valores importantes para el progreso del código principal, a saber: la cantidad de nodos por elemento, la cantidad de elementos, y la cantidad de puntos a localizar. Estas tres variables se condicen con, las dos primeras, las dimensiones del Arreglo IEN (filas y columnas, respectivamente), y la última con la cantidad de filas del Arreglo de Nodos de la GA. Adicionalmente y conociendo estos datos, se pre-dimensiona la Matriz de Localización, como un arreglo de ceros con tantas filas como puntos haya que localizar y dos columnas, una para el índice de cada punto y otra para el índice del elemento que lo contiene.

Capítulo V: Módulo de Selección. Algoritmos e Implementación

V.1. Generalidades

La finalidad que persigue el Módulo de Selección es la de reducir, de una forma relativamente sencilla, el número de elementos que serán evaluados por los algoritmos de localización. Como se verá más adelante, este módulo no es estrictamente necesario desde el punto de vista operativo, pero su implementación genera avances sorprendentes en la eficiencia del Programa de Vinculación.

El segundo módulo es quizás el más versátil de los tres debido a que las consideraciones que pueden efectuarse a la hora de seleccionar o crear un conjunto de elementos son numerosas. Recuérdese que se priorizan los aspectos geométricos y topológicos de la malla del FEM, por lo que la preselección de elementos puede encararse desde varios puntos de vista. Aquí se presentarán cinco alternativas que van desde las más sencillas, y quizás por eso menos eficientes, hasta algunos de complejidad relativamente elevada que han mostrado resultados alentadores.

Nuevamente, rige el criterio de la aplicabilidad práctica, es decir que la eficacia de algunos mecanismos está sujeta a consideraciones sensatas sobre la ME. De ahí que surge una decisión de compromiso y la necesidad de evaluar previamente cada caso particular. Sin embargo, los métodos más simples, como se indicará oportunamente, han demostrado ser de aplicación universal, en desmedro de la velocidad de ejecución.

En la mayoría de los casos, la variable de salida del Módulo de Selección será una matriz, la cual será transferida al módulo siguiente. La Matriz de Selección asigna una fila a cada uno de los puntos a localizar y enlista, de forma ordenada, el conjunto de elementos que fueron seleccionados para dicho punto.

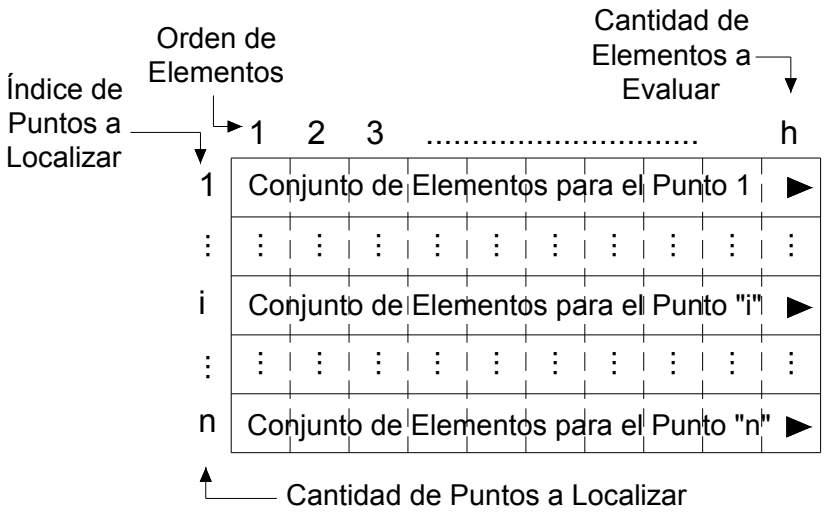


Figura V.1: Matriz de Selección Genérica

En las secciones siguientes se detallan los algoritmos de selección desarrollados, describiendo sus principios y procedimientos. Se considera que, para el funcionamiento de todos ellos, el Módulo de Ingreso ya fue ejecutado y se cuentan con todas las variables provenientes de éste.

V.2. EL ALGORITMO DE LISTA

El método o algoritmo de Lista es el más trivial de todos y equivale a prácticamente prescindir del Módulo de Selección. Este algoritmo utiliza una lista de los elementos según sus índices globales para evaluarlos uno a uno, sin importar qué punto se esté localizando.

En este caso, construir la Matriz de Selección como una variable no resulta conveniente puesto que, todas sus filas serían idénticas. Sin embargo, para explicar más claramente este sistema se presenta en la Figura V.2 cómo se vería la Matriz de Selección si fuese construida.

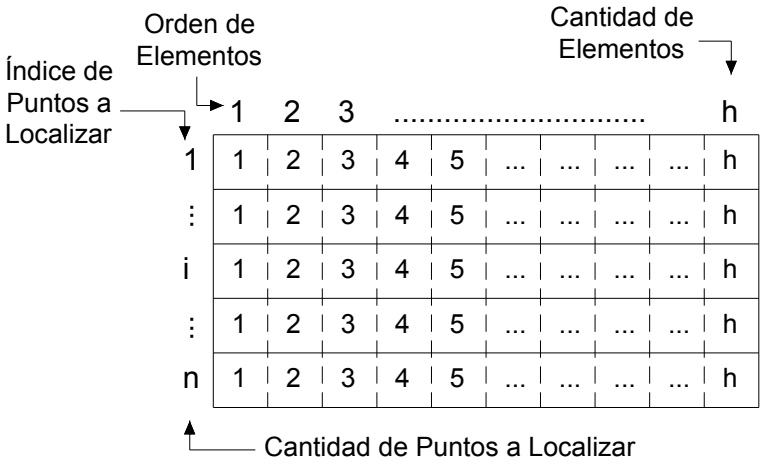


Figura V.2: Matriz de Selección para el Algoritmo de Lista

Como se dijo, la Matriz de Selección no es utilizada aquí, en cambio, se incluye un ciclo que engloba al Módulo de Localización. El bucle se ejecutará una vez para cada punto a localizar y tendrá tantos pasos como elementos tenga la ME. Cada paso se identificará con un índice igual al del elemento que se analizará y si la localización es exitosa el ciclo será interrumpido y se continuará con el punto siguiente.

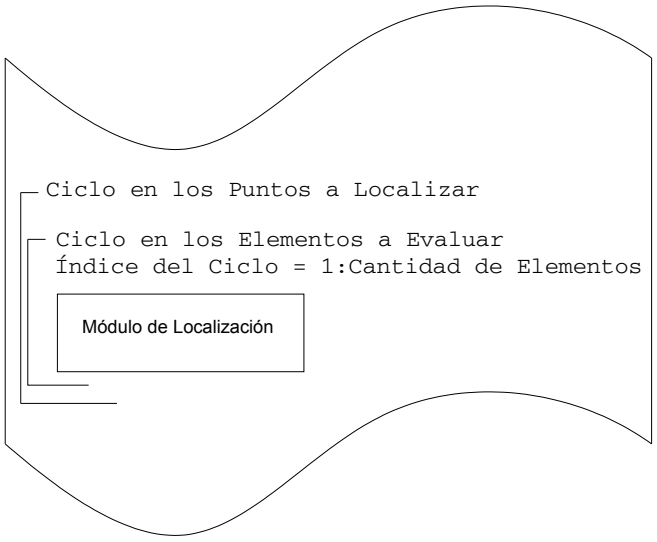


Figura V.3: Esquema de Código para Algoritmo de Lista

Ahora bien, este método carece de eficiencia pero es uno de los que evalúa indiscriminadamente todos los elementos de la ME, por lo que garantiza que todos los puntos serán eventualmente localizados.

V.3. EL ALGORITMO DE LISTA ALEATORIA

El método de Lista Aleatoria es una sutil modificación al caso anterior. Surge de incorporar el concepto del orden de los elementos seleccionados como una forma de reducir el tiempo de ejecución, como se vio en la sección III.5.

En este sentido, iniciar la lista en el elemento número 1 y continuar de forma ordenada para todos los puntos no resulta del todo práctico. Por lo tanto, dado un punto a localizar quizás resulte provechoso designar azarosamente un elemento que podría contenerlo, y así hasta evaluarlos a todos. De esta manera, todos los elementos serán evaluados virtualmente la misma cantidad de veces y no tendrán prioridad los que posean índices bajos.

Se proponen aquí dos modos de implementar la Lista Aleatoria, que se diferencian en el momento en el que se efectúa el reordenamiento azaroso de los elementos a evaluar. Estos dos casos son: Lista Aleatoria Simple y Lista Aleatoria Múltiple.

V.3.1. LISTA ALEATORIA SIMPLE

En este caso, existe mayor similitud con el Algoritmo de la Lista ya que, si se construyera la Matriz de Selección, todas sus filas serían idénticas, como muestra la Figura V.4. Nuevamente, sin embargo, la confección de la matriz no es necesaria debido a que la implementación del algoritmo puede efectuarse de un modo más sencillo.

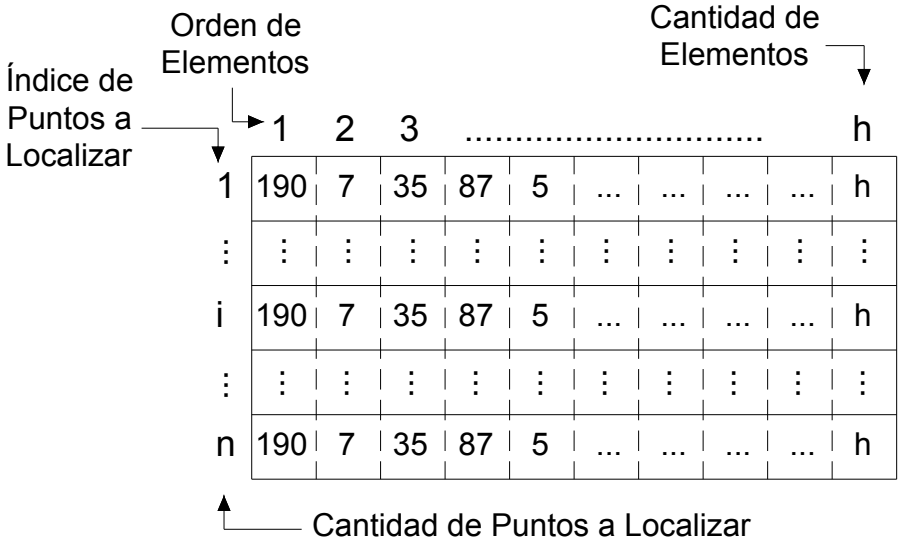


Figura V.4: Matriz de Selección para el Algoritmo de Lista Aleatoria Simple

Este método consiste en reordenar los elementos una única vez con una función específica del entorno de programación. Luego, los elementos serán evaluados en su nuevo orden para todos los puntos por igual.

El mecanismo de implementación consiste en guardar la lista aleatoria en un vector y, de nuevo, programar un ciclo que englobe al Módulo de Localización. El

índice de paso de este ciclo se corresponderá con el índice de posición del vector que contiene la lista aleatoria y, mediante una función de extracción, se evaluarán los elementos en el orden en que la lista lo indique.

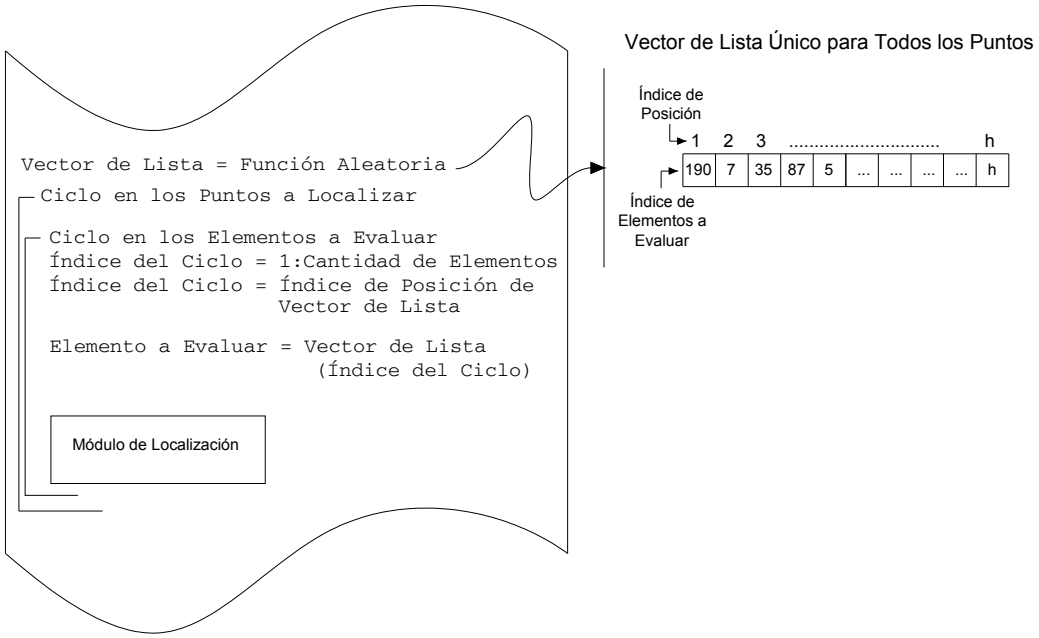


Figura V.5: Esquema de Código para Algoritmo de Lista Aleatoria Simple

V.3.2. LISTA ALEATORIA MÚLTIPLE

Esta variante del método de la lista aleatoria presenta la particularidad de generar un ordenamiento azaroso de elementos distinto para cada punto que se desee localizar. Aquí tampoco es necesario implementar la Matriz de Selección, pero, de hacerlo, sus filas no serían iguales entre sí, como lo muestra la Figura V.6.

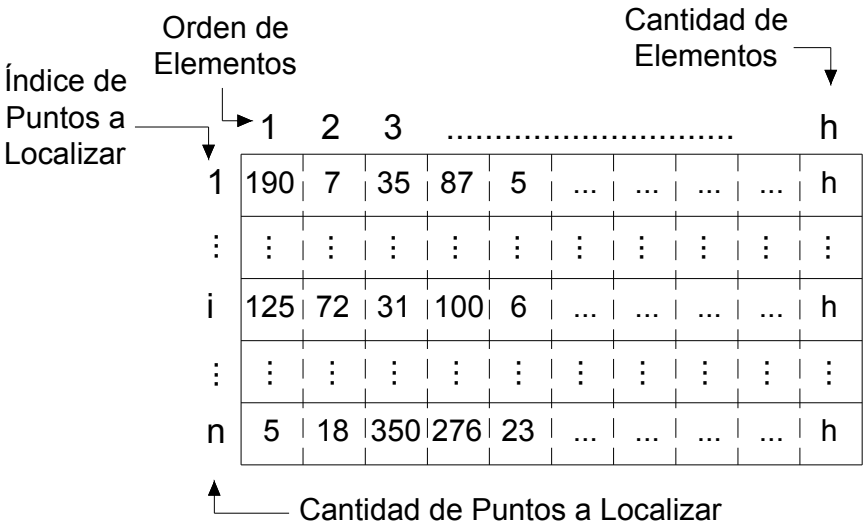


Figura V.6: Matriz de Selección para Algoritmo de Lista Aleatoria Múltiple

Para llevar a cabo el algoritmo de la Lista Aleatoria Múltiple la función que genera el vector de la lista debe ser incluida dentro del ciclo en los puntos a localizar. De este modo, cada vez que se tome un punto nuevo, se generará un nuevo orden de elementos.

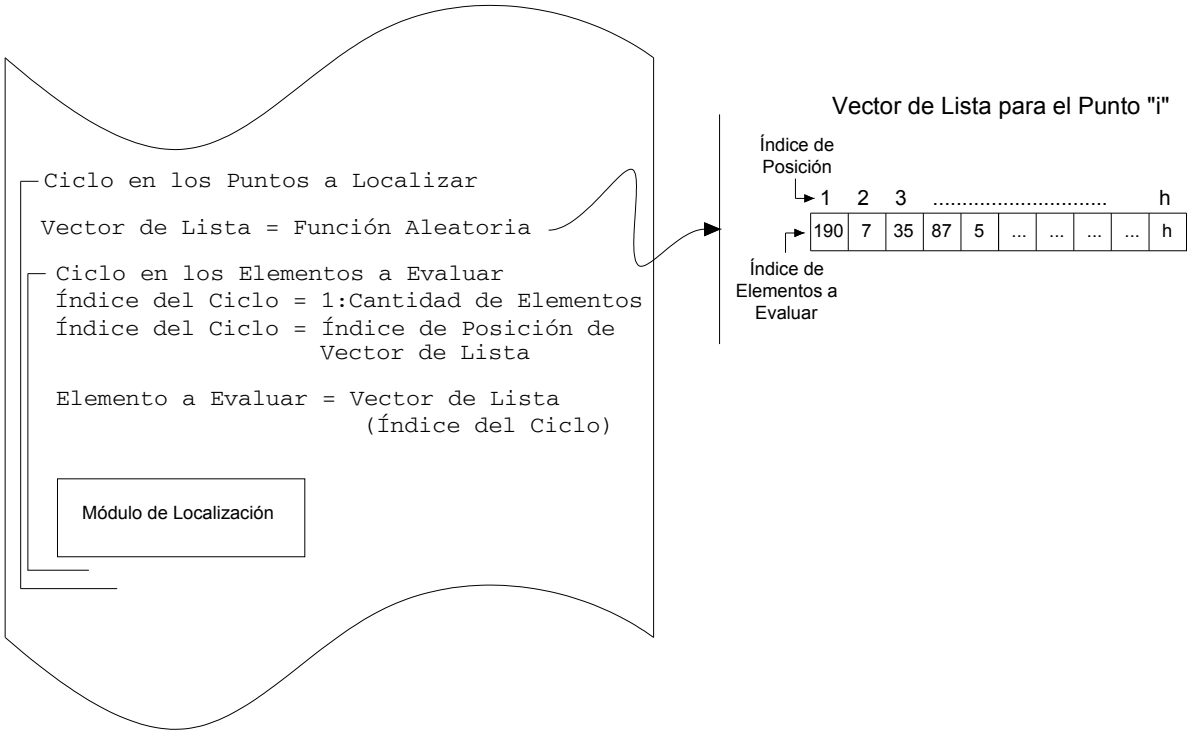


Figura V.7: Esquema de Código para Algoritmo de Lista Aleatoria Múltiple

V.4. ALGORITMO DE CENTROS DE MASA

El algoritmo de Centros de Masa concibe a los elementos como una disposición de masas puntuales de igual valor ubicadas sobre cada uno de sus nodos. Esta concepción permite calcular para cada elemento un punto designado como centro de masa, cuyo vector de coordenadas será

$$\boldsymbol{X_{CM}} = \frac{\sum_{i=1}^{a} \boldsymbol{x_i}}{a}. \tag{V.4.1}$$

Con las siguientes componentes

$$X_{CM} = \frac{\sum_{i=1}^{a} x_i}{a} \tag{V.4.2}$$

$$Y_{CM} = \frac{\sum_{i=1}^{a} y_i}{a} \tag{V.4.3}$$

$$Z_{CM} = \frac{\sum_{i=1}^{a} z_i}{a} \tag{V.4.4}$$

Siendo "a" la cantidad de nodos del elemento.

Es, entonces, en este concepto en el que se basa el criterio de selección del algoritmo de Centros de Masa.

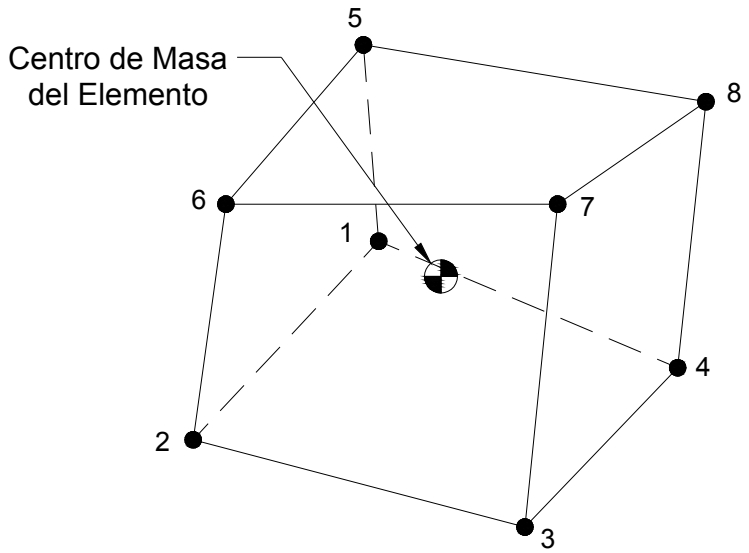


Figura V.8: Centro de Masa de un Elemento

El procedimiento consiste en calcular los centros de masa de todos los elementos de la ME para luego calcular la distancia entre cada uno de ellos y el punto que se desea localizar. Hecho esto, se ordenan de manera ascendente las distancias calculadas y se seleccionan las nueve primeras. De este modo, los elementos que conforman el conjunto son aquellos cuyo centro de masa se encuentra más próximo al punto en análisis.

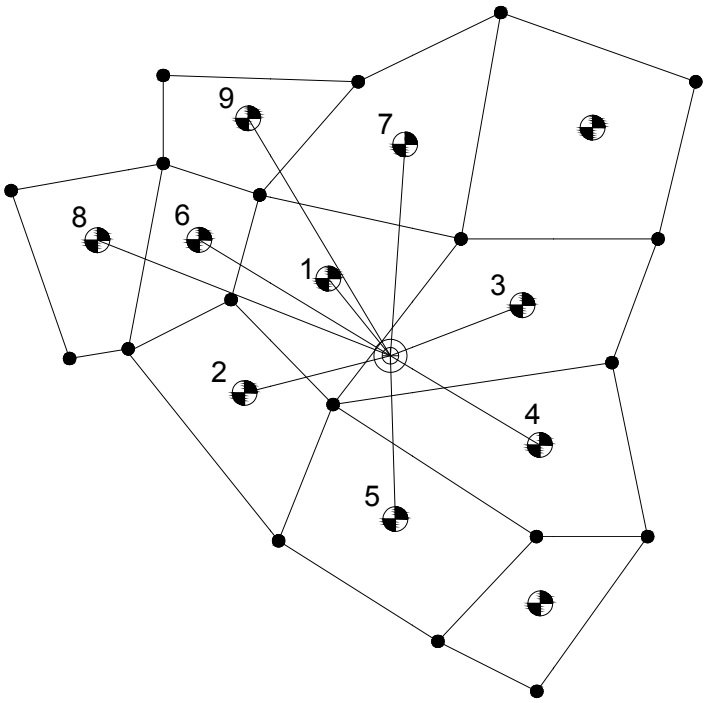


Figura V.9: Nueve Centros de Masa Más Próximos a un Punto

Lo antes descrito debe repetirse para todos los puntos provenientes de la GA, lo que dará por resultado la Matriz de Selección. En este caso, la matriz contará sólo con nueve columnas y tantas filas como puntos deban ser localizados.

Surge una interrogante respecto de la cantidad de elementos seleccionados, ya que optar por considerar nueve elementos puede resultar arbitrario. De hecho, esta decisión se basa en considerar un nodo que se ubica en un vértice de un elemento hexaedro. Dicho nodo se encontrará rodeado por, en general, ocho elementos, como muestra la figura.

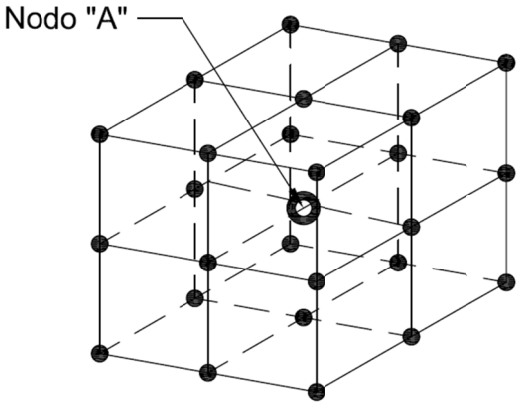


Figura V.10: Ocho Elementos que Rodean al Nodo "A"

Luego, si un punto se localiza sobre el nodo "A", tendrá a su alrededor ocho elementos y, el noveno se adiciona para considerar las posibles variantes o deformaciones que pudieran presentarse. Ciertamente, el número de elementos elegidos puede incrementarse o disminuirse según el criterio que se aplique. No obstante, considerar nueve resulta un intermedio entre un juicio conservador y la intención de reducir el número de evaluaciones.

V.4.1. IMPLEMENTACIÓN

La implementación del algoritmo de Centros de Masa se efectúa siguiendo los lineamientos antes explicados. El primer paso es pre-dimensionar una matriz que contendrá las coordenadas de los cetros de masa de todos los elementos de la ME. Como se acostumbra, el pre-dimensionamiento se realiza con ceros para luego cargar los valores necesarios. La matriz $[XCM]$ tendrá tres filas, una para cada coordenada y tantas columnas como elementos tenga la ME. Se tiene, entonces,

$$[XCM] = \begin{bmatrix} X_{CM_1} & \cdots & X_{CM_i} & \cdots & X_{CM_n} \\ Y_{CM_1} & \cdots & Y_{CM_i} & \cdots & Y_{CM_n} \\ Z_{CM_1} & \cdots & Z_{CM_i} & \cdots & Z_{CM_n} \end{bmatrix}. \qquad \text{(V.4.1.1)}$$

Donde "i" representa un elemento genérico y "n" es el número de elementos de la ME.

Otra herramienta útil en prácticamente todos los códigos del Programa de Vinculación y que se confecciona aquí, es la matriz $[Xa]$. Esta matriz contiene valores que son diferentes cada vez que se analiza un elemento y que corresponden a las coordenadas de cada uno de sus nodos, ordenadas en columnas. Se expresa

$$[Xa] = \begin{bmatrix} x_1 & \cdots & x_a \\ y_1 & \cdots & y_a \\ z_1 & \cdots & z_a \end{bmatrix}. \qquad \text{(V.4.1.2)}$$

Donde "a" indica la cantidad de nodos del elemento.

Para construir la matriz $[Xa]$ se extrae la columna del Arreglo IEN correspondiente al elemento sobre el cual se trabaja en ese momento. Luego, con un ciclo con tantos pasos como nodos tenga el elemento se extraen, del Arreglo de Nodos y Coordenadas, las coordenadas de cada uno de los nodos del elemento para guardarlas en la matriz $[Xa]$. La Figura V.11 esquematiza el proceso.

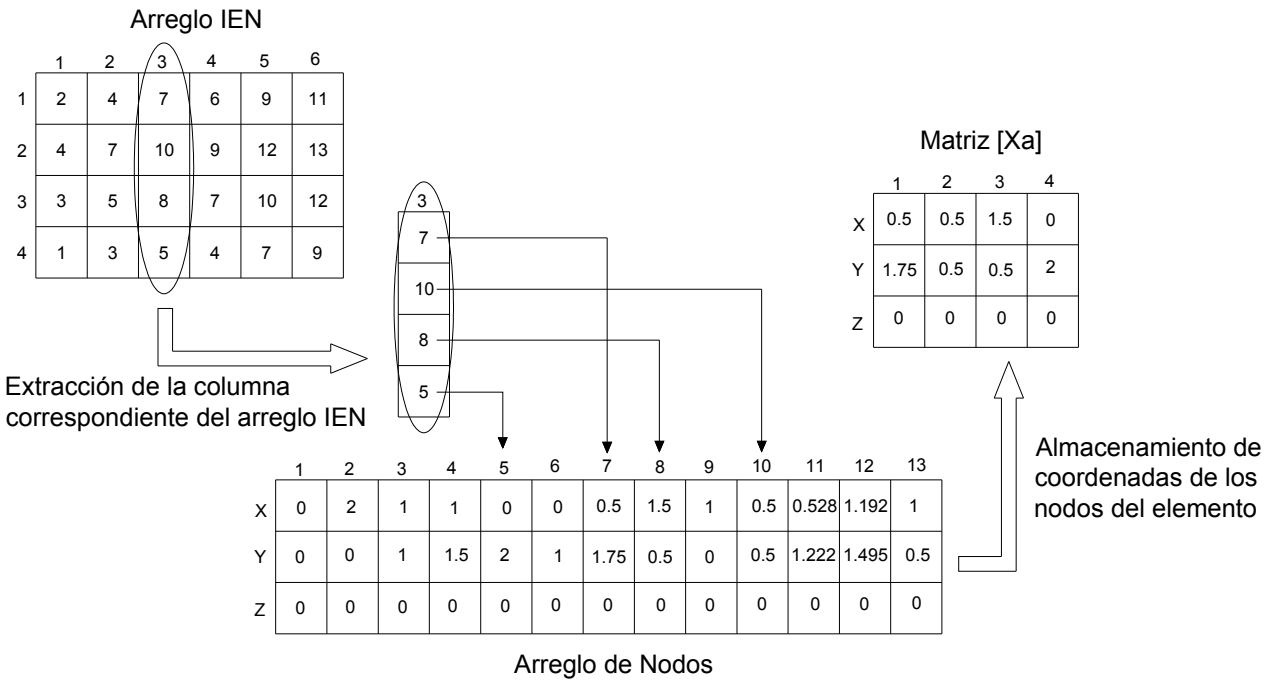


Figura V.11: Esquema para el Armado de la Matriz [Xa]

Los valores de la matriz $[XCM]$ serán calculados, entonces, con un ciclo en los elementos de la ME que en cada uno de sus pasos calculará la matriz $[Xa]$ para el elemento correspondiente. Además, habiendo obtenido $[Xa]$, se calcula, en el mismo paso del ciclo, el vector $\boldsymbol{Xcm}$ que contiene las medias aritméticas de las coordenadas de los nodos del elemento. Siendo

$$\boldsymbol{Xcm} = \frac{1}{a}\begin{bmatrix} \sum_{j=1}^{a} x_j \\ \sum_{j=1}^{a} y_j \\ \sum_{j=1}^{a} z_j \end{bmatrix} = \begin{bmatrix} X_{CM} \\ Y_{CM} \\ Z_{CM} \end{bmatrix}. \qquad \text{(V.4.1.3)}$$

Donde "a" nuevamente indica la cantidad de nodos del elemento.

Nótese que el vector $\boldsymbol{Xcm}$ se construye con el promedio de las filas de la matriz $[Xa]$ y coincide con cada columna de la matriz $[XCM]$. Luego, para generar $[XCM]$, simplemente se almacenan los vectores de cada paso del ciclo como columnas de la matriz.

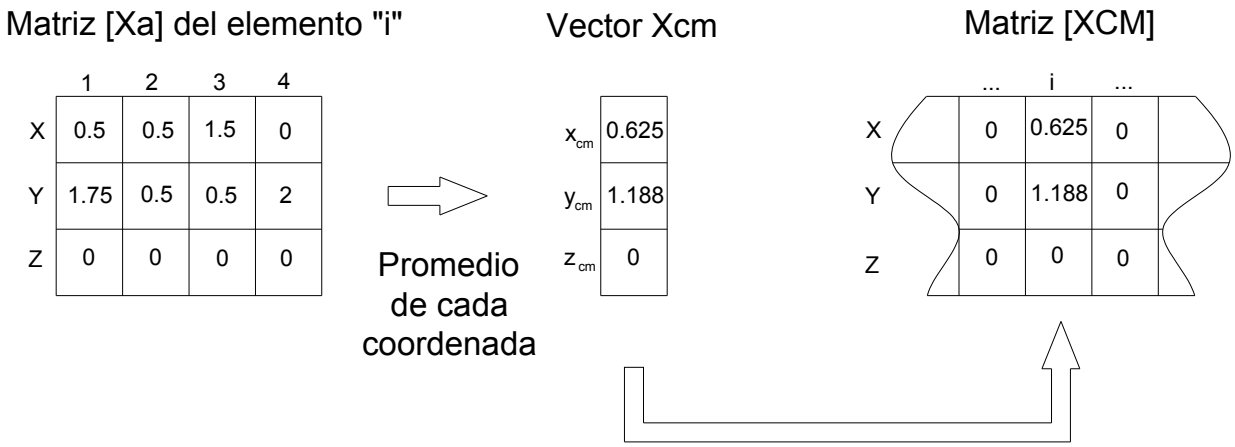


Figura V.12: Esquema para el Armado de la Matriz [XCM]

Una vez que se obtuvo la matriz de coordenadas de los centros de masa de los elementos, es necesario calcular la distancia entre cada centro de masa y el punto que se desea localizar, luego ordenar dichas distancias y seleccionar las nueve menores. Para ello, se crea un ciclo en los puntos a localizar que ejecuta, para cada uno de ellos, el proceso que se describe a continuación.

Primeramente, se genera el vector columna $\boldsymbol{Xp}$ que contiene las coordenadas del punto con el que se trata. Se crea también un vector fila auxiliar, $\boldsymbol{Ones}$, que posee tantas columnas como elementos tiene la ME y todos su valores son iguales a uno. Es decir,

$$\boldsymbol{Xp} = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}, \tag{V.4.1.4}$$

$$\boldsymbol{Ones} = [1 \quad 1 \quad \cdots \quad 1]. \tag{V.4.1.5}$$

Luego, con los vectores anteriores, se calcula el Producto Tensorial o de Kronecker, para obtener la matriz $[Kr]$, según,

$$[Kr] = \boldsymbol{Xp} \otimes \boldsymbol{Ones} = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} \otimes [1 \quad 1 \quad \cdots \quad 1] = \begin{bmatrix} x_p & x_p & \cdots & x_p \\ y_p & y_p & \cdots & y_p \\ z_p & z_p & \cdots & z_p \end{bmatrix}. \tag{V.4.1.6}$$

La construcción de la matriz $[Kr]$ facilita sensiblemente el cálculo de las distancias entre el punto a localizar y los distintos centros de masa.

El paso siguiente es calcular la matriz $[CVL]$ que contendrá las componentes de los vectores libres que unen el punto a localizar y cada uno de los centros de masa. El cálculo ilustra la importancia de $[Kr]$ y se realiza como sigue,

$$[CVL] = [\boldsymbol{CVL_1} \quad \cdots \quad \boldsymbol{CVL_n}] = [Kr] - [XCM]. \tag{V.4.1.7}$$

Finalmente, las distancias que se pretenden serán las normas euclidianas de las columnas de $[CVL]$ consideradas como vectores. Así se construye el vector de distancias para un punto dado,

$$\boldsymbol{D} = [\|\boldsymbol{CVL_1}\| \quad \cdots \quad \|\boldsymbol{CVL_n}\|] = [d_1 \quad \cdots \quad d_n]. \qquad \text{(V.4.1.8)}$$

Con el vector $\boldsymbol{D}$ calculado, se procede a ordenar los valores de distancias obtenidos para seleccionar los nueve menores. No obstante, no debe perderse de vista que el objetivo del procedimiento no son los valores d_i (con i $= 1, 2, \ldots,$ n) sino los índices de los elementos a los que estos valores hacen referencia. Es decir, d_1 hace referencia al elemento 1, d_2, al 2, y así siguiendo. Luego, cuando se realice el ordenamiento, estos índices no deben perderse, debido a que los d_i (con i $= 1, 2, \ldots,$ n) son en realidad números y no existe forma alguna de relacionarlos con los elementos más que con el índice de posición dado por el vector $\boldsymbol{D}$.

El proceso de ordenamiento y selección se realiza con dos ciclos anidados. El primero de ellos consta de nueve pasos en cada uno de los cuales se obtendrá el índice de un elemento a ser evaluado. El segundo, en cambio, recorrerá las posiciones del vector $\boldsymbol{D}$, con tantos pasos como elementos tenga la ME. El ciclo interno con la ayuda de una función condicional selecciona el menor d_i, almacena su índice en una variable y descarta dicho d_i del vector $\boldsymbol{D}$ sin alterar los índices de los restantes. La variable que contiene el índice trasciende el ciclo interno y se almacena en el primer lugar de la fila correspondiente de la Matriz de Selección. Este proceso continuará hasta completar los nueve pasos. La Figura V.13 describe esquemáticamente cómo se programa el procedimiento anterior.

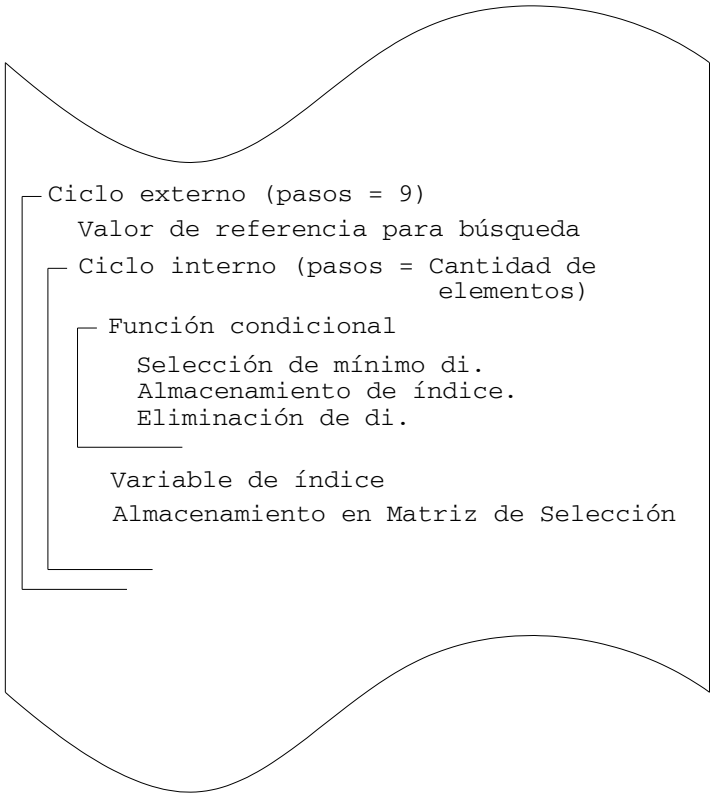


Figura V.13: Esquema de Código para el Algoritmo de Centros de Masa

Evidentemente, lo antes descrito deberá repetirse, como se explicó, para cada punto a localizar, generando así la Matriz de Selección.

V.5. ALGORITMO ESUP

El algoritmo ESUP recoge su nombre de la frase en inglés "Elements Surrounding Points" (elementos que rodean puntos) y toma sus bases en el trabajo de R. Löhner [4]. El algoritmo que Löhner propone en su obra tiene por objetivo minimizar el uso de memoria durante su actuación y, dado que en el presente trabajo el objetivo es minimizar el tiempo de ejecución, se le han realizado algunas variaciones. Por otro lado, debido a que el proceso original no se orienta al sistema de selección que aquí se trata, también fue necesario efectuar modificaciones al respecto.

Conceptualmente, el método ESUP consiste en generar un arreglo que permita, dado un nodo de la ME, identificar los elementos que lo rodean, es decir, aquellos que lo contienen. Luego, se incorpora este arreglo en un criterio de selección para el segundo módulo.

La cantidad de elementos que rodean un nodo varía sensiblemente dentro de una malla de FEM. Por lo tanto, utilizar una matriz que almacene esta información resultaría en la imposibilidad de realizar un pre-dimensionamiento, generando una pérdida importante de eficiencia. La solución a este inconveniente también es

propuesta por Löhner en su trabajo y consiste en el uso de Listas Vinculadas ("Linked Lists"), las cuales se describen más adelante.

El criterio de selección del Algoritmo ESUP consiste en encontrar, para cada punto a localizar, el nodo de la ME más próximo y seleccionar los elementos que rodean a dicho nodo para la evaluación posterior. Si para este proceso se utilizan las Listas Vinculadas, no será posible construir una Matriz de Selección como en los casos anteriores. La cantidad de elementos a evaluar para cada punto no será constante y esto debe ser tenido en cuenta en el Módulo de Localización.

V.5.1. LISTAS VINCULADAS

Las listas vinculadas son una herramienta que permite reemplazar el uso de matrices cuando no es posible conocer a priori la cantidad de datos a almacenar. En el caso de análisis, la cantidad de elementos que rodean a un nodo no es un número constante a lo largo de la ME. Para evitar el uso de un arreglo de dimensión variable que, si bien es soportado por MATLAB, representa una pérdida grande de eficiencia, se utilizan las listas vinculadas.

El sistema de listas vinculadas cuenta con dos vectores filas que se relacionan entre sí. En el primero, denominado "Storage", se almacena la información propiamente dicha. En el segundo, denominado "Links", se encuentran marcadores que indican dónde comienzan y dónde terminan los datos de una entidad particular.

Considérese el caso ejemplo de la Figura V.14, donde se muestra una malla sencilla para la cual se almacenaron los elementos que rodean a cada nodo en listas vinculadas. En el vector "Links", el índice de posición se condice con los nodos de la malla y, los valores que contiene indican la última posición del vector "Storage", donde se almacena un elemento que rodea al nodo indicado por el índice de posición. Por ejemplo, si se quisiera determinar qué elementos rodean al nodo 4, se debe recurrir al índice de posición 4 del vector "Links", donde el valor correspondiente es 7. Este valor indicará que hasta la posición 7 del vector "Storage" se enlistan los elementos que rodean al nodo 4. Como para el nodo anterior, el 3, la aplicación de este proceso indicó que los elementos que lo circundan se enlistan hasta la posición 4 del vector "Storage", es desde allí de donde empiezan los correspondientes al nodo 4. Siendo los elementos 1, 2 y 4 los que efectivamente rodean al nodo 4.

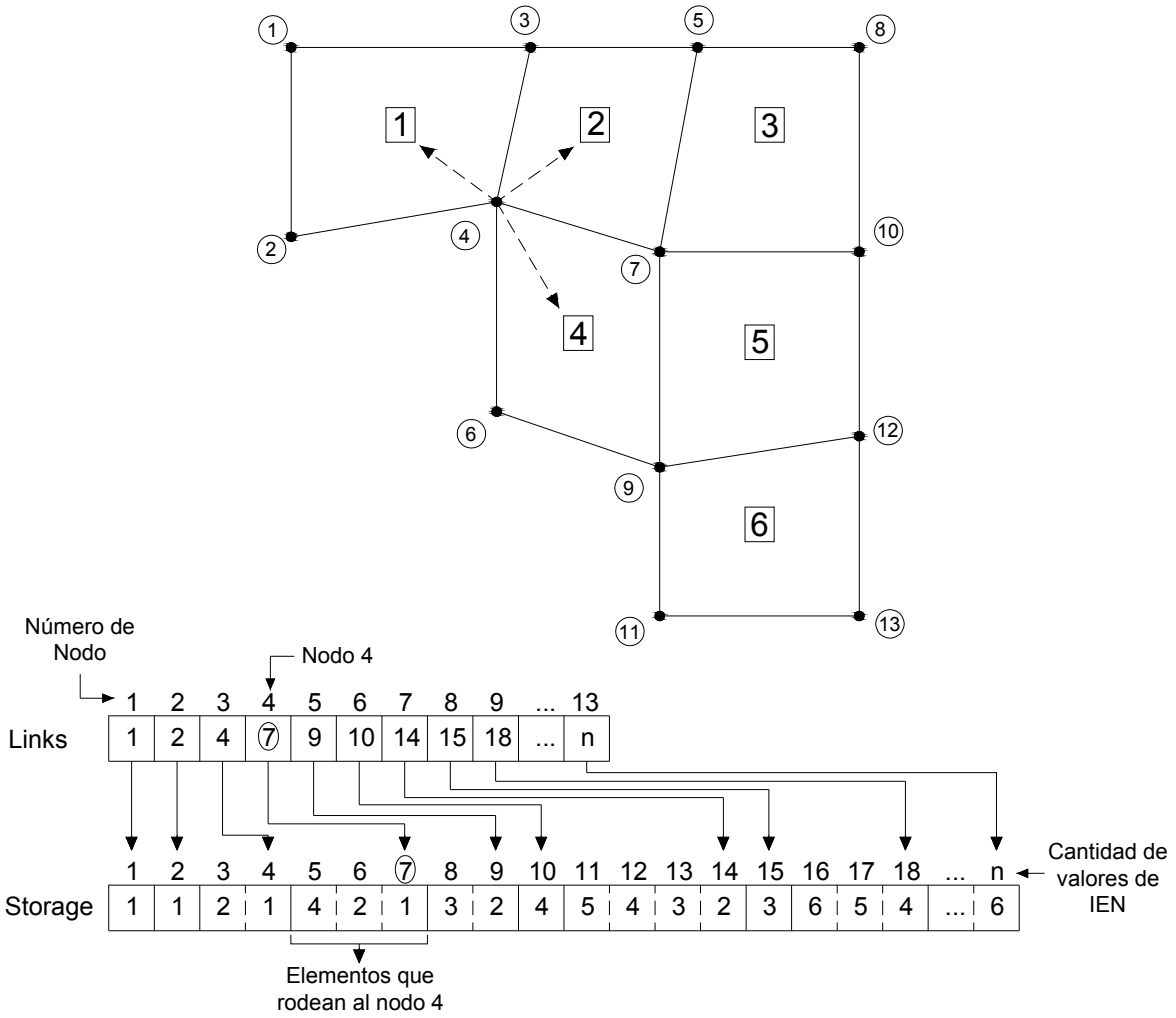


Figura V.14: Listas Vinculadas para el Algoritmo ESUP para Malla Ejemplo

V.5.2. CONFECCIÓN DE LISTAS VINCULADAS

La construcción de las listas vinculadas para el algoritmo ESUP se basa en la utilización del Arreglo IEN de la ME. Como primer paso se realiza el pre-dimensionamiento de ambos vectores. La dimensión del vector "Links" es de sencilla interpretación y está dada por la cantidad de nodos que posee la malla. Para el vector "Storage", en cambio, debe seguirse el siguiente razonamiento: si un elemento será circundante a todos los nodos que posee, cada elemento figurará en el vector "Storage" una vez por cada nodo que posea, luego, la dimensión del vector estará dada por la cantidad de elementos multiplicada por la cantidad de nodos de cada elemento. Este número equivale a la cantidad de valores que guarda el Arreglo IEN ya que, por cada elemento, guarda tantos valores como nodos éste posea.

Con las dimensiones de los vectores establecidas, se procede a completarlos. Para dicha tarea es preciso utilizar un vector auxiliar o temporal, que posee igual

dimensión que "Links" y debe confeccionarse previamente. El vector auxiliar asignará cada una de sus elementos a un nodo de la malla y contendrá la cantidad de veces que cada nodo figura en el Arreglo IEN. La cuenta de apariciones de cada nodo se realiza utilizando un ciclo que recorra el Arreglo IEN y sume una unidad al elemento correspondiente cada vez que lea un valor del arreglo. En la figura siguiente se muestran el Arreglo IEN y el vector auxiliar para la malla del ejemplo de la Figura V.14.

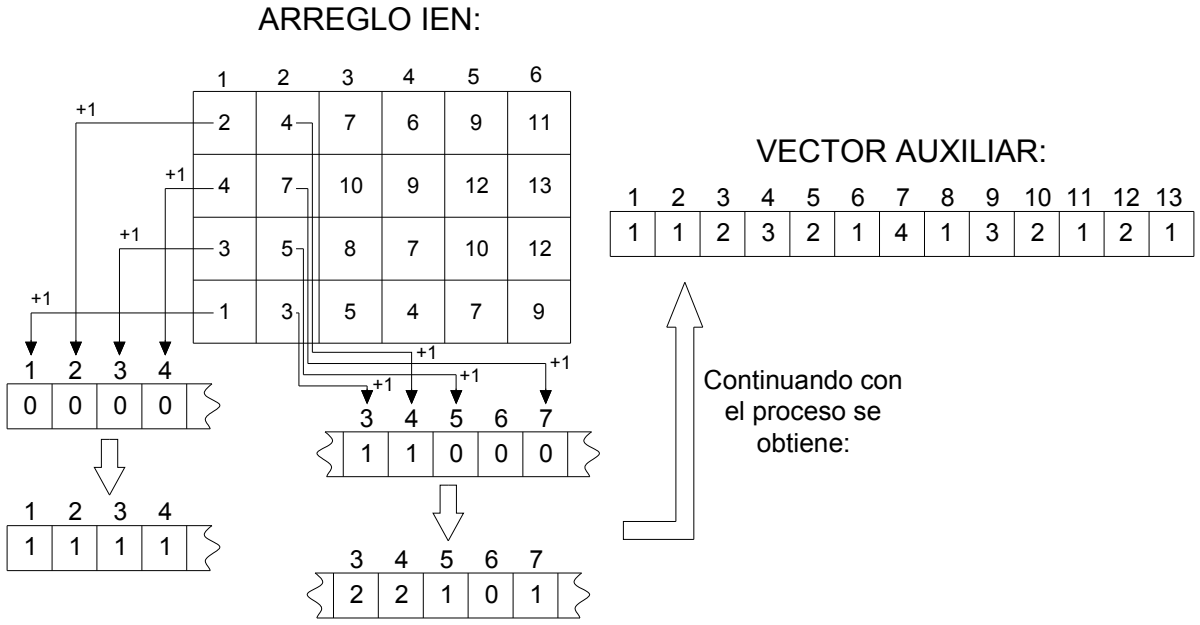


Figura V.15: Esquema de Confección de Vector Auxiliar a partir de Arreglo IEN

El vector auxiliar será útil tanto para confeccionar al vector "Links" como al vector "Storage". En el primer caso, el vector "Links" se completa con un ciclo que recorre el vector auxiliar y, en cada paso, almacena sobre sí mismo la suma de cada valor con el que lo precede. Desde otro punto de vista, el mecanismo equivale a asignar a cada posición del vector "Links" la suma de todos los elementos anteriores a esa posición del vector auxiliar. La Figura V.16 representa este proceso.

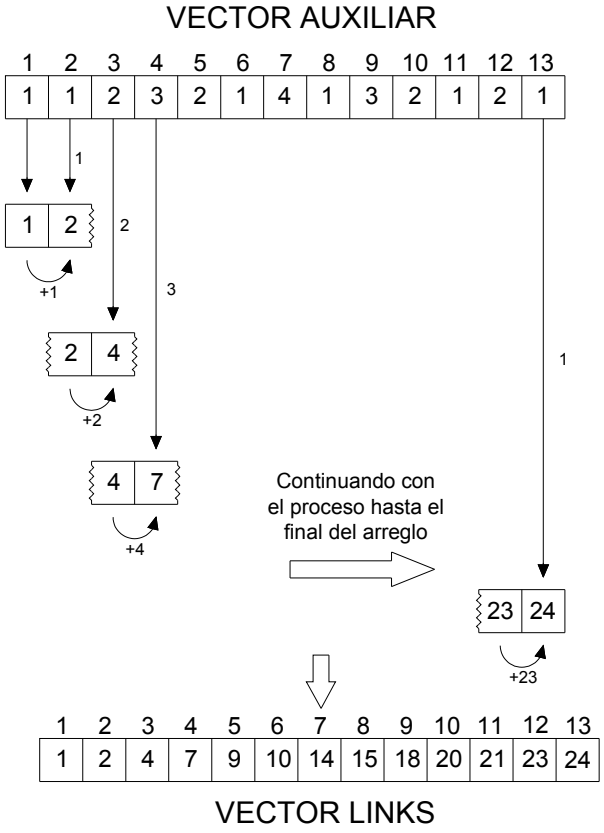


Figura V.16: Esquema de Confección del Vector "Links" a partir del Vector Auxiliar

En la práctica, los pasos anteriores se efectúan sólo sobre el vector auxiliar. Es decir, a medida que avanza el ciclo, se toma un valor del vector auxiliar, se lo suma con su precedente y se lo almacena en su misma posición. Así, en cada paso se actualizan los valores del vector auxiliar y al finalizar el ciclo, dicho vector se convierte en el vector "Links". En este punto, el vector auxiliar actualizado se duplica y se almacena una copia como "Links" y la restante se emplea para construir el vector "Storage"

El vector "Storage" se genera interactuando simultáneamente con el Arreglo IEN y el vector auxiliar actualizado (que es equivalente al vector "Links"). El proceso se repite a lo largo de todo los valores del Arreglo IEN y se describe en la siguiente secuencia:

- Se lee un valor del Arreglo IEN, sea "i".
- Se lee el valor del elemento "i" del vector auxiliar, sea "a".
- Se guarda el número de elemento que corresponde a "i" dado por el Arreglo IEN, en la posición "a" del vector "Storage".
- Se reduce en una unidad el valor de "a" en el vector auxiliar.
- Se repite la secuencia con el siguiente valor del Arreglo IEN.

El cuarto paso de la secuencia evita la superposición de datos, desplazando el cursor convenientemente hacia atrás. En la figura que sigue se ilustra el proceso.

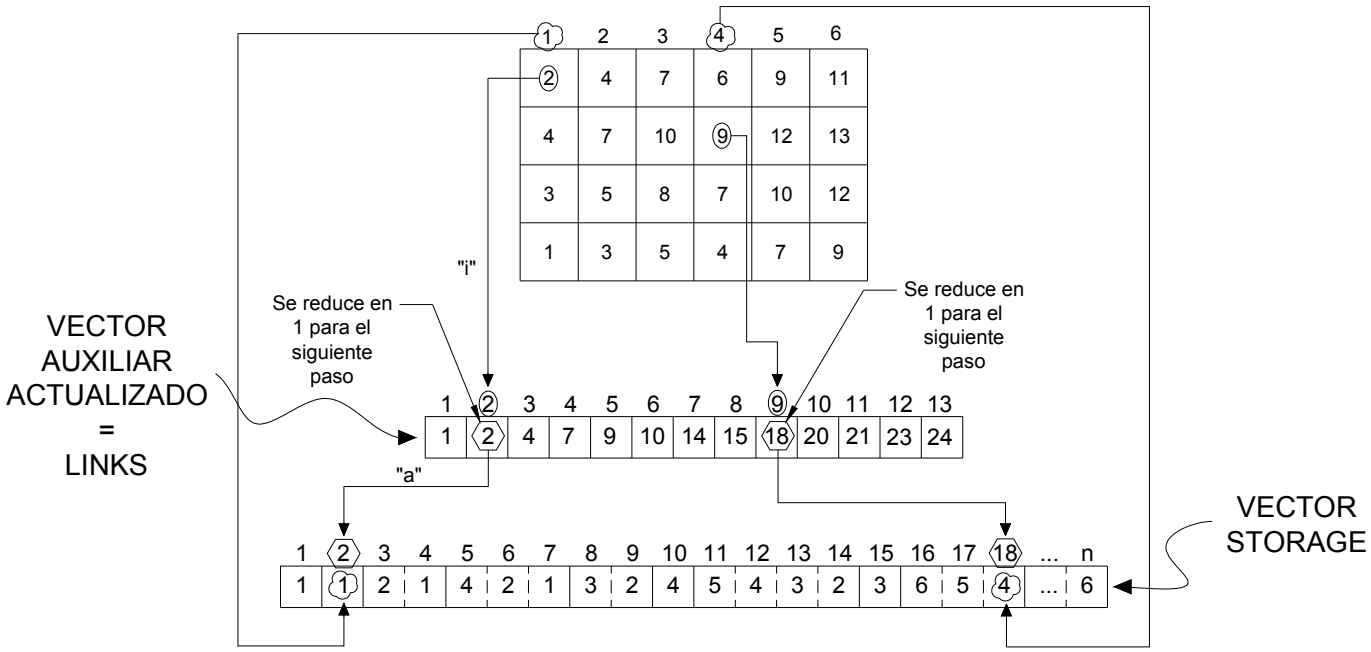


Figura V.17: Esquema de Confección del Vector "Storage"

V.5.3. IMPLEMENTACIÓN

La utilización de las listas vinculadas de elementos que rodean a un punto como un sistema de selección aplicable al segundo módulo se lleva a cabo, como ya se había anticipado, encontrando el nodo más cercano a cada punto a localizar. Esta tarea no reviste mayor complejidad y su resultado se plasma en un vector que contendrá los nodos más cercanos a cada punto, denominado "Closest".

La búsqueda del nodo más próximo a un punto se realiza de manera similar a lo detallado en la sección V.4.1, es decir, se obtiene de forma análoga el vector ***D***. El paso siguiente consiste en la búsqueda del menor valor de d_i (con i $= 1, 2, \dots,$ n), y el almacenamiento del índice correspondiente en el vector "Closest". El índice de posición del vector de nodos más próximos se condice con el orden de los puntos a localizar dados por el Arreglo de Nodos de la GA.

Finalmente, los tres vectores, "Storage", "Links" y "Closest" serán transferidos al Módulo de Localización donde, con el correcto procesamiento, permitirán seleccionar los elementos a evaluar. Primero, para un punto a localizar, se identifica del vector "Closest" el índice del nodo más próximo. Con este índice se recurre al vector "Links" donde se recogen los marcadores. Luego, se utilizan los marcadores en el vector "Storage" para identificar los elementos a evaluar.

V.6. ALGORITMO ESUP ORDENADO POR CENTROS DE MASA

Este último algoritmo es una combinación de los dos anteriores, ESUP y Centros de Masa, y constituye quizás la variante más compleja de las que aquí se presentan. El sistema consiste en evaluar de forma ordenada según el criterio de Centros de Masa a los elementos que son seleccionados con el criterio de ESUP.

Este método desdibuja la distinción que existe entre el Módulo de Selección y el Módulo de Localización. Esto se debe a que no existe una forma práctica de generar un arreglo invariante, que se transfiera al último módulo del Programa de Vinculación, con los datos suficientes para evaluar los elementos seleccionados. En contraste, se genera un bloque de programa intermedio entre ambos módulos que, entre cada ciclo de evaluación del Módulo de Localización, calcula un vector con los elementos a evaluar a partir de los datos aportados por el Módulo de Selección.

V.6.1. IMPLEMENTACIÓN

No es mucho lo que se puede agregar sobre la implementación de este algoritmo combinado. Teniendo en cuenta que en las secciones anteriores se detalló el funcionamiento de cada uno de los procesos por separado, no se requiere aquí mayor explicación. En este sentido, se abordarán sólo las particularidades del caso.

El objetivo final será obtener un vector que contenga los elementos a evaluar en el orden correcto. Este vector, denominado “Elements” será distinto para cada punto a localizar y se transferirá al Módulo de Localización en cada ciclo de evaluación.

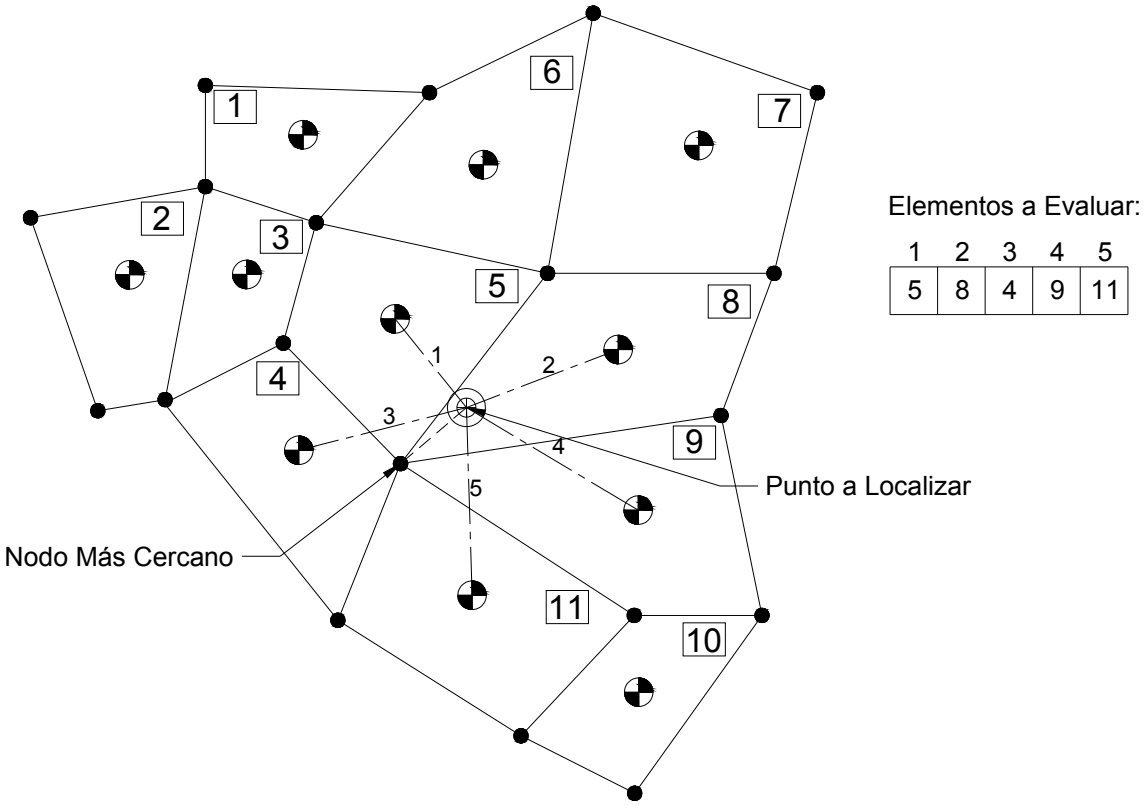


Figura V.18: Vector "Elements" para un Punto a Localizar

Como primer paso, se ejecuta el algoritmo ESUP idénticamente como se mostró en la sección V.5. También se genera la matriz $[XCM]$ dada por la expresión (V.4.1.1) y el vector de nodos más próximos a cada punto a localizar denominado "Closest" presentado en la sección V.5.3.

Luego, dado un punto a localizar se identifica el nodo más próximo y se recurre a las Listas Vinculadas de ESUP, como se explicó oportunamente. En este punto, se crean tres entes auxiliares, a saber: una matriz de coordenadas de centros de masa, $[XCM\_Temp]$; un vector de distancias $\boldsymbol{D\_Temp}$; y, un vector de índices $\boldsymbol{Index\_Temp}$. Cada uno de estas variables auxiliares se pre-dimensiona en función de la cantidad de elementos que las Listas Vinculadas asocian al nodo dado por "Closest". En los tres casos, se crearán tantas columnas como elementos indiquen Listas Vinculadas y, la matriz $[XCM\_Temp]$, tendrá además una fila para cada coordenada espacial. También se crea el vector "Elements" que tendrá idéntica dimensión que los dos anteriores.

Con la ayuda de las listas vinculadas, se identifican los índices de los elementos que rodean al nodo más cercano al punto que se pretende localizar, es decir, aquellos que se hubiesen evaluado en el Algoritmo ESUP tradicional. Estos índices son copiados en el vector $\boldsymbol{Index\_Temp}$ y, simultáneamente, se copian en la matriz $[XCM\_Temp]$, las coordenadas de sus centros de masa provenientes de $[XCM]$, transferidos como vectores columna.

El siguiente paso es calcular las distancias entre el punto a localizar y cada uno de los centros de masa extraídos. El proceso es análogo al caso explicado en la sección V.4.1, recurriendo nuevamente al Producto Tensorial o de Kronecker y a la norma euclidiana de los vectores columnas. Como resultado de este paso se construye el vector ***D_Temp***.

Finalmente, se inicia un ciclo clásico de ordenamiento que recorre el vector ***D_Temp***, y el vector ***Index_Temp***, simultáneamente. En función de las distancias, los índices de los elementos se ordenan en el vector "Elements" y se cierra el ciclo.

Lo explicado anteriormente debe repetirse para cada punto a localizar y es aquí donde se combinan los Módulos de Selección y Localización. Cada vez que se confecciona el vector "Elements" para un punto a localizar, se ejecuta el Módulo de Localización, y luego se reinicia el proceso. El cual se ilustra en la Figura V.19.

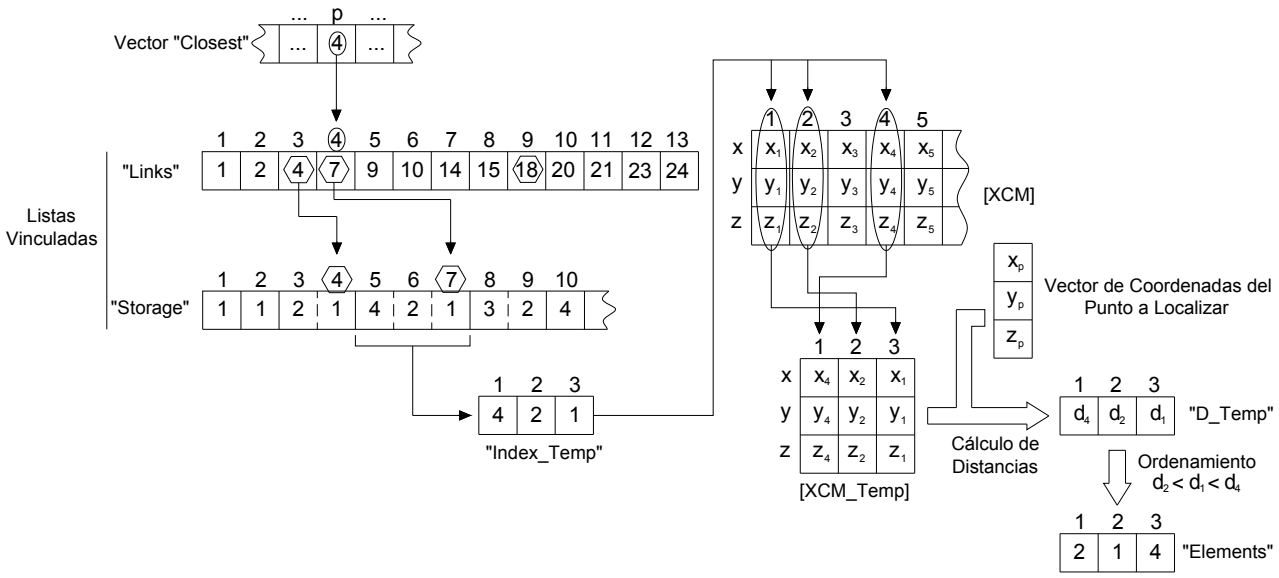


Figura V.19: Esquema para la Confección del Vector "Elements"

De este modo, se tienen, para cada punto a localizar, los elementos que serán evaluados.

Capítulo VI: Módulo de Localización. Algoritmos e Implementación

VI.1. Generalidades

El objetivo de este módulo es identificar biunívocamente el elemento de la ME que contenga a cada punto de la GA, para poder confeccionar así la Matriz de Localización que será el resultado final del Programa de Vinculación.

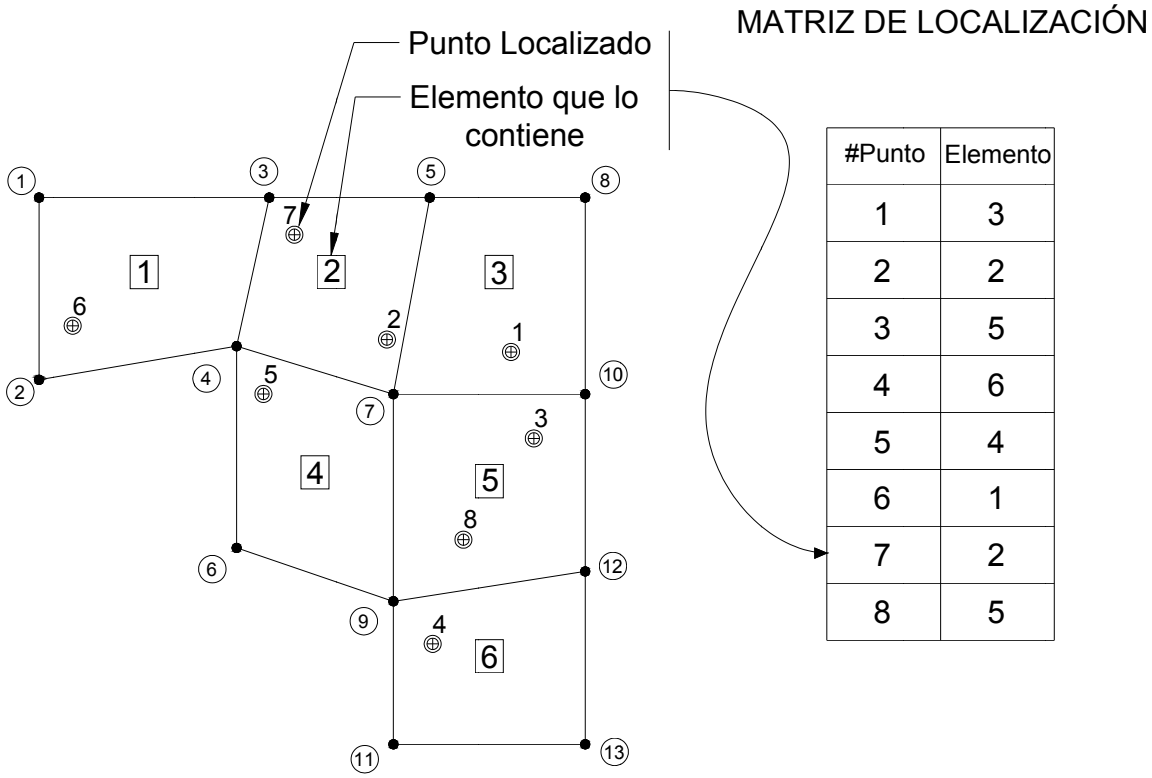


Figura VI.1: Matriz de Localización para Malla de Ejemplo

En el último módulo, la aplicabilidad práctica no pierde protagonismo. Por el contrario, existe una fuerte dependencia entre los algoritmos que se tratarán a continuación y las características particulares del problema que se desee resolver.

Nuevamente, se acude al criterio y a la experiencia para decidir en qué casos los métodos podrían presentar fallas.

El Módulo de Localización no es tan versátil en relación a la cantidad de variantes por las que se puede optar para llevar a cabo su objetivo. De hecho, son dos los métodos que se proponen, el primero toma la forma de un método numérico clásico, el método de Newton-Raphson. Mientras que el segundo recurre al concepto de las Coordenadas de Área, también conocidas como Coordenadas Triangulares.

Las descripciones de estos sistemas se realizan bajo la premisa de localizar un único punto, proveniente de la GA, en alguno de los elementos de la ME. Lógicamente, deberán ejecutarse tantas veces como puntos se deseen localizar y deberán acoplarse apropiadamente con los módulos anteriores. Se entiende también que tanto el Módulo de Ingreso como el Módulo de Selección ya fueron ejecutados y se cuentan con toda la información que proveen al código principal.

VI.2. El Algoritmo de Newton-Raphson

Este sistema de localización aplica una variante del algoritmo propuesto por Isaac Newton en 1669 y modificado por Joseph Raphson en 1690, de ahí que se escogió denominarlo de ese modo. En términos generales, se pretende la solución de un sistema de ecuaciones para determinar si un punto pertenece o no a un elemento, y es en dicha solución donde interviene este clásico método numérico.

El punto de partida de la localización a través de este proceso se encuentra en la aplicación del Dominio de Origen a los elementos que conforman la ME, según lo descrito en la sección II.5. Se vio que un elemento puede construirse a partir de su Dominio de Origen con un mapeo del tipo de la expresión (II.3.1.4), que se repite aquí por comodidad,

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{a=1}^{n} N_a(\boldsymbol{\xi})\, \boldsymbol{x_a} \qquad \text{(VI.2.1)}$$

Donde “n” indica la cantidad de nodos del elemento. Se indicó también que utilizando el Método de Degeneración (II.6.1) es posible referir los elementos comúnmente utilizados al Hexaedro Trilineal o al Cuadrilátero Bilineal, según se trabaje en dos o tres dimensiones.

Teniendo esto bajo consideración se plantea que si un punto en el “espacio ξ” pertenece al Dominio de Origen, su transformación, a través del mapeo de la expresión (VI.2.1), será interior al elemento en el “espacio x”, y viceversa. En la Figura VI.2, se muestran ambos casos en un ejemplo plano.

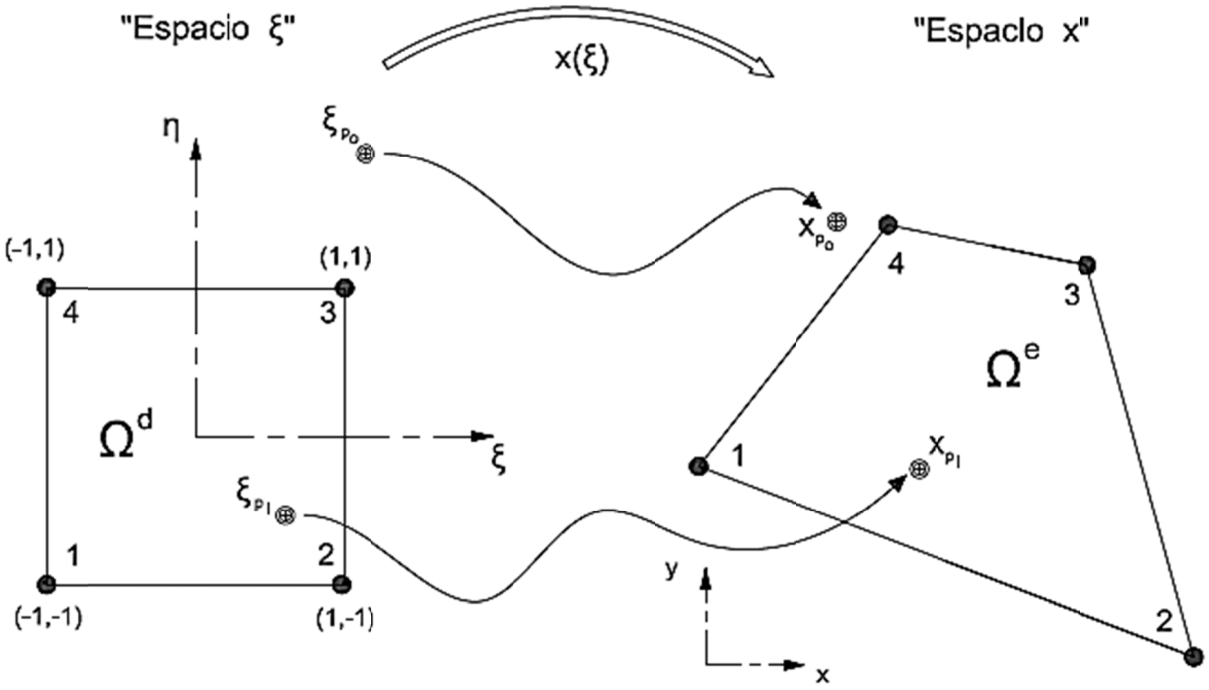


Figura VI.2: Mapeo de Puntos Internos y Externos al Dominio de Origen

Como puede observarse, el punto $\boldsymbol{X_{p_i}}$ pertenece al elemento en el "espacio x", por lo tanto, su imagen, $\boldsymbol{\xi_{p_i}}$, pertenecerá al Dominio de Origen. En oposición, el punto $\boldsymbol{X_{p_o}}$ es externo al elemento en el "espacio x" por lo que su imagen, $\boldsymbol{\xi_{p_o}}$, también será externa al Dominio de Origen. Simbólicamente, puede expresarse que

$$\boldsymbol{X_{p_i}} \in \Omega^e \Leftrightarrow \boldsymbol{\xi_{p_i}} \in \Omega^d. \qquad \text{(VI.2.2)}$$

Donde "Ω^e" y "Ω^d" indican los dominios del elemento y de origen respectivamente.

Comprendiendo que el Dominio de Origen en el caso plano es, y lo será para todos los elementos considerados aquí, el Cuadrilátero Bilineal, la condición de pertenencia puede expresarse en términos de las coordenadas del punto $\boldsymbol{\xi_{p_i}}$ como sigue

$$\boldsymbol{\xi_{p_i}} \in \Omega^d \Leftrightarrow \begin{cases} \left(-1 < \xi_{p_i} < 1\right) \\ \left(-1 < \eta_{p_i} < 1\right) \end{cases}. \qquad \text{(VI.2.3)}$$

Condiciones que deberán cumplirse simultáneamente.

Volviendo a la ecuación (VI.2.1), dado que cualquier punto del "espacio x" puede expresarse a través del mapeo, puede escribirse para un punto arbitrario que

$$\boldsymbol{X_p} = \sum_{a=1}^{n} N_a(\boldsymbol{\xi})\, \boldsymbol{x_a}. \tag{VI.2.4}$$

Reordenando,

$$\boldsymbol{X_p} - \sum_{a=1}^{n} N_a(\boldsymbol{\xi})\, \boldsymbol{x_a} = \boldsymbol{0}. \tag{VI.2.5}$$

La expresión (VI.2.5) representa un sistema de ecuaciones en el cual es posible identificar a $\boldsymbol{X_p}$ como el punto que se desea localizar, a $\boldsymbol{x_a}$ como los nodos del elemento que se evalúa, y a $N_a(\boldsymbol{\xi})$ como sus funciones de forma. Quedando sólo como incógnitas las coordenadas del punto $\boldsymbol{\xi}$. Evidentemente, el sistema tendrá tantas ecuaciones como dimensiones tenga el espacio de trabajo, dos para el plano y tres para el espacio. En este primer caso, se escribe,

$$\begin{cases} X_p - \sum_{a=1}^{n} N_a(\xi,\eta,\zeta)\, x_a = 0 \\ Y_p - \sum_{a=1}^{n} N_a(\xi,\eta,\zeta)\, y_a = 0 \end{cases}. \tag{VI.2.6}$$

Finalmente, la solución del sistema dará las coordenadas ξ y η. Si las coordenadas obtenidas cumplen con la condición dada por (VI.2.3), el punto en análisis pertenecerá al elemento evaluado, caso contrario deberá optarse por la evaluación de otro elemento.

Un razonamiento idéntico al anterior pero aplicado a un caso espacial deriva en el siguiente sistema

$$\begin{cases} X_p - \sum_{a=1}^{n} N_a(\xi,\eta,\zeta)\, x_a = 0 \\ Y_p - \sum_{a=1}^{n} N_a(\xi,\eta,\zeta)\, y_a = 0 \\ Z_p - \sum_{a=1}^{n} N_a(\xi,\eta,\zeta)\, z_a = 0 \end{cases}. \tag{VI.2.7}$$

Para el cual el Dominio de Origen es el Hexaedro Trilineal, y las condiciones de pertenencia se escriben como sigue

$$\boldsymbol{\xi_{p_i}} \in \Omega^d \Leftrightarrow \begin{cases} \left(-1 < \xi_{p_i} < 1\right) \\ \left(-1 < \eta_{p_i} < 1\right) \\ \left(-1 < \zeta_{p_i} < 1\right) \end{cases}. \tag{VI.2.8}$$

El sistema de ecuaciones de la expresión (VI.2.5) debe ser resuelto y es aquí donde interviene el método de Newton-Raphson.

VI.2.1. EL MÉTODO DE NEWTON-RAPHSON

La elección de este método para la solución del sistema de ecuaciones se justifica en la diversidad de tipos de funciones de forma N_a que pueden presentarse. Si pudiera garantizarse que, para una aplicación determinada, el sistema a resolver será lineal en las variables no tendría sentido desde el punto de vista de la eficiencia, el proceso iterativo propuesto. Sin embargo, dado que en general éste no es el caso, el método de Newton-Raphson proporciona una alternativa de aplicación general.

El método de Newton-Raphson está concebido como un algoritmo eficiente para encontrar ceros o raíces de funciones reales. Sin embargo, aquí se propone una variante aplicable al sistema de ecuaciones antes presentado.

El proceso de solución se inicia definiendo una función vectorial de la siguiente manera,

$$\boldsymbol{F_e}(\boldsymbol{\xi}, \boldsymbol{X_p}) = \boldsymbol{X_p} - \sum_{a=1}^{n} N_a(\boldsymbol{\xi})\, \boldsymbol{x_a}. \qquad \text{(VI.2.1.1)}$$

Debiendo ser, según lo planteado por la expresión (VI.2.5),

$$\boldsymbol{F_e}(\boldsymbol{\xi}, \boldsymbol{X_p}) = \boldsymbol{0}. \qquad \text{(VI.2.1.2)}$$

Ahora bien, proponiendo un proceso iterativo en la variable $\boldsymbol{\xi}$ y expandiendo la función $\boldsymbol{F_e}$, puede expresarse para un punto $\boldsymbol{\xi_{k+1}}$,

$$\boldsymbol{F_e}(\boldsymbol{\xi_{k+1}}, \boldsymbol{X_p}) = \boldsymbol{F_e}(\boldsymbol{\xi_k}, \boldsymbol{X_p}) + \left.\frac{\partial \boldsymbol{F_e}(\boldsymbol{\xi}, \boldsymbol{X_p})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi_k}} \Delta\boldsymbol{\xi_k} + H_2 \qquad \text{(VI.2.1.3)}$$

Donde,

$$\boldsymbol{\xi_{k+1}} = \boldsymbol{\xi_k} + \Delta\boldsymbol{\xi_k} \quad con\ k = 1,2,3 \ldots np. \qquad \text{(VI.2.1.4)}$$

Siendo "np" la cantidad de pasos de la iteración.

Además, la derivada parcial de la función $\boldsymbol{F_e}$ se identifica con la matriz jacobiana de la transformación (VI.2.1) según se presentó en la sección IV.4.3, se tiene

$$\frac{\partial \boldsymbol{F_e}(\boldsymbol{\xi}, \boldsymbol{X_p})}{\partial \boldsymbol{\xi}} = \left[J_e(\xi, X_p)\right] = [Xa] * [\mathrm{N}']. \qquad \text{(VI.2.1.5)}$$

Donde [N′] proviene de la expresión (IV.4.3.1).

"H_2" representa los términos de orden superior (desde segundo orden en adelante) que completarían la igualdad pero que son despreciados.

En la convergencia, debido a la definición de $\boldsymbol{F_e}$, se cumplirá que

$$\boldsymbol{F_e}(\boldsymbol{\xi_{k+1}}, \boldsymbol{X_p}) = \mathbf{0}. \quad \text{(VI.2.1.6)}$$

Luego,

$$\boldsymbol{F_e}(\boldsymbol{\xi_k}, \boldsymbol{X_p}) + \left.\frac{\partial \boldsymbol{F_e}(\boldsymbol{\xi}, \boldsymbol{X_p})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi_k}} \Delta\boldsymbol{\xi_k} = \mathbf{0}. \quad \text{(VI.2.1.7)}$$

Reemplazando (VI.2.1.5) en (VI.2.1.7) y reordenando, resulta

$$\left[J_e(\xi_k, X_p)\right] * \Delta\boldsymbol{\xi_k} = -\boldsymbol{F_e}(\boldsymbol{\xi_k}, \boldsymbol{X_p}) \quad \text{(VI.2.1.8)}$$

Expresión que permite calcular $\Delta\boldsymbol{\xi_k}$ fijando un valor inicial de $\boldsymbol{\xi_k}$. Seguidamente, teniendo en cuenta (VI.2.1.4) será posible calcular el valor $\boldsymbol{\xi_{k+1}}$ para continuar el proceso iterativo.

VI.2.2. IMPLEMENTACIÓN

La implementación del Algoritmo de Newton-Raphson en el Módulo de Localización implica la ejecución de los Módulos de Ingreso y Selección. Esto se debe a que se requieren el Arreglo de Nodos y el Arreglo IEN de la ME, el Arreglo de Nodos de la GA, la Biblioteca de Funciones de Forma y la de Funciones de Forma Derivadas.

El Módulo de Localización se compone de dos ciclos anidados: uno, el externo, recorre los puntos a localizar, y otro, el interno, recorre los elementos a evaluar para cada punto de acuerdo al conjunto dado por el Módulo de Selección.

El primer paso consiste en generar la matriz $[Xa]$ para el elemento a evaluar, dada por la expresión (V.4.1.2). Seguidamente, se toma $[Xa]$, el vector de coordenadas del punto a localizar, $\boldsymbol{Xp}$, y el vector de funciones de forma planteado en (II.5.1.12), $\boldsymbol{N}$, para construir de forma simbólica la función vectorial $\boldsymbol{F_e}$, como sigue,

$$\boldsymbol{F_e} = \boldsymbol{Xp} - [Xa] * \boldsymbol{N}. \quad \text{(VI.2.2.1)}$$

En coincidencia con (VI.2.1.1).

Por otro lado, se construye la matriz $[J_e]$, con la ayuda de la matriz de funciones de forma derivadas, [N′], (IV.4.3.1), y la matriz $[Xa]$. Siguiendo la expresión (VI.2.1.5), se hace

$$[J_e] = [Xa] * [\mathrm{N}']. \quad \text{(VI.2.2.2)}$$

Habiendo construido $\boldsymbol{F_e}$ y $[J_e]$, se da paso a la resolución del sistema dado por (VI.2.1.8). Sin embargo, previo a esto se definen dos parámetros: el valor inicial y el error aceptado. El valor inicial se corresponde con el primer valor del vector $\boldsymbol{\xi_k}$, que dará inicio a las iteraciones. Por otro lado, el error aceptado será aquel valor de tolerancia que se le dará a la convergencia, y a las condiciones de pertenencia. Por defecto, el valor inicial se fija en el origen y el valor de tolerancia es un número razonablemente pequeño, a saber,

$$\boldsymbol{\xi_0} = \mathbf{0} \quad \text{(VI.2.2.3)}$$

$$Err = 0{,}0001. \quad \text{(VI.2.2.4)}$$

Ahora se resuelve el sistema de ecuaciones con la ayuda de un algoritmo de reducción y sustitución. Luego, según (VI.2.1.4) en cada paso se obtendrá un nuevo valor de $\boldsymbol{\xi_k}$.

Se tienen entonces tres condiciones de ruptura del ciclo interno (el que recorre los elementos). La primera corresponde a la convergencia con localización, es decir, alcanzada la convergencia, el valor de $\boldsymbol{\xi_k}$ obtenido cumple las condiciones de pertenencia (VI.2.3) ó (VI.2.8) y se garantiza que el punto es interno al elemento evaluado. En este caso, en dos dimensiones se tiene

$$\begin{cases} (-1 - Err) < \xi_k < (1 + Err) \\ (-1 - Err) < \eta_k < (1 + Err) \\ \|\boldsymbol{\xi_{k+1}} - \boldsymbol{\xi_k}\| < Err \end{cases} \qquad \text{(VI.2.2.5)}$$

Una ventaja de esta condición es que, además de determinar que el punto es interno al elemento, permite conocer las coordenadas locales de dicho punto, $\boldsymbol{\xi_k}$, dentro del elemento. Esta propiedad es necesaria en muchas aplicaciones y le da a este método preferencia en dichos casos.

La segunda se da cuando existe convergencia pero sin localización y ocurre cuando la entre dos pasos sucesivos el módulo del vector $\boldsymbol{\xi_k}$ no es mayor que la tolerancia. Además, no se cumplen las condiciones de pertenencia. Esto es, en dos dimensiones,

$$\begin{cases} \|\boldsymbol{\xi_{k+1}} - \boldsymbol{\xi_k}\| < Err \\ |\xi_k|, |\eta_k| > 1 + Err \end{cases}. \qquad \text{(VI.2.2.6)}$$

La última condición refiere a las características del proceso mismo. El método de Newton-Raphson posee un orden convergencia por lo menos cuadrático, lo que indica que no deberían requerirse una gran cantidad de iteraciones para arribar a un resultado. Luego, se estima que, a lo sumo, serán necesarias siete iteraciones para converger con la tolerancia propuesta, de lo contrario el punto inicial no es correcto y el resultado no es el buscado. En este sentido, el ciclo iterativo cuenta sólo con siete pasos para completar la evaluación.

Como es de suponerse, la ruptura del ciclo interior conlleva a la evaluación del siguiente elemento pero sin modificar el punto que se analiza, puesto que continúa activo el ciclo externo. Esto es correcto para los dos últimos casos. No obstante, si se cumple la primera condición, es decir que el punto se localiza en el elemento, el ciclo externo también debe ser interrumpido. Esto se debe a que no tendrá sentido seguir evaluando elementos si ya se conoce la ubicación del punto. Lo que se realiza, entonces, es asignar el número de elemento con respuesta positiva (localización) en el lugar correspondiente en la Matriz de Localización y continuar con el siguiente punto reiniciando el proceso.

Finalmente y luego de que el ciclo externo concluya todos sus pasos, la Matriz de Localización estará completa. Así el objetivo estará cumplido y el Programa de Vinculación podrá darse por concluido.

VI.2.3. REDUCCIÓN DE ELEMENTOS

La Reducción de Elementos es un complemento pensado para implementar en el Algoritmo de Newton-Raphson y, en definitiva, en cualquier sistema de localización que utilice las funciones de forma de los elementos en el proceso. Conceptualmente consiste en recurrir a los elementos incompletos o de transición sobre los que se hizo referencia en la sección II.7. Estos elementos tienen la particularidad de prescindir de los nodos "interiores" respecto de sus contrapartes, los elementos completos. Por ejemplo, considérense los siguientes casos representados en la Figura VI.3.

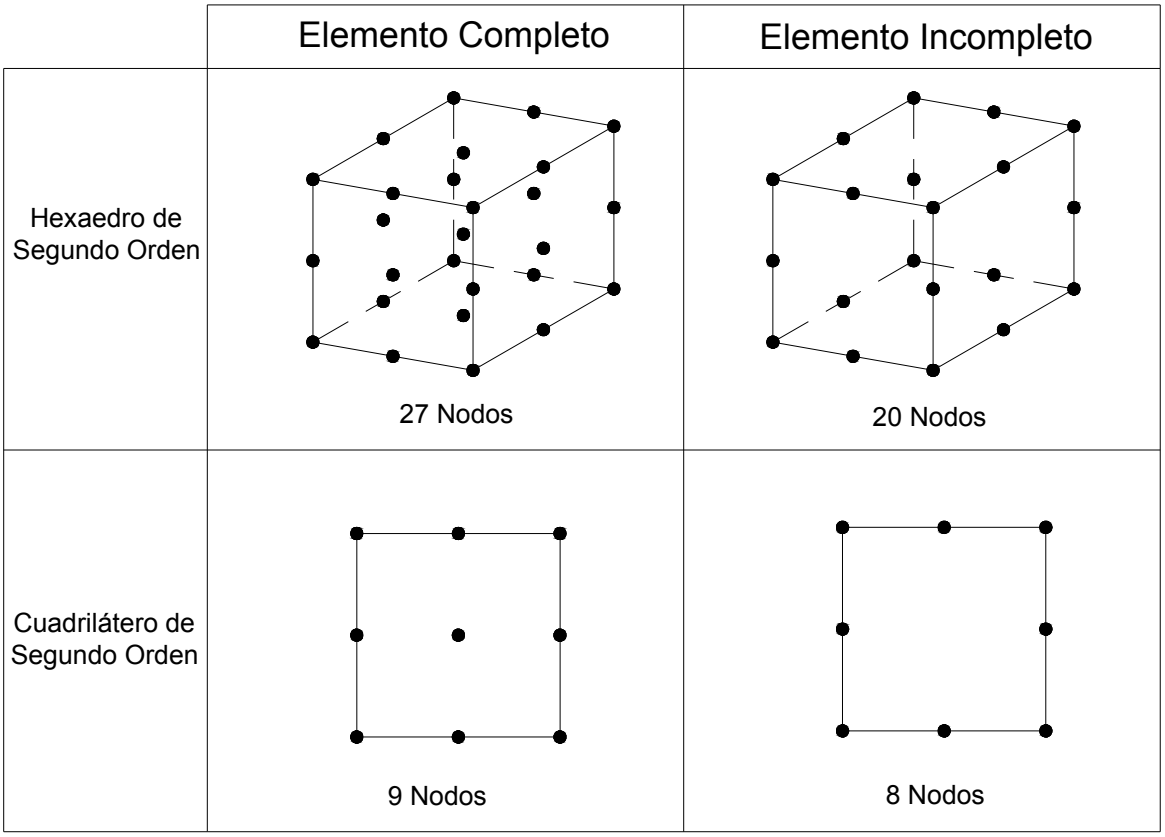


Figura VI.3: Casos de Elementos Incompletos

En ambos casos, el elemento completo cuenta con una serie de nodos interiores que no modifican su geometría y, sin embargo, incrementan el número de funciones de forma. El hexaedro de segundo orden completo cuenta con 27 nodos, mientras que el mismo elemento incompleto cuenta con 20 nodos, lo que implica que el vector de funciones de forma posee una dimensión un 26% menor. Adicionalmente, si los nodos exteriores poseen las mismas coordenadas en ambos casos, la geometría del elemento permanece inalterada. En consecuencia, realizar la localización con el hexaedro de 27 nodos o con el de 20 es indistinto. La única diferencia es la reducción en el número de

funciones de forma que significa una disminución en la cantidad de operaciones del método. Así, este fenómeno permite aumentar la eficiencia del código.

IMPLEMENTACIÓN

La implementación de la Reducción de Elementos es sencilla. Consiste sólo en configurar la Biblioteca de Tipos de Elementos (presentada en la sección IV.4.1) para que, en los casos que sea posible, altere la variable "ElemetType" y trabaje con la versión incompleta del elemento en cuestión. De este modo, las Bibliotecas de Funciones de Forma y de Funciones de Forma Derivadas, el Arreglo IEN, la matriz $[Xa]$, y todos los demás procesos reconocerán el elemento incompleto.

Una salvedad debe realizarse aquí respecto del Ordenamiento Nodal expuesto en la sección IV.2. Como se dijo, se optó por respetar la secuencia propuesta por T. Hughes que en este sentido resulta ventajosa. La relación entre los elementos completos e incompletos, en referencia a la numeración local de sus nodos, proporcionada por el Ordenamiento escogido permite evitar cualquier tipo de reordenamiento. Esto es, si se desea, por ejemplo reducir el hexaedro de segundo orden de 27 nodos a su versión incompleta de 20 nodos, sólo es necesario suprimir los últimos 7 nodos, puesto que las numeraciones coinciden.

VI.3. ALGORITMO DE COORDENADAS DE ÁREA

El algoritmo que se trata en esta sección es de aplicación exclusiva a elementos de primer orden. Debido a que es condición necesaria que las aristas o bordes del elemento sean rectos.

El método deriva de la utilización de coordenadas de área, también conocidas como coordenadas triangulares. Este sistema se formuló para la esquematización en el plano de funciones de tres variables procurando salvar las posibles confusiones que acarrean los gráficos en tres coordenadas cartesianas. Sin embargo, rápidamente fue implementado en el FEM como funciones de interpolación para los clásicos elementos conocidos como Triángulos de Tensión Constante. Aquí se utilizan sus fundamentos para desarrollar un mecanismo de localización que con algunas modificaciones se aplica a cuadriláteros y a elementos espaciales.

Conceptualmente, las coordenadas de área permiten posicionar biunívocamente un punto dentro de un plano triangular con una terna ordenada. Cada valor de la terna corresponde al área de cada triángulo que surge de unir el punto interno con cada uno de los vértices, expresada de manera relativa al área total. En la Figura VI.4 se observa un esquema que representa las coordenadas de área de un punto arbitrario ("P") dentro del triángulo.

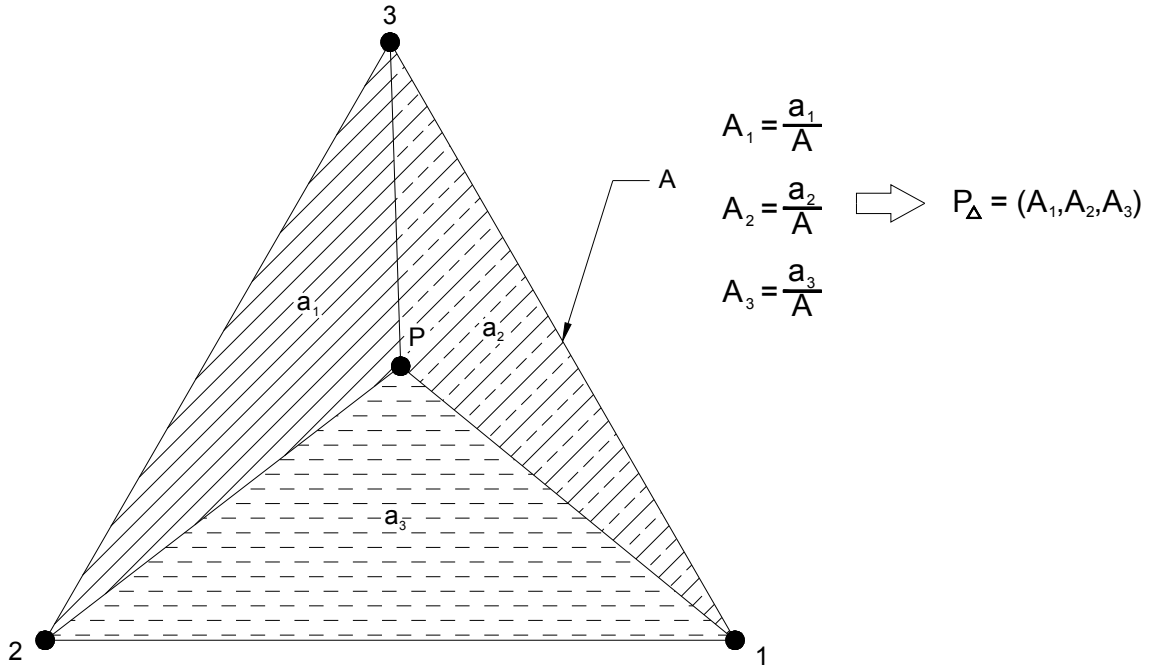


Figura VI.4: Coordenadas de Área o Triangulares para un Punto Arbitrario "P"

La característica más importante de este sistema de coordenadas es la condición que se establece entre los elementos de la terna,

$$a_1 + a_2 + a_3 = A \qquad \text{(VI.3.1)}$$

$$A_1 + A_2 + A_3 = 1. \qquad \text{(VI.3.2)}$$

Estas nociones pueden ser utilizadas en un algoritmo de localización, cuyo principio emerge de las expresiones anteriores. La condición dada por (VI.3.1) se cumplirá siempre que el punto "P" sea interno al triángulo, caso contrario la igualdad no verificará. A modo de ejemplo considérese el siguiente caso, en el cual se ilustra esta situación.

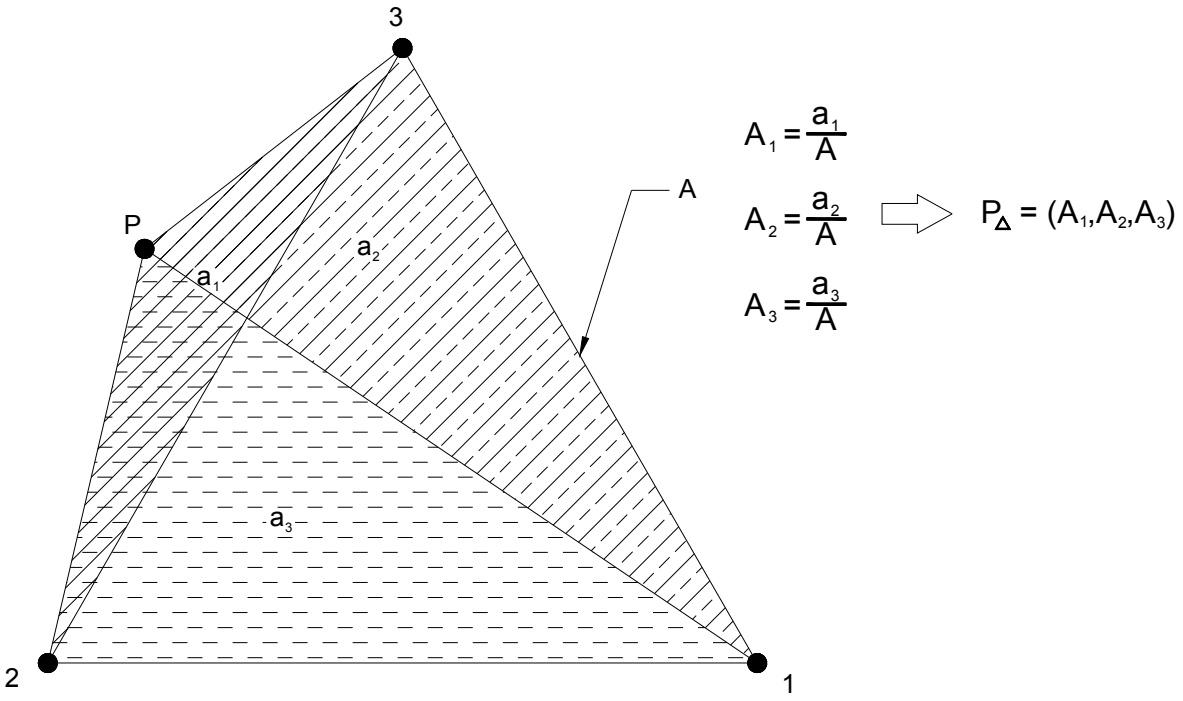


Figura VI.5: Coordenadas de Área para un Punto Externo al Plano Triangular

En este caso, se tiene,

$$a_1 + a_2 + a_3 > A \qquad \text{(VI.3.3)}$$

$$A_1 + A_2 + A_3 > 1. \qquad \text{(VI.3.4)}$$

Este fenómeno permite diferenciar un punto interno al plano triangular de uno externo, lo que resulta conveniente en relación al objetivo del Módulo de Localización. A este respecto, si se identifica el plano triangular con un elemento, sólo será necesario calcular las áreas parciales a_i (con i=1, 2, 3), sumarlas y comparar el resultado con el área total A. Luego, si la condición (VI.3.1) verifica, el punto es interno al elemento, caso contrario se cumplirá (VI.3.3).

Evidentemente, propuesto de esta manera el método sólo aplica a elementos triangulares planos pero como se mencionó es posible extenderlo a otros casos. Para un elemento cuadrilátero se requerirán cuatro coordenadas, es decir, se deberá calcular el área de cuatro triángulos, como muestra la Figura VI.6.

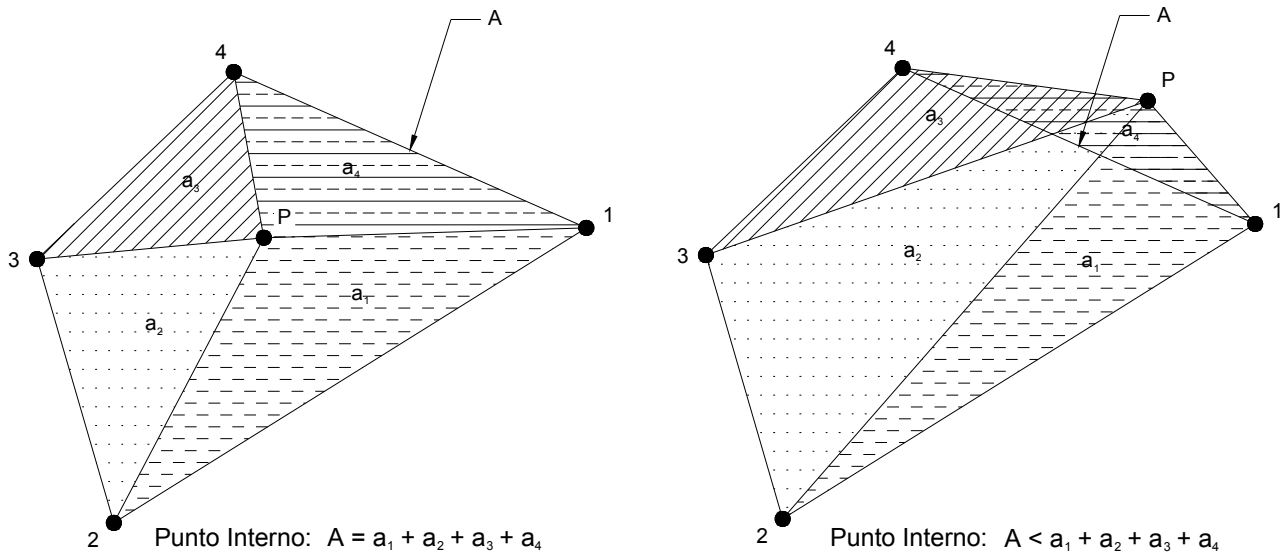


Figura VI.6: Coordenadas de Área en un Elemento Cuadrilátero

Nuevamente, si el punto es interior la suma de las áreas parciales igualará al área total, es decir,

$$a_1 + a_2 + a_3 + a_4 = A. \qquad \text{(VI.3.5)}$$

Por el contario para un punto externo no se verificará la igualdad, siendo

$$a_1 + a_2 + a_3 + a_4 > A. \qquad \text{(VI.3.6)}$$

Los elementos definidos en el espacio tridimensional requieren un tratamiento diferente pero que no pierde de vista el concepto propuesto. En estos casos, no se calculan áreas sino volúmenes y la cantidad de éstos dependerá del tipo de elemento. Se tratarán aquí los elementos hexaedros, puesto que representan la versión más general de los elementos espaciales. Otros elementos como pirámides, prismas y tetraedros, pueden considerarse como porciones de los hexaedros y su tratamiento se deriva con facilidad bajo esta premisa.

Primeramente, debe plantearse que un elemento hexaedro puede dividirse en seis fragmentos. Estas partes son pentaedros (pirámides) que tienen por base cada una de las caras del elemento original y como cúspide un punto arbitrario, “P”.

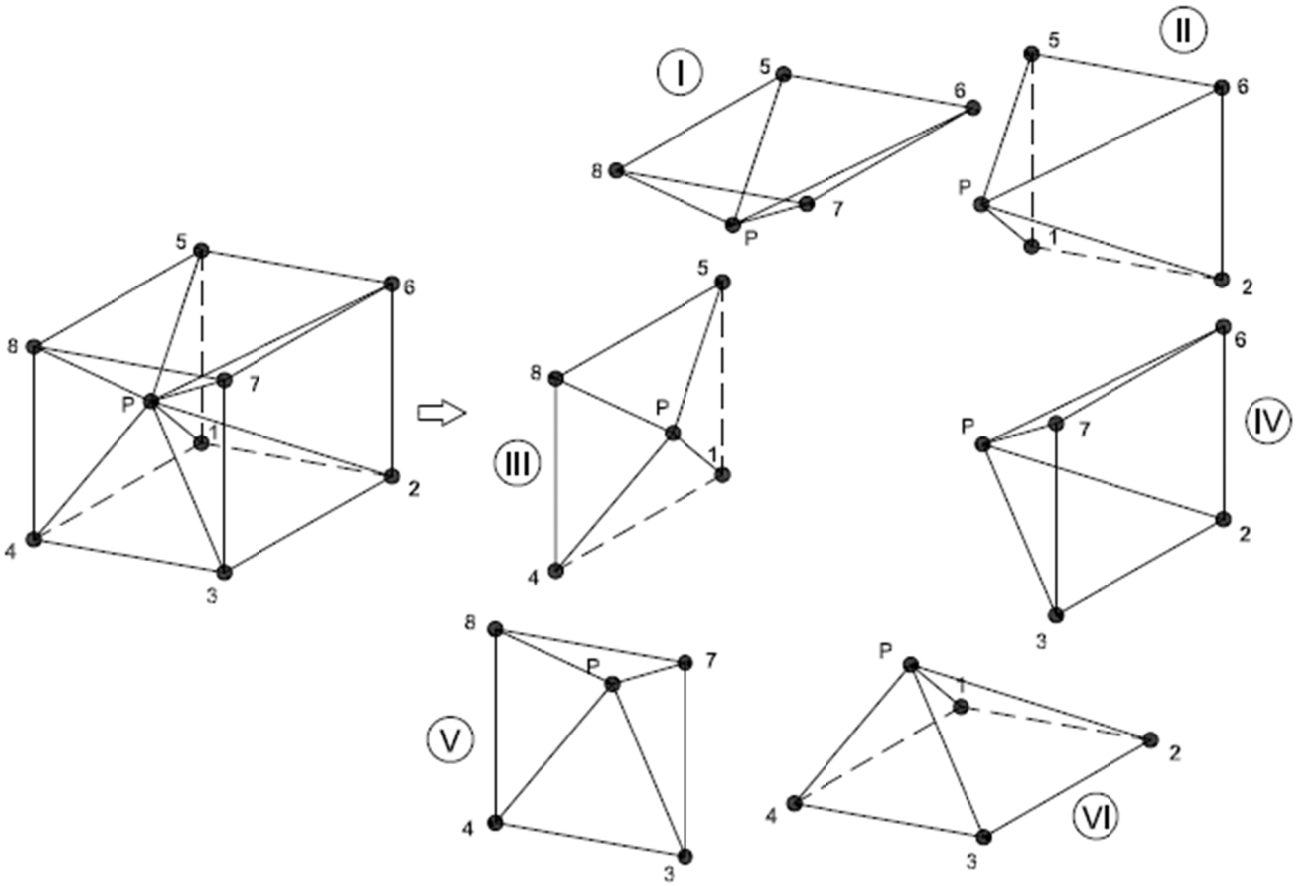


Figura VI.7: Elemento Hexaedro Fragmentado a Partir del Punto "P"

Luego, se retoma el razonamiento anterior y se expresa la comparación entre volúmenes. La suma de los volúmenes de los seis subelementos será igual al volumen total si, y sólo si, el punto "P" es interno al elemento original. Se tendrá entonces que, habrá localización para el caso en que

$$v_1 + v_2 + v_3 + v_4 + v_5 + v_6 = V. \qquad \text{(VI.3.7)}$$

Contrariamente, si el punto es externo al elemento, se verifica que

$$v_1 + v_2 + v_3 + v_4 + v_5 + v_6 > V. \qquad \text{(VI.3.8)}$$

La aplicación de este proceso a otros tipos de elementos espaciales no reviste mayor dificultad. Debe comprenderse que, por ejemplo, un prisma triangular puede dividirse en cinco fragmentos, dos tetraedros y dos pentaedros. Siempre considerando el punto "P" como vértice de los subelementos.

VI.3.1. HERRAMIENTAS DE CÁLCULO PARA EL MÉTODO

El Algoritmo de Coordenadas de Áreas requiere, como se dijo, el cálculo del valor de áreas y volúmenes, de elementos y de porciones de éstos. Para realizar esta tarea, se cuenta únicamente con las coordenadas de los nodos de cada elemento dadas

por las Matrices de Procesamiento de Datos. Es por ello que se recurre a un principio del álgebra de vectores que utiliza el producto vectorial y el producto mixto como herramientas para calcular características geométricas.

El módulo del producto cruz (o vectorial) entre dos vectores coplanares es igual al área del paralelogramo que describen. Este concepto se utiliza para el cálculo de las áreas requeridas por el método y se ilustra en la Figura VI.8.

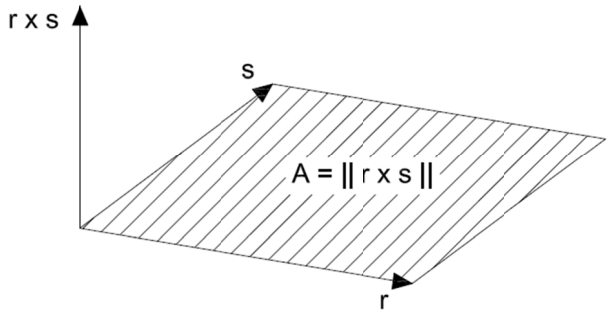


Figura VI.8: Representación Gráfica del Producto Vectorial, Iguala al Área del Paralelogramo

Por lo tanto, se tiene,

$$A = \|\boldsymbol{r} \times \boldsymbol{s}\| \qquad \text{(VI.3.1.1)}$$

Por otro lado, el álgebra de vectores indica que el producto mixto entre tres vectores es equivalente al volumen del paralelepípedo que forman. Esto es, siguiendo la Figura VI.9.

$$V = [\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{t}] = \boldsymbol{r} \cdot (\boldsymbol{s} \times \boldsymbol{t}) \qquad \text{(VI.3.1.2)}$$

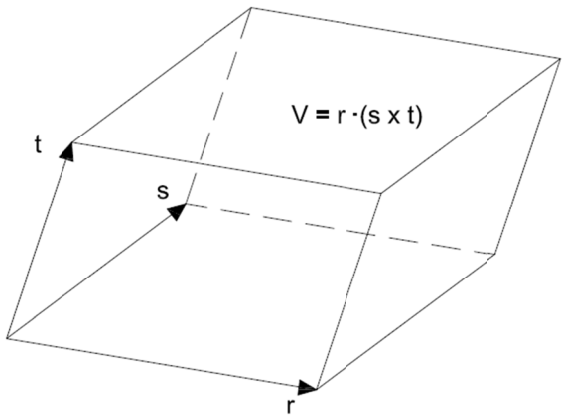


Figura VI.9: Representación Gráfica del Producto Mixto, Iguala al Volumen del Paralelepípedo

Con las expresiones (VI.3.1.1) y (VI.3.1.2) es posible calcular las áreas y volúmenes de virtualmente todos los elementos aquí tratados y sus fragmentos. Se requiere simplemente la correcta definición de los vectores intervinientes en cada producto y la interpretación geométrica de lo que se pretende calcular. A continuación

se presentan algunos métodos para calcular volúmenes y áreas de geometrías irregulares que serán los aplicados más tarde en la implementación del algoritmo.

ELEMENTOS PLANOS

En dos dimensiones se toma como caso principal el elemento cuadrilátero. Como lo indica la expresión (VI.3.5) es necesario calcular cuatro áreas triangulares, incluyendo al punto "P", y el área total del elemento independientemente de dicho punto. Además, los valores mencionados deben obtenerse sólo con las coordenadas de los nodos que forman el elemento. Luego, se tienen los dos casos siguientes:

- Área Triangular

Si son conocidas las coordenadas de los tres puntos que conforman un triángulo, es posible definir entre ellos, a través de la diferencia entre sus coordenadas, seis pares de vectores libres concurrentes. De este conjunto de vectores, sólo un par es necesario para calcular el producto vectorial y obtener el área del paralelogramo que describen. Finalmente, el área triangular que se pretende es la mitad de aquella que se calcula a partir del módulo del producto cruz entre los vectores. Se tiene entonces y de acuerdo a la Figura VI.10,

$$A_{\Delta} = \frac{\|\boldsymbol{r} \times \boldsymbol{s}\|}{2} \qquad \text{(VI.3.1.3)}$$

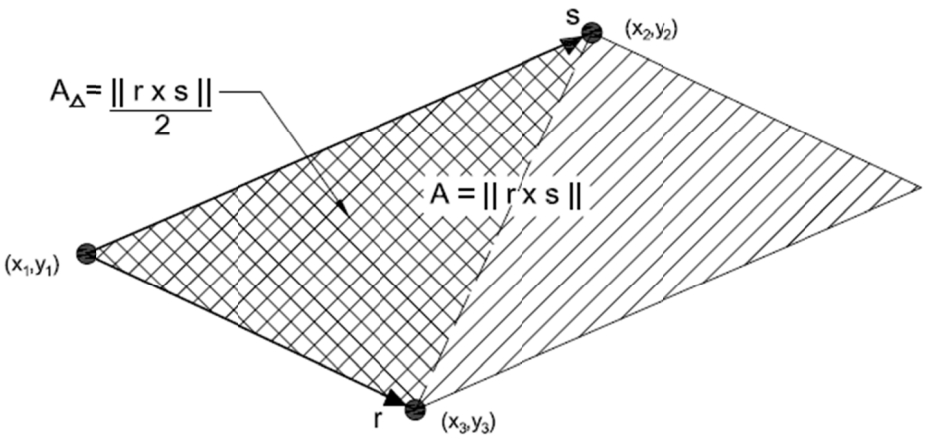


Figura VI.10: Cálculo de Área Triangular

- Área de un Cuadrilátero Irregular

Para obtener el área total de un elemento cuadrilátero, no basta con calcular un único producto vectorial puesto que, en la mayoría de los casos, el cuadrilátero no es un paralelogramo. En este sentido, la alternativa es dividir al elemento en dos triángulos a través de una línea media y repetir el proceso anterior para el cálculo del área. Resultará así que el área del elemento será la suma de las áreas de los dos triángulos que lo conforman,

$$A_{\square} = A_{\Delta_1} + A_{\Delta_2} = \frac{\|\boldsymbol{r_1} \times \boldsymbol{s_1}\|}{2} + \frac{\|\boldsymbol{r_2} \times \boldsymbol{s_2}\|}{2} \qquad \text{(VI.3.1.4)}$$

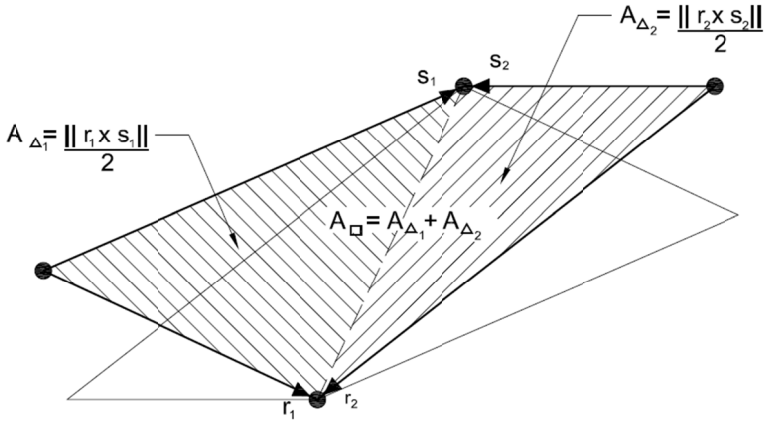


Figura VI.11: Cálculo del Área de un Cuadrilátero

ELEMENTOS TRIDIMENSIONALES

Los elementos tridimensionales son quizás algo más complejos de interpretar, pero su tratamiento y los conceptos que se aplican son análogos a los anteriores. Puede considerarse que el elemento hexaedro engloba a los demás elementos espaciales debido a que éstos pueden incluirse dentro de su geometría. Por lo tanto, el análisis se centra en los elementos de seis caras y puede hacerse extensivo a los restantes.

Siguiendo la expresión (VI.3.7), es necesario calcular seis volúmenes de pentaedros que incluyen el punto "P" y el volumen total del elemento independientemente del punto. Por lo tanto, se requerirán nuevamente dos planteos.

- Volumen de un Tetraedro

El tetraedro representa el fragmento más reducido de los elementos espaciales, es decir que todos los demás elementos pueden conformarse a partir de éste. Es por esto que cobra importancia poder calcular el volumen de un poliedro de cuatro caras a partir de las coordenadas de los puntos que lo forman.

Se mencionó anteriormente que el producto mixto entre tres vectores es equivalente al volumen del paralelepípedo que estos describen. Adicionalmente, el tetraedro definido por esos mismos vectores constituye un sexto del volumen del paralelepípedo mencionado y es precisamente así como se obtiene su volumen. Es decir,

$$V_{tetra} = \frac{\boldsymbol{r} \cdot (\boldsymbol{s} \times \boldsymbol{t})}{6} \qquad \text{(VI.3.1.5)}$$

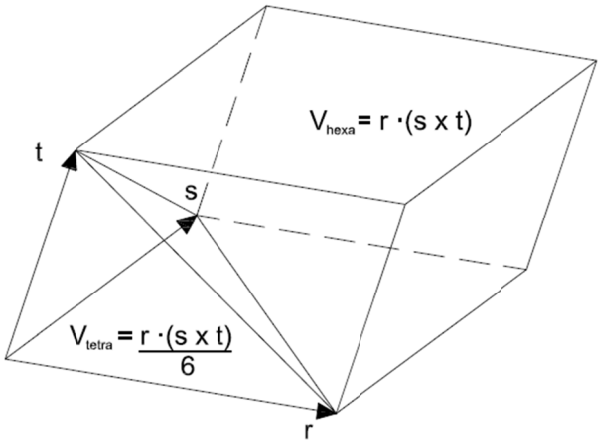


Figura VI.12: Volumen de un Tetraedro respecto del Volumen de un Paralelepípedo

- Volumen de un Elemento Hexaedro

Para obtener el volumen de un hexaedro irregular será necesario dividirlo en porciones cuyos volúmenes puedan ser calculados con la expresión (VI.3.1.5). Esta división consiste en definir seis subelementos, tetraedros, y repetir el proceso anterior. Esto es,

$$V_{hexa} = \frac{1}{6}\left[V_{tetra_1} + V_{tetra_2} + V_{tetra_3} + V_{tetra_4} + V_{tetra_5} + V_{tetra_6}\right] \qquad \text{(VI.3.1.6)}$$

$$V_{hexa} = \frac{1}{6}[\boldsymbol{r_1} \cdot (\boldsymbol{s_1} \times \boldsymbol{t_1}) + \boldsymbol{r_2} \cdot (\boldsymbol{s_2} \times \boldsymbol{t_2}) + \boldsymbol{r_3} \cdot (\boldsymbol{s_3} \times \boldsymbol{t_3}) + \boldsymbol{r_4} \cdot (\boldsymbol{s_4} \times \boldsymbol{t_4}) + \boldsymbol{r_5} \cdot (\boldsymbol{s_5} \times \boldsymbol{t_5}) + \boldsymbol{r_6} \cdot (\boldsymbol{s_6} \times \boldsymbol{t_6})]. \qquad \text{(VI.3.1.7)}$$

La Figura VI.13 ilustra la división que se realiza sobre el elemento hexaedro para calcular su volumen total.

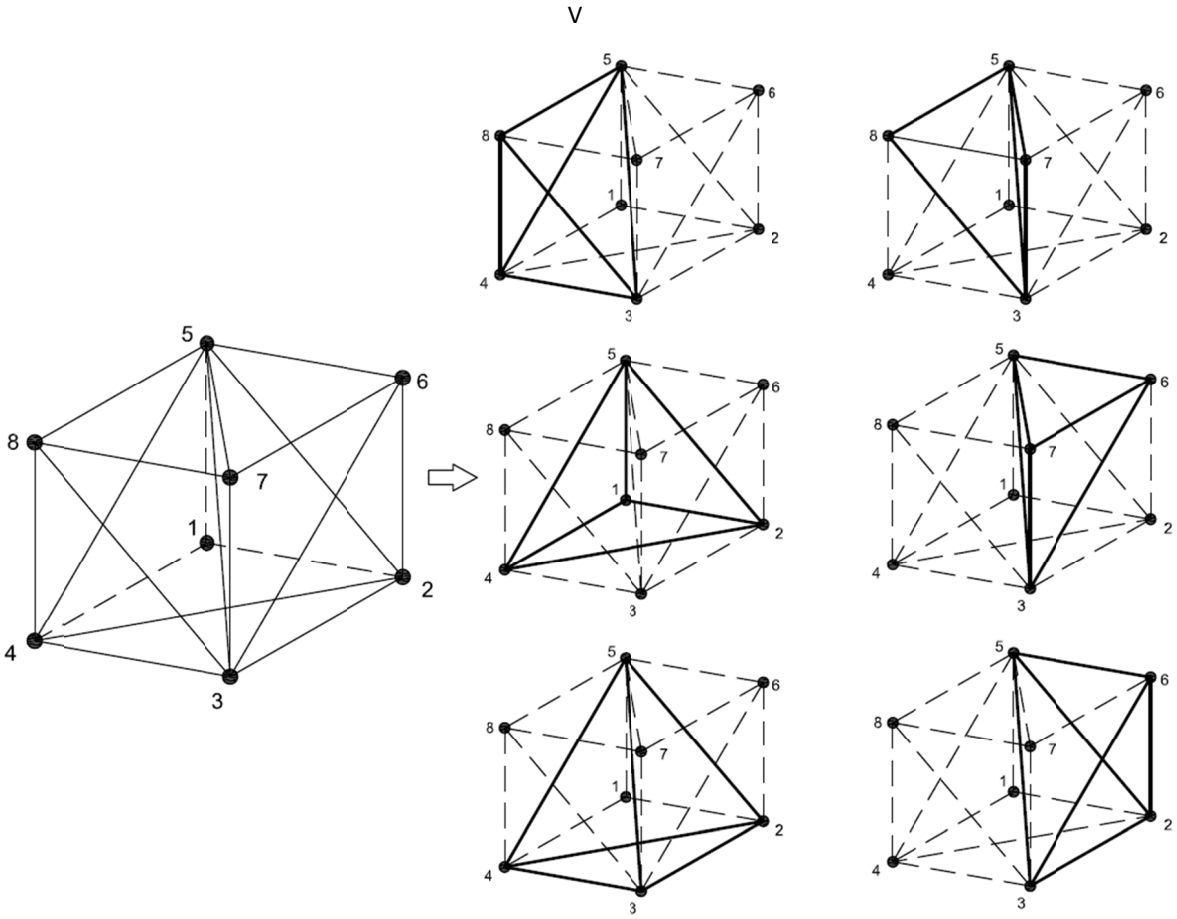


Figura VI.13: Hexaedro Fragmentado para el Cálculo de su Volumen Total

VI.3.2. IMPLEMENTACIÓN

La implementación de este método consiste nuevamente en dos ciclos anidados. El ciclo externo recorre los puntos a localizar y el interno, los elementos que se evaluarán según lo indicado por el Módulo de Selección. El ciclo externo conformará para cada punto a localizar el vector $\boldsymbol{Xp}$ con las coordenadas del mismo, que de aquí en adelante se identificará con el punto "P" antes mencionado. Luego se dará paso al ciclo interno que tomará el primer elemento de la lista para iniciar la evaluación.

El primer paso de la evaluación es la conformación de la matriz $[Xa]$ dada por la expresión (V.4.1.2).

Seguidamente, se da paso a la definición de los vectores que serán necesarios para calcular las áreas o volúmenes según sea el caso. El Algoritmo de Coordenadas de Área presenta una desventaja en este sentido, no es posible utilizar el mismo procedimiento para todos los tipos de elemento. Por lo tanto, debe identificarse con qué elemento se trabaja para poder determinar cuántos y qué vectores deben definirse. Este inconveniente se resuelve con la programación de una función tipo

menú que opere con la variable "ElementType" y active el código correspondiente al tipo de elemento en cuestión.

En el caso de un cuadrilátero en dos dimensiones se requieren como mínimo seis vectores distintos para poder efectuar todas las operaciones necesarias. En el proceso de definición de estos vectores, nuevamente, cobra importancia el ordenamiento nodal local. En la Figura VI.14 se representa un elemento cuadrilátero con un punto arbitrario en su interior, "P", y se indican una combinación de seis vectores que pueden utilizarse.

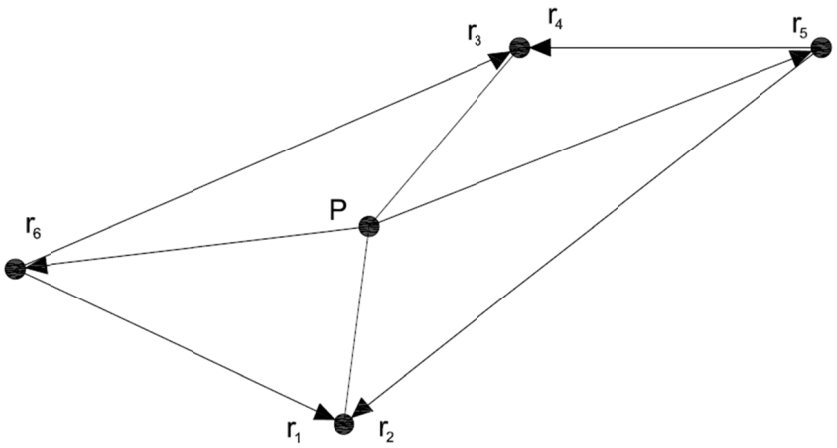


Figura VI.14: Seis Vectores Necesarios para un Cuadrilátero

En tres dimensiones, si se trabaja con un hexaedro, son requeridos al menos quince vectores para poder calcular los volúmenes necesarios para el proceso de localización. Nueve de estos vectores se conforman entre los nodos del elemento y en los seis restantes interviene el punto "P". La figura a continuación ilustra una posible combinación de vectores representados en cada división del elemento hexaedro.

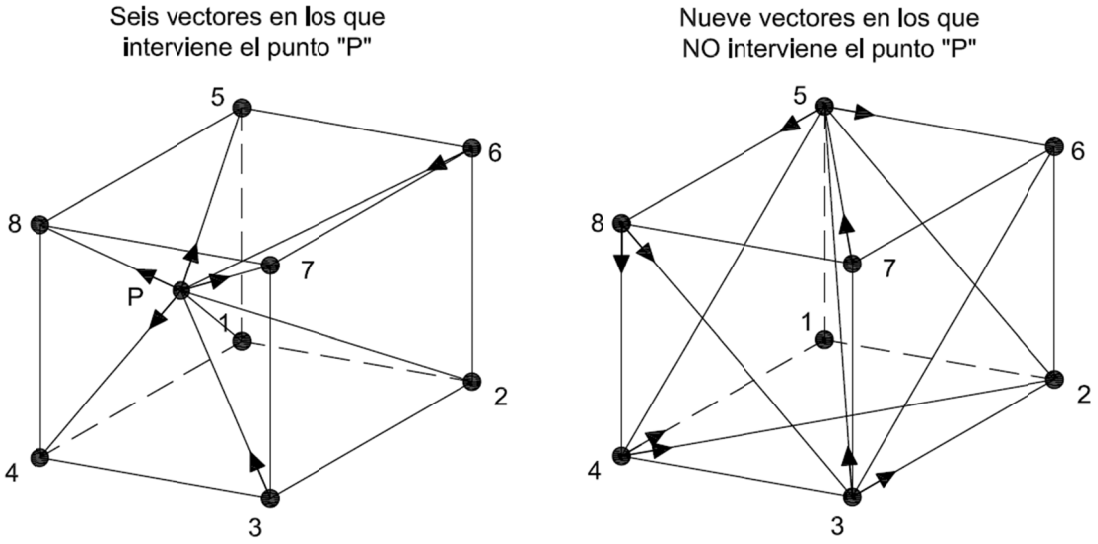


Figura VI.15: Quince Vectores Necesarios para un Hexaedro

El tercer paso comprende las operaciones para obtener los volúmenes y áreas de los elementos y sus porciones. Estos cálculos se realizan combinando los vectores antes definidos en los productos mixtos y productos vectoriales correspondientes. Se aplican aquí los conceptos dados en la sección VI.3.1 prestando especial cuidado al ordenamiento nodal y a la denominación utilizada en la creación de los vectores. Para simplificar la operatoria, es posible agrupar factores entre productos mixtos y generar con éstos variables intermedias que resultan repetitivas.

Como resultado de lo anterior se obtienen dos variables, que serán volúmenes en el caso tridimensional y áreas, en el bidimensional. Para los elementos planos, una de las variables será la suma de las áreas calculadas a partir de la división del elemento según el punto a localizar. La restante será la suma de las áreas calculadas sin la intervención del punto "P". En el espacio sucede algo similar, la primera variable provendrá de los cálculos que incluyen el vector $\boldsymbol{Xp}$, y la segunda de aquellos realizados independientemente de éste.

Así, el último paso consistirá en comparar las variables obtenidas para verificar si el punto es interno o no al elemento. La comparación se realiza teniendo en cuenta las expresiones (VI.3.5) y (VI.3.7), y arrojará un resultado afirmativo, si se produce la localización, o uno negativo, en el caso contrario. Para considerar posibles errores numéricos, se trabajará con un término de tolerancia que dará un rango a la igualdad en las comparaciones entre variables. Dicho término será igual a la tolerancia aplicada en otros métodos, resultando de manera simbólica,

$$\sum v_i = V \pm Err \qquad \text{(VI.3.1.6)}$$

$$Err = 0{,}0001. \qquad \text{(VI.3.1.6)}$$

Finalmente, deben mencionarse las acciones a tomar en caso de que el punto se localice o no en el elemento. En el segundo caso, no se requieren mayores tareas, si el elemento no contiene al punto el ciclo interno automáticamente realizará la evaluación del elemento siguiente. Sin embargo, si el punto a localizar es interior al elemento, la información obtenida debe ser incorporada a la Matriz de Localización y el ciclo interno deberá finalizarse. Así, el ciclo externo continuará con el siguiente punto a localizar.

VI.4. PUNTOS SIN LOCALIZAR

Una aclaración debe realizarse para, llegado el caso, aquellos puntos de la GA que por algún motivo no puedan ser ubicados en algún elemento de la ME del conjunto preseleccionado. Esta situación puede deberse a tres casos en los cuales el Módulo de Localización, cualquiera sea el algoritmo que utilice, finalizará sus ciclos sin asignar un elemento al punto en cuestión, a saber:

- El punto a localizar no se encuentra dentro de la ME.
- El elemento que contiene al punto no fue incluido en la preselección.
- Se produjo algún tipo de error o incompatibilidad en el proceso de localización.

Como se dijo en la sección IV.5, la Matriz de Localización se pre-dimensiona como un arreglo de ceros, lo que no es trivial puesto que el índice global de los elementos de la ME es siempre mayor o igual a uno. De este modo, si un punto no es localizado, el valor cero que contenía la Matriz de Localización, en la posición correspondiente, permanecerá inalterado. Por lo tanto, la presencia de un cero en la Matriz de Localización indicará que la localización no fue efectiva para ese punto y el caso deberá analizarse.

Ante la situación anterior, deberá evaluarse si es posible que el punto a localizar realmente sea externo a la ME. De no ser así, deberá ampliarse el conjunto dado por el Módulo de Selección mediante el cambio del algoritmo utilizado, siempre y cuando se haya descartado la posibilidad de fallas en los procesos.

Capítulo VII: Testeo y Medición de Performances

VII.1. Generalidades

El siguiente paso para alcanzar el objetivo del presente trabajo es efectuar la evaluación del funcionamiento de los algoritmos implementados. En este sentido, se pretenden dos tipos de análisis, el primero es de tipo comparativo entre cada algoritmo y los de su misma clase, es decir, de aquellos que cumplen la misma función. El segundo, hace referencia a la totalidad del Programa de Vinculación, esto es, realizar interpretaciones sobre la performance de las distintas variantes de cada módulo combinadas entre sí.

En términos generales, cualquier algoritmo que se desee implementar debe contar con dos cualidades, a saber, eficacia y eficiencia. La primera hará referencia a que el algoritmo efectivamente cumpla con su objetivo y, la segunda, a que lo haga del mejor modo posible. Para evaluar la eficacia de los métodos se utilizarán contextos relativamente sencillos, con mallas estructurales de pocos elementos y una cantidad reducida de puntos a localizar. El propósito de estas pruebas es tratar casos que sean manejables manualmente, permitiendo así constatar que los resultados obtenidos son correctos. En cuanto a la eficiencia y habiendo garantizado que los algoritmos funcionan, se generarán contextos más reales y, por ende, más complejos. Aquí se trabajará con modelos de alas de insectos que, no sólo poseen un mayor número de elementos, sino que su geometría es más complicada. También se utilizarán modelos de grillas aerodinámicas para proporcionar los puntos a localizar.

VII.2. Tipos de Evaluaciones

Dado que cada módulo del Programa de Vinculación cumple una función determinada y distinta, es necesario definir qué parámetros se considerarán para la evaluación en cada caso. En consecuencia, no tendrá sentido comparar, por ejemplo, el desempeño de un algoritmo para el Módulo de Selección con aquellos que corresponden al Módulo de Localización, más allá de que sea posible realizar conclusiones sobre la combinación de ambos. Sin embargo y debido a la naturaleza del contexto en el que se desarrolla el Programa de Vinculación, en todos los casos se pretende reducir virtualmente al mínimo el tiempo de ejecución.

Evidentemente, el tiempo de ejecución dependerá de distintos factores en cada uno de los módulos, dando múltiples posibles enfoques a cada proceso de testeo. Es por esto que, siempre que sea posible, se buscará determinar la relación entre los factores variables y el tiempo de ejecución de cada algoritmo. A continuación se presentan las características de los procesos de evaluación para los algoritmos aplicables a cada uno de los módulos.

VII.2.1. Evaluación del Módulo de Ingreso

El Módulo de Ingreso cuenta con un único sistema de implementación que, como se sabe, se ejecuta una única vez en el Programa de Vinculación. La velocidad del algoritmo del primer módulo dependerá casi exclusivamente de las características de las mallas que se procesen. Aunque, en menor medida, también será función de los métodos aplicados a los módulos siguientes. Por ejemplo, si se utiliza en el Módulo de Localización el Algoritmo de Coordenadas de Área, no se requerirá ejecutar la Biblioteca de Funciones de Forma Derivadas.

En definitiva, el objetivo de la evaluación del Módulo de Ingreso será obtener la relación entre el tiempo de ejecución y las dimensiones de la ME y la GA. Por lo tanto, se pretende una ley de variación entre, por un lado, la cantidad de nodos y elementos de la ME, y los puntos a localizar de la GA, y por otro, el tiempo empleado por el algoritmo.

VII.2.2. Evaluación del Módulo de Selección

El Módulo de Selección presenta quizás mayor dificultad a la hora de calificar su desempeño. Nuevamente, el tiempo de ejecución se convierte en el parámetro preponderante, pero también es menester medir la precisión de la selección realizada por el algoritmo.

La precisión del segundo módulo se medirá con la media aritmética de la cantidad de elementos que se evalúan para cada punto, antes de efectuarse la localización. Puesto de un modo más sencillo, dado que el Módulo de Selección asigna a cada punto a localizar un conjunto ordenado de elementos en los que posiblemente éste se encuentre, se desea conocer cuántos de éstos elementos deben evaluarse antes de encontrar aquél que efectivamente contiene al punto.

En cuanto al tiempo de ejecución, debe tenerse en cuenta que en algunas veces no podrá calcularse un tiempo neto para el Módulo de Selección. Este es el caso, por ejemplo, del Algoritmo ESUP que superpone ciclos de selección con ciclos de localización. En este sentido y para poder comparar resultados, se trabajará con el tiempo de selección para cada punto a localizar de forma individual.

VII.2.3. Evaluación del Módulo de Localización

El último módulo del programa no admite un análisis tan profundo y su evaluación estará dada por el tiempo de ejecución. En este sentido, se tomará un valor medio de tiempo por elemento evaluado.

Una consideración debe hacerse en este punto y es respecto a la restricción del Algoritmo de Coordenadas de Áreas en relación a elementos de órdenes mayores a uno. Debido a dicha restricción, los análisis comparativos de la performance de este algoritmo sólo podrán considerar elementos de primer orden.

VII.2.4. EVALUACIÓN DEL PROGRAMA DE VINCULACIÓN COMO CONJUNTO

Pasados los análisis individuales de cada módulo se puede evaluar el tiempo de ejecución de todos los módulos como un conjunto, con distintas combinaciones de algoritmos. Para esto, se seleccionarán combinaciones entre aquellos métodos que hayan mostrado mejor desempeño en el análisis individual, puesto que serán los que mejor resultado producirán en conjunto.

Las evaluaciones del Programa de Vinculación completo se harán sobre los casos más complejos, es decir, aquellos correspondientes a las modelos de alas de insectos. Pretendiendo simular una aplicación real del producto del presente trabajo.

VII.3. IMPLEMENTACIÓN

La implementación de los mecanismos de evaluación de algoritmos se realiza a partir de introducir pequeñas modificaciones a los códigos. Estas modificaciones consisten en incluir en determinados lugares del Programa de Vinculación sentencias marcadoras que generen variables de control con los datos requeridos.

En cuanto al tiempo de ejecución, se aplicarán dos métodos proporcionados por MATLAB que resultan de suma utilidad debido a su simplicidad y la claridad en los resultados que aportan. El primero de ellos es la función “tic-toc”, y el segundo la herramienta “Profiler”.

La función “tic-toc” permite contar el tiempo, en segundos, que demora la ejecución que se encuentre entre dos sentencias predeterminadas, pudiendo ubicarlas en cualquier parte del código. De este modo, si se requiere contabilizar el tiempo de un módulo en particular o de sólo un paso de un ciclo, se colocan las sentencias en las líneas colindantes al bloque de programa que se evalúa.

La utilidad de la herramienta “Profiler” radica en la posibilidad de conocer cómo se distribuye el tiempo de ejecución de un código entre sus distintas funciones. Es decir, el “Profiler” proporciona una lista de valores porcentuales de acuerdo al tiempo y consumo de recursos parcial de cada sentencia con la que cuenta un código. Esto permite, entre otras cosas, conocer cuáles son los bloques menos eficientes y registrar línea por línea su tiempo de ejecución.

Una consideración importante debe realizarse aquí respecto al tiempo de ejecución. La misma hace referencia a que la performance de cualquier programa se encuentra íntimamente ligada a la capacidad de procesamiento del hardware de soporte. Es por esto que es necesario que todas las ejecuciones se realicen en el mismo ordenador. Además, en todos los casos que se evalúen tiempos de

procesamiento, estos se presentarán de manera comparativa, ya sea aprovechando el virtuosismo de un parámetro adimensional, o bien de forma porcentual. De este modo, se evita supeditar la interpretación de resultados a factores ajenos al desempeño de los códigos desarrollados. En este sentido y siempre que se trabaje con un parámetro adimensional, es necesario indicar respecto de qué valor de referencia se realiza la adimensionalización.

En cuanto al recuento de evaluaciones, las mediciones se realizan introduciendo contadores que se incrementan conforme los ciclos del código avanzan. Estas variables se reservan en algunos casos como datos de salida, o bien, se exponen en consola para su análisis.

VII.4. MALLAS DE PRUEBA

Los testeos sobre los distintos algoritmos se llevarán a cabo con la ayuda de mallas especialmente confeccionadas para tal fin. Los casos más sencillos permiten obtener resultados sin lidiar con tiempos de ejecución excesivamente largos. Además, son adecuados para generar, a partir de una única geometría, mallas con distintas densificaciones, produciendo así resultados que son función únicamente de la dimensión de la malla procesada. Un ejemplo de esto son las mallas en dos dimensiones, que quizás revisten menor interés por cuanto no representan las aplicaciones que aquí se tratan, pero son adecuadas para un primer escalón de análisis. Por otro lado, este tipo de mallas, pueden utilizarse indistintamente como ME o GA, aportando una versatilidad que resulta provechosa.

Las mallas más complejas, en contraparte, presentan ejemplos más realistas y son más exigentes con los códigos, permitiendo realizar análisis más profundos. Estos casos se trabajan en tres dimensiones, con modelos específicos para mallas estructurales y grillas aerodinámicas.

VII.4.1. MALLAS EN DOS DIMENSIONES

Se utilizarán dos geometrías de mallas planas, una para elementos triángulos (presentados en la sección II.6) y otra para cuadriláteros, denominadas “CASO A” y “CASO B”, respectivamente. Estas mallas serán trabajadas con tres escalas de densificación cuando actúen como mallas estructurales. En particular el “CASO A”, tendrá dos densificaciones más para modelar grillas aerodinámicas. A continuación se presentan ambas geometrías y una tabla con los datos de cada una de ellas actuando como ME o GA.

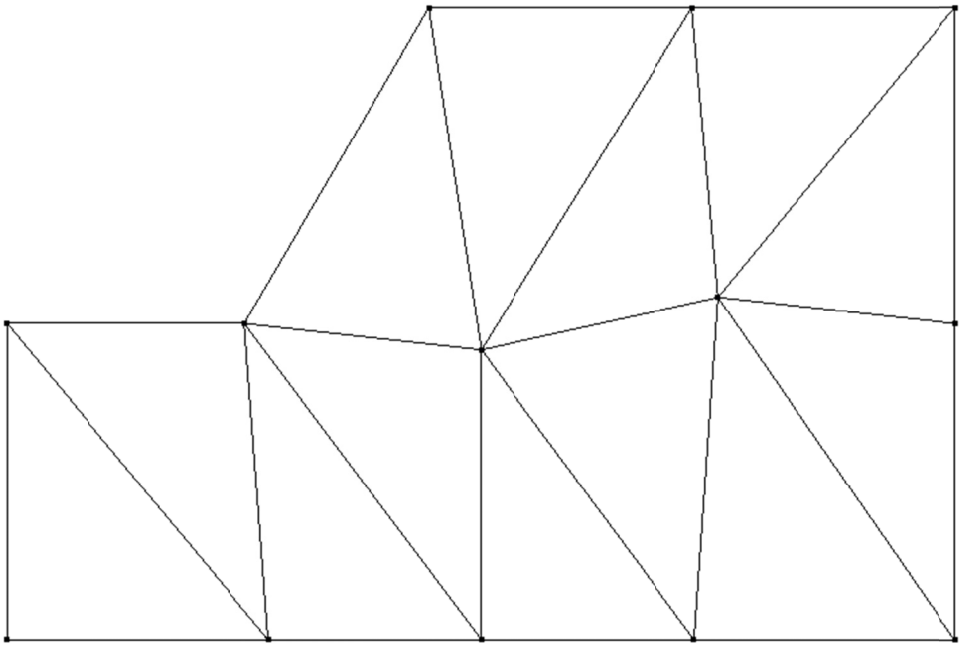


Figura VII.1: Geometría de la Malla "CASO A" con Primer Escala de Densificación (x1)

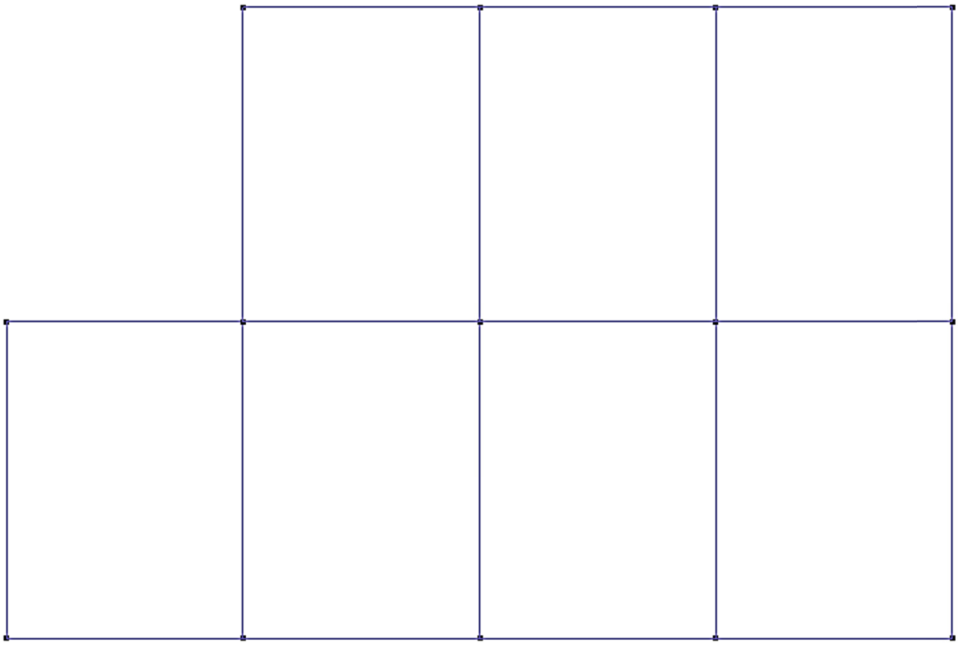


Figura VII.2: Geometría de la Malla "CASO B" con Primer Escala de Densificación (x1)

Denominación	Densificación	Aplicación	Nodos	Elementos	Tipo de Elementos
CASO A	x1	ME	13	13	Triángulos Primer Orden
	x3	ME	127	208	Triángulos Primer Orden
	x6	ME	6833	13312	Triángulos Primer Orden
CASO B	x1	ME	14	7	Cuadriláteros Primer Orden
	x2	GA	41	28	Cuadriláteros Primer Orden
	x3	ME	137	112	Cuadriláteros Primer Orden
	x4	GA	497	448	Cuadriláteros Primer Orden
	x6	ME	7361	7168	Cuadriláteros Primer Orden

Tabla VII.1: Datos de las Mallas "CASO A" y "CASO B"

VII.4.2. MALLAS EN TRES DIMENSIONES

Los ejemplos tridimensionales se dividen en dos tipos, el primero de ellos hace referencia a un concepto similar al presentado en el apartado anterior: una geometría sencilla que puede densificarse en distintas escalas. Sin embargo, y para guardar coherencia con las aplicaciones reales, será necesario generar un modelo de GA apropiado. Este caso se denomina "CASO C" y su geometría y características se exponen a continuación.

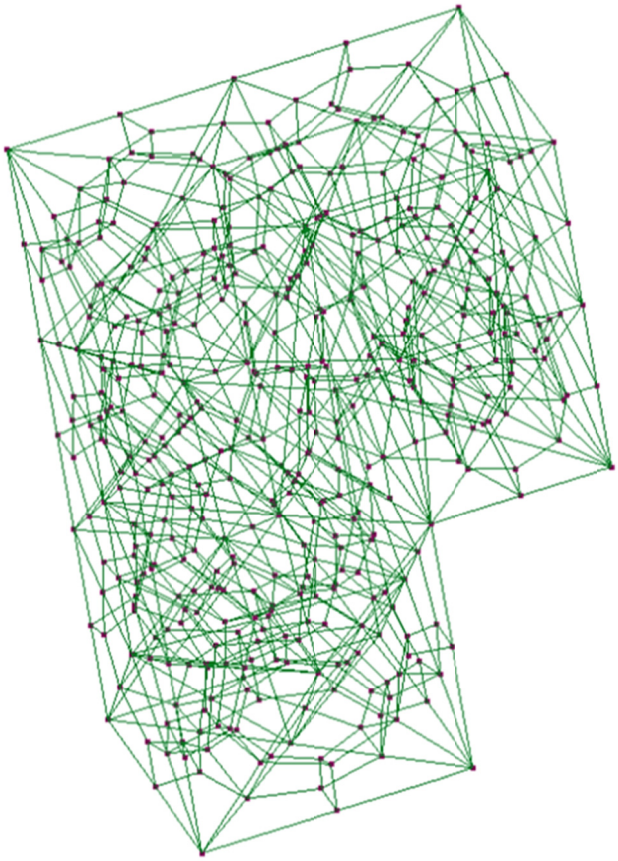


Figura VII.3: Geometría de la Malla "CASO C" con Primer Escala de Densificación (x1)

Denominación	Densificación	Aplicación	Nodos	Elementos	Tipo de Elementos
CASO C	x1	ME	455	312	Hexaedros de Segundo Orden
	x2	ME	2991	2496	Hexaedros de Segundo Orden
GA CASO C	x1	GA	203	176	Cuadriláteros Primer Orden
	x2	GA	757	704	Cuadriláteros Primer Orden

Tabla VII.2: Datos de la Mallas "CASO C"

Las mallas más interesantes de analizar los modelos de alas de insectos que fueron generados a partir del trabajo de S. Combes [5]. Se consideraron cuatro especies diferentes, Pepsis, Ischnura, Eristalis y Pieris, según su nombre científico.

Figura VII.4: Imágenes de Insectos Cuyas Alas se Modelaron, de izquierda a derecha: Eristalis, Pepsis, Ischnura, Pieris. (Gentileza: www.bugguide.net; www.austinbugs.com; www.austurnatura.com)

Los modelos de alas revisten una complejidad elevada y poseen cantidades de nodos y de elementos que alcanzan algunas decenas de miles. Nuevamente, los modelos de GA son generados especialmente para cada caso.

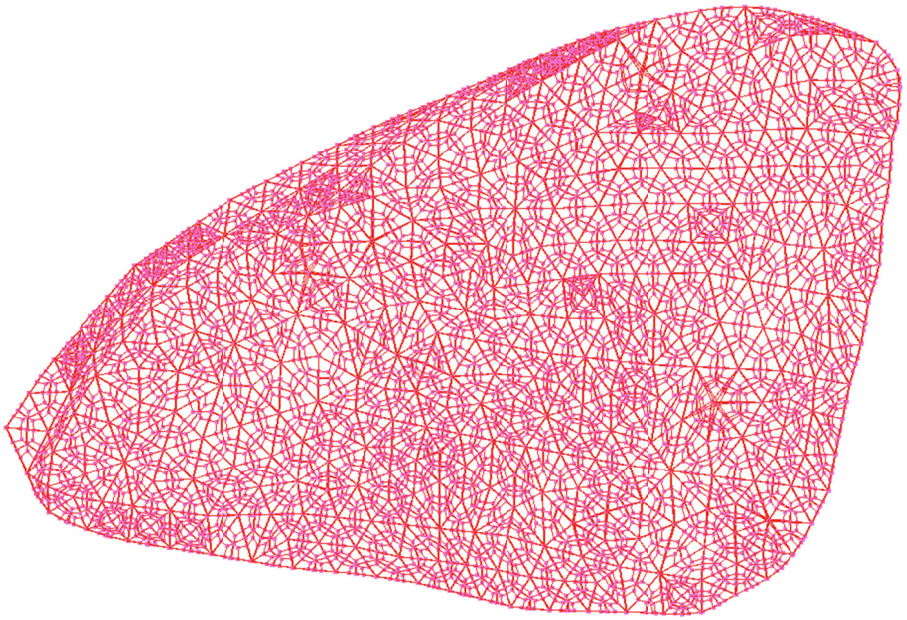


Figura VII.5: Malla Modelo de Ala de Pieris

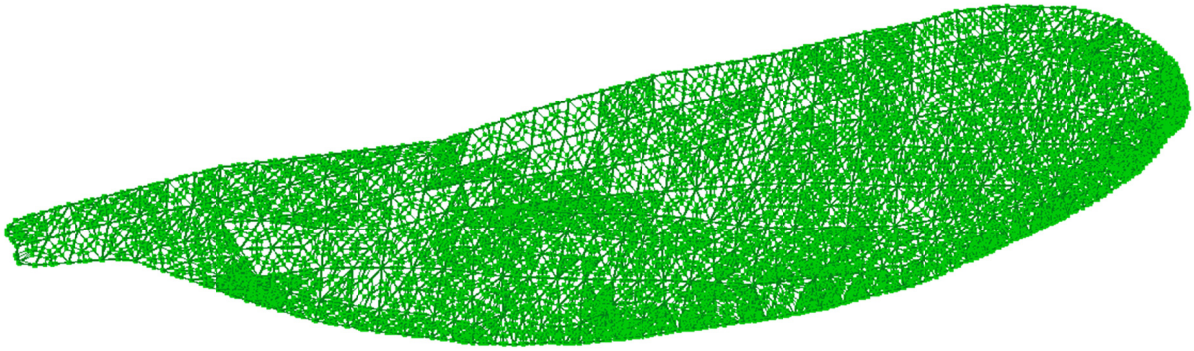


Figura VII.6: Malla Modelo de Ala de Ischnura

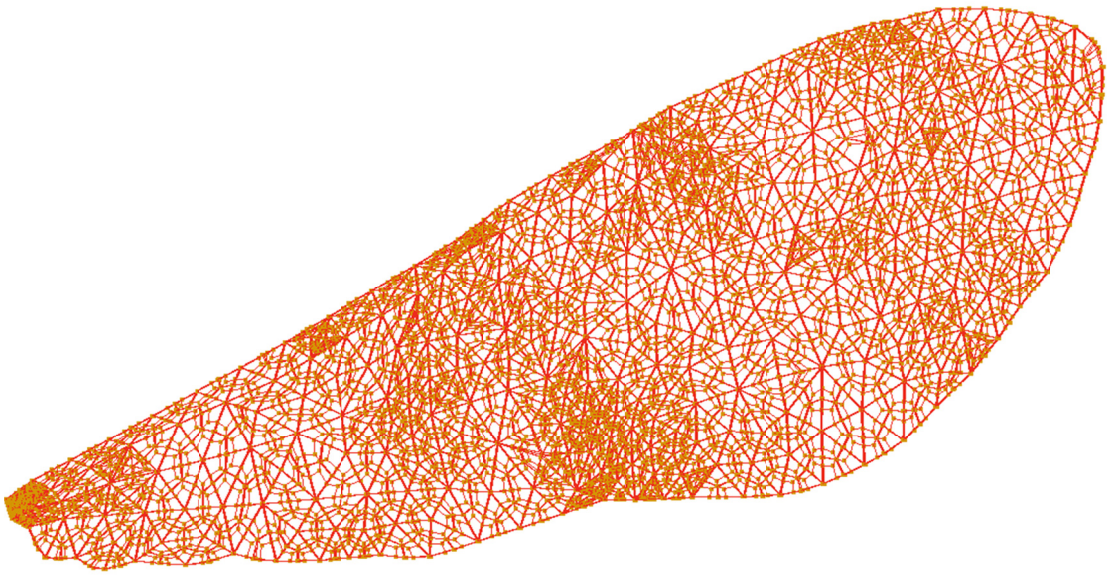


Figura VII.7: Malla Modelo de Ala de Pepsis

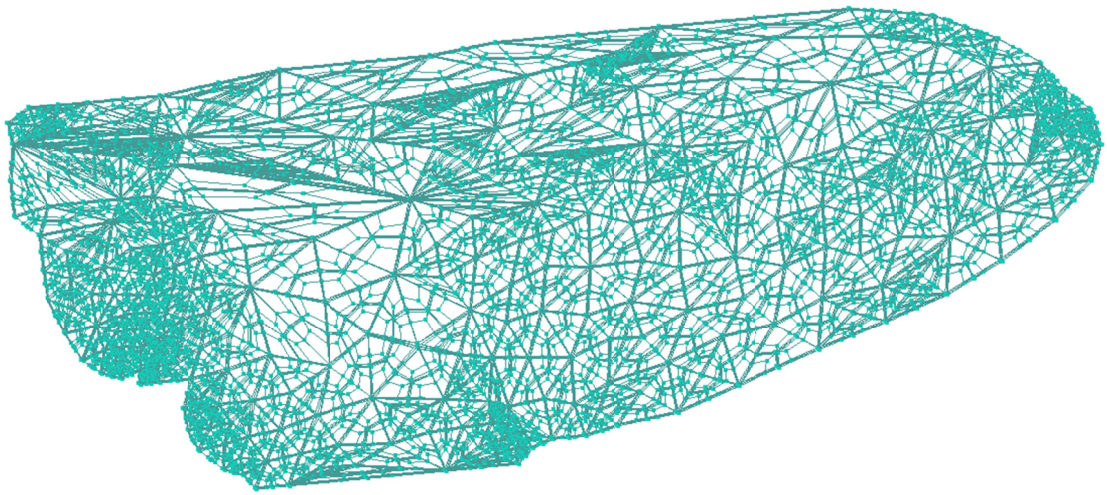


Figura VII.8: Malla Modelo de Ala de Eristalis

Denominación	Densificación	Aplicación	Nodos	Elementos	Tipo de Elementos
PIERIS	x1	ME	12205	9124	Hexaedros de Primer Orden
GA PIERIS	x1	GA	1582	1666	Cuadriláteros Primer Orden
PEPSIS	x1	ME	10141	19504	Hexaedros de Primer Orden
GA PEPSIS	x1	GA	1841	1742	Cuadriláteros Primer Orden
ISCHNURA	x1	ME	20637	30604	Hexaedros de Primer Orden

GA ISCHNURA	x1	GA	10738	10498	Cuadriláteros Primer Orden
ERISTALIS	x1	ME	9701	7284	Hexaedros de Primer Orden
GA ERISTALIS	x1	GA	1541	1470	Cuadriláteros Primer Orden

Tabla VII.3: Datos de las Mallas Modelos de Alas de Insectos

VII.5. Resultados y Análisis de Valores Obtenidos

VII.5.1. Módulo de Ingreso

La medición de performance del Módulo de Ingreso se realizó a partir de la toma de tiempos de procesamiento para las distintas mallas de prueba presentadas. Estos valores se adimensionalizaron respecto de la media aritmética de los tiempos contabilizados y se expresan como un parámetro comparativo, a saber,

$$\tau_{ej} = \frac{Tiempo\ Medido}{Tiempo\ Medio}. \qquad \text{(VII.5.1.1)}$$

Siendo "τ_{ej}" el tiempo de ejecución adimensional.

A continuación la tabla de resultados.

Malla Estructural	Grilla Aerodinámica	τ_{ej}
CASO A x1	CASO B x2	0,0403
CASO A x1	CASO B x4	0,0546
CASO A x3	CASO B x2	0,0739
CASO A x3	CASO B x4	0,0819
CASO A x6	CASO B x2	0,9408
CASO A x6	CASO B x4	0,9756
CASO B x1	CASO B x2	0,0369
CASO B x1	CASO B x4	0,0453
CASO B x3	CASO B x2	0,0603
CASO B x3	CASO B x4	0,0695
CASO B x6	CASO B x2	0,7211
CASO B x6	CASO B x4	0,7313
CASO C x1	GA CASO C x1	0,4339
CASO C x1	GA CASO C x2	0,4932
CASO C x2	GA CASO C x1	1,6401
CASO C x2	GA CASO C x2	1,7073
PIERIS	GA PIERIS	1,7764

PEPSIS	GA PEPSIS	3,0522
ISCHNURA	GA ISCHNURA	5,4939
ERISTALIS	GA ERISTALIS	1,5716

Tabla VII.4. Resultados de Evaluación del Módulo de Ingreso

A partir de los datos anteriores y teniendo en cuenta las características de las mallas y grillas dadas por la Tabla VII.4, puede verse que el incremento en la dimensión de éstas conlleva un aumento en el tiempo de ejecución. Sin embargo, un análisis más detallado indica que, la influencia de la dimensión de la ME es mucho más marcada que su contraparte referida a la GA. Relacionando el promedio de las variaciones y calculando las tendencias es posible confeccionar un gráfico como el siguiente donde se indica el claro dominio de la dimensión de la ME respecto de las variaciones del tiempo de ejecución.

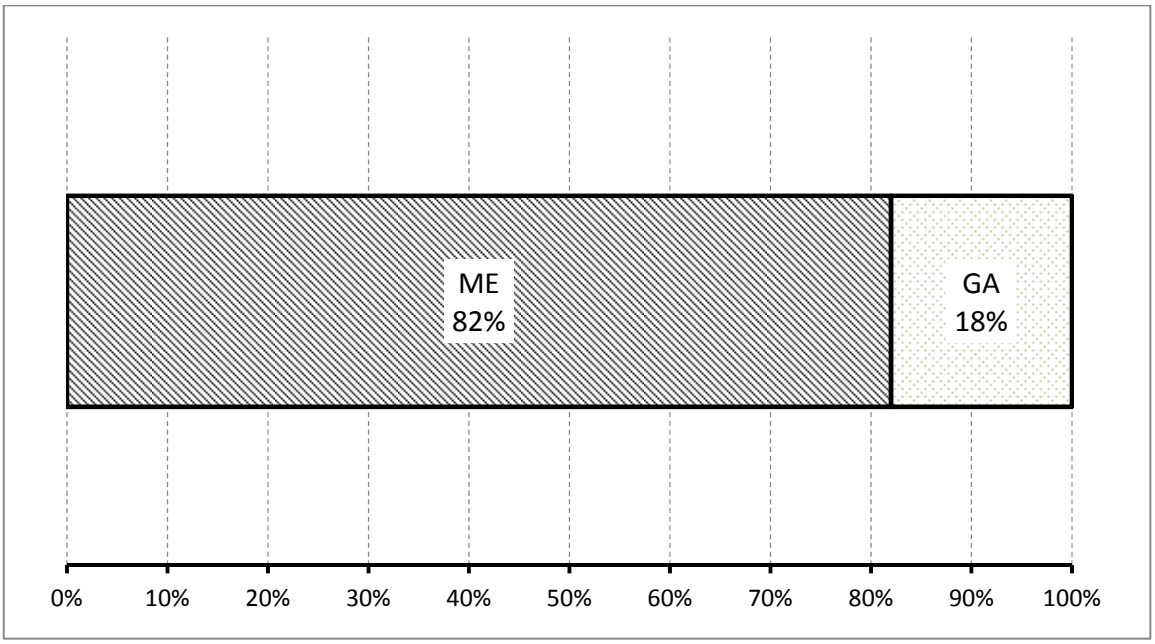


Figura VII.9: Representación de la Influencia del Aumento en la Dimensión de la ME y la GA, en el Tiempo de Ejecución

El resultado final muestra que un poco más del 80% del cambio en el tiempo de ejecución está dado por la variación de la dimensión de la malla de FEM, es decir, por el aumento de su cantidad de nodos y elementos. Este fenómeno encuentra motivos en que el Módulo de Ingreso sólo confecciona para la grilla del UVLM el Arreglo de Nodos y Coordenadas, presentado en la sección IV.3.1, mientras que, para la ME debe además generar el Arreglo IEN (visto en la sección IV.3.2).

VII.5.2. MÓDULO DE SELECCIÓN

Los algoritmos del segundo módulo del Programa de Vinculación son analizados en dos grupos. La división es debida a la gran diferencia que existe en el desempeño entre el Algoritmo de Lista y sus variantes, y los restantes métodos.

ALGORITMOS DE LISTA Y DE LISTA ALEATORIA

El Algoritmo de Lista, el Algoritmo de Lista Aleatoria Simple y el Algoritmo de Lista Aleatoria Múltiple fueron evaluados con un sistema particular debido a sus características. Este sistema, si bien difiere de los aplicados a otros algoritmos, permite obtener interpretaciones concluyentes evitando realizar ejecuciones excesivamente prolongadas que no aportan al análisis.

Con la ayuda de las mallas planas, sobre las que se generaron aleatoriamente puntos que simulasen a aquellos provenientes de una grilla aerodinámica, se midió el tiempo total de ejecución del Programa de Vinculación con el Algoritmo de Newton-Raphson para efectuar las localizaciones. Se utilizaron las tres densificaciones aplicables como ME tanto de la malla "CASO A" como de la malla "CASO B" y se proporcionaron 50 puntos a localizar. A continuación, los resultados.

Malla Estructural	Grilla Aerodinámica	Tiempo de Ejecución (segundos)		
		Lista	Lista Aleatoria Simple	Lista Aleatoria Múltiple
CASO A x1	50 Puntos Aleatorios	27,9	24,7	26,9
CASO A x3	50 Puntos Aleatorios	517,0	462,2	464,0
CASO A x6	50 Puntos Aleatorios	17179,1	15341,5	15219,7
CASO B x1	50 Puntos Aleatorios	13,9	13,7	14,2
CASO B x3	50 Puntos Aleatorios	225,5	279,5	253,0
CASO B x6	50 Puntos Aleatorios	18714,7	16785,6	14894,3

Tabla VII.5: Resultados de Evaluación de Algoritmos de Lista

Tal y como fueron descritos oportunamente, el Algoritmo de Lista y sus variantes constituyen métodos conocidos como "de fuerza bruta" y se caracterizan por ser extremadamente simples. Por otro lado, garantizan casi completamente la eficacia en su ejecución por lo que pueden considerarse virtualmente a prueba de fallos. No obstante, estos dos beneficios van notablemente en desmedro de su eficiencia. Prestando atención a la tabla anterior, se observa que el procesamiento de una malla plana con unos pocos miles de elementos requiere de algunas horas de trabajo, lo que hace sumamente impráctico aplicar estos proceso a modelos más realistas.

En adición, puede verse que el beneficio que se obtiene de la implementación del cambio de ordenamiento no es significativo. En efecto, si se comprende que a medida que la cantidad de nodos y elementos se incrementa un ordenamiento es tan buen candidato como cualquier otro, el sustento de estas modificaciones depende de una variable estadística aleatoria.

Ahora bien, estos algoritmos, a pesar de sus debilidades, son sumamente útiles en las etapas de desarrollo de otros métodos. Se mencionó anteriormente que el primer paso para la evaluación consistía en constatar el resultado de un algoritmo con una resolución visual del mismo caso pero, eventualmente, se alcanza un punto en el que este trabajo se vuelve impracticable. Es precisamente ahí donde cobran importancia estos métodos ya que, si se cuenta con el tiempo suficiente, permiten obtener resultados confiables a comparar con los de otros sistemas.

ALGORITMOS DE CENTROS DE MASA, ESUP Y ESUP ORDENADO POR CENTROS DE MASA

Los restantes algoritmos destinados al Módulo de Selección permiten un análisis algo más profundo. Aquí se evalúan dos aspectos de cada uno de ellos, uno referido al tiempo de ejecución y otro a la precisión en la selección.

El primer aspecto consiste en contabilizar el tiempo que transcurre para seleccionar los elementos a evaluar para un punto a localizar de la GA. Es decir, cuánto demora el algoritmo en confeccionar el conjunto de elementos que pasará al módulo siguiente para cada punto de la GA. Esta medida se materializa en un parámetro adimensional, $\bar{\tau}_{e_{GA}}$, que se calcula en relación al valor promedio, según indica la expresión (VII.5.1.1).

El aspecto restante a evaluar es la precisión con la que se escogen los elementos del conjunto. La medida para esta característica es el promedio del número de elementos que se evalúan antes de realizar una localización efectiva, para cada punto de la GA, dado por $\bar{e}_{e_{GA}}$.

Los resultados obtenidos de las mediciones se plasman en la Tabla VII.6.

Malla Estructural	Grilla Aerodinámica	Centros de Masa		ESUP		ESUP Ord. Por C.M.	
		$\bar{\tau}_{e_{GA}}$	$\bar{e}_{e_{GA}}$	$\bar{\tau}_{e_{GA}}$	$\bar{e}_{e_{GA}}$	$\bar{\tau}_{e_{GA}}$	$\bar{e}_{e_{GA}}$
CASO C x1	GA CASO C x1	0,1814	1,33	0,2290	2,754	1,7233	1,213
CASO C x1	GA CASO C x2	0,1941	1,293	0,2346	2,651	1,7741	1,167
CASO C x2	GA CASO C x1	0,2195	1,367	0,2445	2,889	2,0479	1,254
CASO C x2	GA CASO C x2	0,2208	1,45	0,2566	2,911	2,1481	1,273
PIERIS	GA PIERIS	0,3077	1,587	0,4382	2,513	1,8105	1,176
PEPSIS	GA PEPSIS	0,4155	1,421	0,4823	2,612	2,3368	1,254
ISCHNURA	GA ISCHNURA	0,8012	1,389	1,8068	2,313	4,1112	1,311
ERISTALIS	GA ERISTALIS	0,2979	1,495	0,3410	2,412	1,3770	1,202

Tabla VII.6: Resultados de Evaluación de Algoritmos para Módulo de Selección

En términos generales y en base a los valores anteriores, el Algoritmo ESUP Ordenado por Centros de Masa es el que produce menor cantidad de evaluaciones. Sin embargo, es este mismo algoritmo el que consume mayor tiempo para generar un conjunto de selección de elementos. Ante esto, la alternativa de Centros de Masa pareciera ser la más apropiada debido a que, a pesar de tener un promedio de evaluaciones algo mayor, el reducido tiempo de selección es alentador. Estas dos conclusiones muestran la gran influencia que tiene el orden que se asigna al conjunto en la performance de los algoritmos, percibida en el inferior desempeño del Algoritmo ESUP, el único que no cuenta con un sistema de ordenamiento.

Otra característica a tener en cuenta en los tres casos es la influencia de la dimensión de las mallas en el tiempo de ejecución. En primer lugar y respecto a la GA, pareciera que si se incrementan los puntos a localizar y se mantienen constantes los demás factores se produce una reducción en los dos parámetros de medida. Esta disminución no debe ser considerada como tal porque encuentra motivos en la naturaleza del indicador utilizado para la medición. Al calcular el valor promedio para cada punto de la GA, un incremento en la cantidad de puntos implica un denominador mayor en la obtención tanto de $\bar{\tau}_{e_{GA}}$ como de $\bar{e}_{e_{GA}}$. Por otro lado y de manera más intuitiva, un incremento en la dimensión de la ME, trae aparejado un incremento en los parámetros.

VII.5.3. MÓDULO DE LOCALIZACIÓN

Con el propósito de poder comparar los desempeños de los distintos algoritmos del último módulo del Programa de Vinculación, se utilizó como medida el tiempo de ejecución correspondiente a cada evaluación individual. Esta cifra fue calculada introduciendo dos variables de control en los códigos. Una de ellas corresponde a un contador de tiempo que aísla el ciclo interno del Módulo de Localización (ver secciones VI.2.2 y VI.3.2) y es acumulativo en sí mismo. La variable restante es un contador iterativo que, se incrementa en una unidad cada vez que se evalúa un elemento. De esta manera, los valores de salida serán el tiempo total empleado en todas las evaluaciones y la cantidad total de evaluaciones efectuadas.

El objetivo último es conocer el tiempo medio por evaluación, que representa un parámetro adecuado para un análisis comparativo. El indicador empleado es un número adimensional identificado como $\bar{\tau}_{ev_{ME}}$. Nuevamente, se toma como referencia para prescindir de las unidades el valor medio de las lecturas, según lo indica la expresión (VII.5.1.1).

Malla Estructural	Grilla Aerodinámica	Coordenadas de Área	Newton-Raphson	Newton-Raphson con Reducción
		$\bar{\tau}_{ev_{ME}}$	$\bar{\tau}_{ev_{ME}}$	$\bar{\tau}_{ev_{ME}}$
CASO C x1	GA CASO C x1	0,0050	1,2897	1,1957
CASO C x1	GA CASO C x2	0,0058	1,5051	1,3954

CASO C x2	GA CASO C x1	0,0063	1,6173	1,4994
CASO C x2	GA CASO C x2	0,0064	1,6418	1,5221
PIERIS	GA PIERIS	0,0059	1,5167	1,4061
PEPSIS	GA PEPSIS	0,0063	1,6173	1,4994
ISCHNURA	GA ISCHNURA	0,0065	1,6908	1,5675
ERISTALIS	GA ERISTALIS	0,0060	1,5503	1,4372

Tabla VII.7: Resultados de Evaluación de Algoritmos para Módulo de Localización

Las cifras en la tabla anterior marcan una notable diferencia entre la velocidad del Algoritmo de Coordenadas de Área y las dos variables del Algoritmo de Newton-Raphson. Puede inferirse que, al tratarse de un método iterativo, el Algoritmo de Newton-Raphson resuelve más de una vez un sistema de ecuaciones, que si bien es sencillo, implica un número de operaciones que se multiplican en cada paso. Mientras tanto, el Algoritmo de Coordenadas de Área realiza operaciones directas que se ejecutan una única vez por evaluación, lo que incrementa su velocidad. Sin embargo, no debe olvidarse la fuerte restricción respecto del orden de los elementos de la ME que posee el Algoritmo de Coordenadas de Área, que disminuye sensiblemente su rango de aplicación.

Otra diferencia, quizás menos significativa, es la que existe al aplicar la Reducción de Elementos al Algoritmo de Newton-Raphson. Las mejoras en los tiempos rondan el 10% en contrapartida a que sólo se aplican a un cierto tipo de elementos.

VII.5.4. PROGRAMA DE VINCULACIÓN COMPLETO

Teniendo presente los datos y consideraciones anteriores se da lugar a la evaluación del Programa de Vinculación como un todo. Para esto se tomaron dos de los modelos de alas antes presentados, Pieris e Ischnura, y se computó el tiempo total de ejecución para distintas combinaciones de algoritmos.

El propósito de estas pruebas es tener una idea general de cómo se desempeñan los algoritmos de cada módulo cuando se interrelacionan y formar un criterio de selección para futuras aplicaciones. Nuevamente, se expresan los resultados, obtenidos de la implementación de una variable contadora de tiempo que abarca el código completo, en relación al valor medio de las mediciones, a saber:

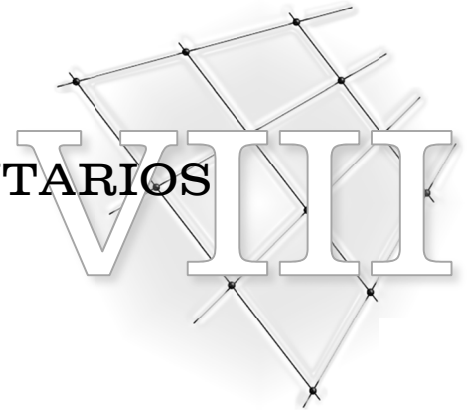
Malla Estructural	Grilla Aerodinámica	Algoritmos			τ_{ej}
		Módulo de Ingreso	Módulo de Selección	Módulo de Localización	
PIERIS	**GA PIERIS**	Ingreso	Centros de Masa	Coordenadas de Área	0,0049
				Newton-Raphson	0,2992
			ESUP	Coordenadas de Área	0,0053
				Newton-Raphson	0,3174

			ESUP Ordenado por Centros de Masa	Coordenadas de Área	0,0049
				Newton-Raphson	0,3054
ISCHNURA	**GA ISCHNURA**	Ingreso	Centros de Masa	Coordenadas de Área	0,0566
				Newton-Raphson	3,5051
			ESUP	Coordenadas de Área	0,0577
				Newton-Raphson	3,7057
			ESUP Ordenado por Centros de Masa	Coordenadas de Área	0,0557
				Newton-Raphson	3,6821

Tabla VII.8: Resultados de Evaluación de Algoritmos para Módulo de Localización

Los números obtenidos son coherentes con los análisis individuales de cada módulo. Las posibles variantes para el Módulo de Selección no generan alteraciones notables en la velocidad del programa. Mientras que, la diferencia entre los algoritmos de localización es marcada.

Capítulo VIII: Comentarios Finales



VIII.1. Generales

Durante todas las etapas del proceso de realización del Trabajo Final de Grado, surgieron diversas situaciones en las cuales fue preciso aportar soluciones criteriosas. En definitiva, la motivación de todo el desarrollo proviene de una dificultad real que afecta a los problemas de aeroelasticidad computacional en cuanto a la interacción fluido-estructura. Por lo tanto, es necesario en este punto adoptar un enfoque retrospectivo para dilucidar si dicha motivación fue satisfecha y en qué medida el objetivo propuesto en las notas introductorias fue alcanzado.

A pesar de esto, concluir sólo con un balance evaluativo no es suficiente. Si se comprende que el trabajo realizado contribuye en alguna medida al desarrollo de un campo más amplio, el capítulo final resulta oportuno para esbozar dicha contribución. La cual, en definitiva, será un conjunto de recomendaciones provenientes de la experiencia adquirida durante las etapas de realización.

Un tercer punto a plantear es el de generar motivaciones adicionales. La intención es aportar un componente al contexto en el cual se realicen otras incursiones que enriquezcan los temas aquí tratados. De este modo, se procura un abanico de posibilidades para trabajos futuros mencionando aquellas áreas en las que se podría ahondar el análisis.

VIII.2. Cumplimentación

El contexto descrito en la parte inicial del trabajo bosqueja un área de estudio muy dinámica, lo que constituye una fuente permanente de problemáticas a resolver. Aislando una porción de este contexto, se planteó un objetivo específico en relación a la interacción entre mallas estructurales de FEM y grillas aerodinámicas de ULVM. Es esta especificidad en el propósito de este trabajo la que posibilita afirmar que se alcanzaron las metas propuestas.

El Programa de Vinculación es el respaldo de todo el proceso de realización, que tiene como producto un conjunto razonable de algoritmos con algunas posibles variantes. Así, la implementación de métodos disímiles aporta versatilidad al código

principal y sus módulos brindando la posibilidad de adaptarlos a distintas aplicaciones prácticas.

Por otro lado, la evaluación comparativa de códigos contribuye a la formación de un criterio de selección de métodos a implementar en cada problemática. De esta manera, se sientan bases o principios a los cuales recurrir si no se cuenta con otra directiva. De ahí que la importancia de este aporte sea relativa ya que algunas conclusiones tienen una validez más amplia que otras.

Por último, un desarrollo no es trascendente si no genera opciones de profundizar los temas que trata. Luego, en aspectos tales como la utilización del Dominio de Origen como método de localización, la inclusión de conceptos físico-geométricos en el análisis de elementos, el orden en los conjuntos de selección, el modelado de casos reales y la combinación de criterios de selección, entre otros, se fijan puntos de partida para vastos desarrollos subsiguientes.

VIII.3. RECOMENDACIONES

La interpretación de los resultados de las evaluaciones y la experiencia adquirida durante la realización permiten dictar ciertas premisas que resultan de aplicación general. Como primera medida no debe perderse de vista la emisión de un juicio de valor. En todo ámbito de la ciencia y la ingeniería se transita entre una variedad de opciones que derivan en la toma de decisiones de compromiso. La multitud de contextos que pueden presentarse hace virtualmente imposible tomar determinaciones universalmente válidas, por lo que el criterio adecuadamente formado se valora por sobre todas las cosas. En este sentido, ningún método o sistema puede considerarse infalible, debe recordarse que no se trabajó sobre conceptos axiomáticos sino sobre razonamientos deductivos que mostraron resultados favorables en los casos aplicados.

En un tratamiento más práctico y en busca de la eficiencia como premisa del Programa de Vinculación, se recomienda fuertemente el uso del Módulo de Selección, en cualquiera de sus variantes. Los resultados muestran claramente que incluso los métodos más sencillos son altamente efectivos en la reducción de los tiempos de ejecución. En contraparte, la implementación de algoritmos de fuerza bruta debe ser evitada a menos que se trabaje en una etapa inicial de validación.

La última recomendación hace referencia a un tema que no fue tratado en los capítulos precedentes: las precauciones a tomar para la generación de mallas. Es de conocimiento común que el FEM exige ciertos lineamientos en la confección de mallas para garantizar la convergencia de los cálculos. Pero, en el ámbito de lo hasta aquí tratado, la utilización de elementos angulosos, la presencia de distorsiones fuertes y la densificación excesivamente concentrada, pueden generar errores inesperados en los algoritmos. Es por esto que la recomendación es procurar mallas ordenadas, sin sobresaltos y lo más regulares posibles, para asegurar de cierto modo el funcionamiento de todos los métodos.

VIII.4. TRABAJOS A FUTURO

Anteriormente se habló de la intención de generar posibles espacios para desarrollos subsiguientes, de estos, la confección de nuevos algoritmos es quizás el más tangible. Se mostró que pueden incorporarse teorías y conceptos de varias naturalezas a la hora de generar sistemas que apliquen a cualquiera de las etapas del código principal, algunos con mejores resultados que otros, pero no es sino hasta que se realizan las evaluaciones correspondientes que esto puede afirmarse.

Otro pendiente se constituye a partir de los tipos de elementos. En la mayoría de las aplicaciones se trató con elementos cuadriláteros y hexaedros, dejando de lado una amplia gama de variantes. Es cierto que muchos de los algoritmos funcionan sin problemas para diversos tipos de elementos, pero algunos otros no cuentan con tal característica. Por tanto, resultaría provechoso hacer una ampliación de los algoritmos a aquellos elementos que no fueron considerados.

El uso de MATLAB como soporte de programación reduce, por sus características, las posibilidades reales de aplicación de los algoritmos. Si bien, un lenguaje de alto nivel es sumamente útil en la etapa de desarrollo y evaluación, sería deseable traducir los códigos a lenguajes de niveles más bajos que se acostumbran en el uso de programas de FEM, como FORTRAN o C++.

Una mejora adicional puede realizarse en referencia a la programación. A pesar que el propósito planteado para el trabajo dista del perfeccionamiento en la confección de software, se estima que una optimización netamente computacional podría incrementar hasta un 30% las performances obtenidas. A este respecto, también podría implementarse una interface gráfica más amigable con el usuario que permita visualizar tanto las mallas y las grillas que se procesan, así como también los resultados.

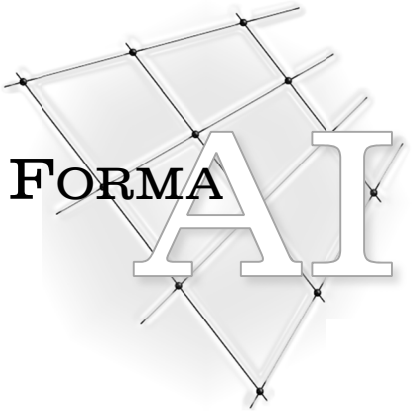
El incremento en la exigencia de los procesos de evaluación puede representar otra línea de trabajo a seguir. Para conocer fehacientemente en qué casos un algoritmo puede no funcionar correctamente puede recurrirse la generación de contextos en los cuales se someta a los algoritmos a situación críticas. La búsqueda fallas o incompatibilidades derivará en métodos más robustos y de mayor aplicabilidad.

Por último y no por eso menos importante, profundizar sobre el procesamiento de la GA podría implicar sensibles aumentos en la eficiencia. Todos los métodos implementados trabajan sobre la ME para alcanzar su objetivo, mientras que, como se dijo, la grilla de ULVM se traduce como un conjunto de puntos. Sin embargo, generar Matrices de Procesamiento de Datos más complejas, arreglos similares a ESUP o alguno de los otros algoritmos propuestos por R. Löhner [4] sobre la grilla aerodinámica permitiría procesar puntos cercanos o vinculados en conjunto y, posiblemente, sea una fuente de disminución del número de evaluaciones de localización.

VIII.5. CIERRE

Aquí concluye una etapa de algunos meses de arduo trabajo que materializa el final de una carrera de grado universitaria. Valorando lo aprendido, reconociendo algunos errores y atesorando la experiencia adquirida, fue cuantioso el provecho obtenido de todo este desarrollo. De esta manera y con la esperanza de haber hecho un aporte valorable al tema tratado, se da por finalizado el trabajo.

Anexo I: Funciones de Forma



A lo largo de los capítulos que tratan sobre las propiedades de los elementos utilizados en el FEM, se hizo mención a sus Funciones de Forma, las cuales juegan un rol preponderante en la aplicación de los diversos métodos y algoritmos. Las Funciones de Forma a las que se hace referencia son aquellas que emergen de la aplicación del Cuadrilátero Bilineal y el Hexaedro Trilineal como Dominios de Origen, para casos en dos y tres dimensiones respectivamente.

A continuación y a modo de complemento de los temas tratados, se presentan las Funciones de Forma de algunos elementos comúnmente utilizados. Las cuales se enlistan en vectores según se introdujo en la expresión (II.5.1.12).

A I.1 Elementos Triángulos

Triángulo de Primer Orden (3 nodos)

$$\boldsymbol{N}(\boldsymbol{\xi},\boldsymbol{\eta}) = \begin{Bmatrix} N_1(\xi,\eta) \\ N_2(\xi,\eta) \\ N_3(\xi,\eta) \end{Bmatrix} = \frac{1}{2}\begin{Bmatrix} (\eta-1)*(\xi-1) \\ -(\eta-1)*(\xi+1) \\ (\eta+1) \end{Bmatrix} \qquad \text{(AI.1.1)}$$

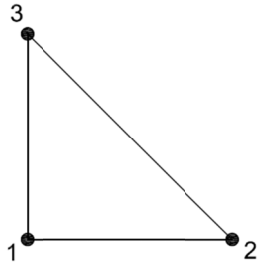


Figura A I.1: Triángulo de Primer Orden con Tres Nodos

Triángulo de Segundo Orden (6 nodos)

$$\boldsymbol{N}(\boldsymbol{\xi},\boldsymbol{\eta}) = \begin{Bmatrix} N_1(\xi,\eta) \\ N_2(\xi,\eta) \\ N_3(\xi,\eta) \\ N_4(\xi,\eta) \\ N_5(\xi,\eta) \\ N_6(\xi,\eta) \end{Bmatrix} = \frac{1}{2} \begin{Bmatrix} \xi * \eta * (\eta - 1) * (\xi - 1) * {}^{1}/_{2} \\ \xi * \eta * (\eta - 1) * (\xi + 1) * {}^{1}/_{2} \\ \eta * (\eta + 1) \\ -\big(\eta * (\eta - 1) * (\xi - 1) * (\xi + 1)\big) \\ -\big(\xi * (\eta - 1) * (\eta + 1) * (\xi + 1)\big) \\ -\big(\xi * (\eta - 1) * (\eta + 1) * (\xi - 1)\big) \end{Bmatrix} \quad \text{(AI.1.2)}$$

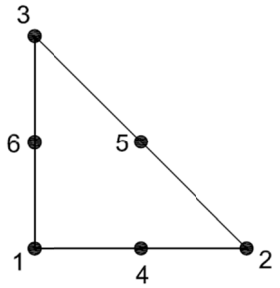


Figura A I.2: Triángulo de Segundo Orden con Seis Nodos

A I.2 ELEMENTOS CUADRILÁTEROS

CUADRILÁTERO DE PRIMER ORDEN (4 NODOS)

$$\boldsymbol{N}(\boldsymbol{\xi},\boldsymbol{\eta}) = \begin{Bmatrix} N_1(\xi,\eta) \\ N_2(\xi,\eta) \\ N_3(\xi,\eta) \\ N_4(\xi,\eta) \end{Bmatrix} = \frac{1}{2} \begin{Bmatrix} (\eta - 1) * (\xi - 1) \\ -(\eta - 1) * (\xi + 1) \\ (\eta + 1) * (\xi + 1) \\ -(\eta + 1) * (\xi - 1) \end{Bmatrix} \quad \text{(AI.2.1)}$$

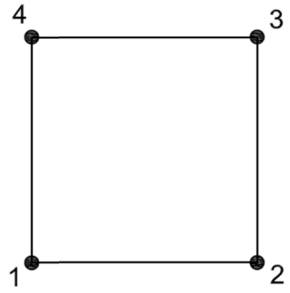


Figura A I.3: Cuadrilátero de Primer Orden con Cuatro Nodos

CUADRILÁTERO DE SEGUNDO ORDEN INCOMPLETO (8 NODOS)

$$\boldsymbol{N(\xi,\eta)} = \begin{Bmatrix} N_1(\xi,\eta) \\ N_2(\xi,\eta) \\ N_3(\xi,\eta) \\ N_4(\xi,\eta) \\ N_5(\xi,\eta) \\ N_6(\xi,\eta) \\ N_7(\xi,\eta) \\ N_8(\xi,\eta) \end{Bmatrix} = \frac{1}{2} \begin{Bmatrix} \xi * \eta * (\eta - 1) * (\xi - 1) * {}^1/_2 \\ \xi * \eta * (\eta - 1) * (\xi + 1) * {}^1/_2 \\ \xi * \eta * (\eta + 1) * (\xi + 1) * {}^1/_2 \\ \xi * \eta * (\eta + 1) * (\xi - 1) * {}^1/_2 \\ -\big(\eta * (\eta - 1) * (\xi - 1) * (\xi + 1)\big) \\ -\big(\xi * (\eta - 1) * (\eta + 1) * (\xi + 1)\big) \\ -\big(\eta * (\eta + 1) * (\xi - 1) * (\xi + 1)\big) \\ -\big(\xi * (\eta - 1) * (\eta + 1) * (\xi - 1)\big) \end{Bmatrix} \qquad \text{(AI.2.2)}$$

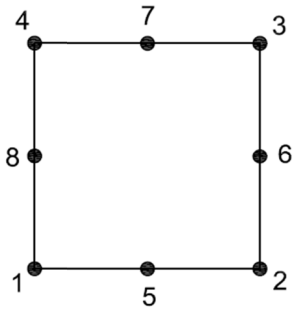


Figura A I.4: Cuadrilátero de Segundo Orden Incompleto con Ocho Nodos

CUADRILÁTERO DE SEGUNDO ORDEN COMPLETO (9 NODOS)

$$\boldsymbol{N(\xi,\eta)} = \begin{Bmatrix} N_1(\xi,\eta) \\ N_2(\xi,\eta) \\ N_3(\xi,\eta) \\ N_4(\xi,\eta) \\ N_5(\xi,\eta) \\ N_6(\xi,\eta) \\ N_7(\xi,\eta) \\ N_8(\xi,\eta) \\ N_9(\xi,\eta) \end{Bmatrix} = \frac{1}{2} \begin{Bmatrix} \xi * \eta * (\eta - 1) * (\xi - 1) * {}^1/_2 \\ \xi * \eta * (\eta - 1) * (\xi + 1) * {}^1/_2 \\ \xi * \eta * (\eta + 1) * (\xi + 1) * {}^1/_2 \\ \xi * \eta * (\eta + 1) * (\xi - 1) * {}^1/_2 \\ -\big(\eta * (\eta - 1) * (\xi - 1) * (\xi + 1)\big) \\ -\big(\xi * (\eta - 1) * (\eta + 1) * (\xi + 1)\big) \\ -\big(\eta * (\eta + 1) * (\xi - 1) * (\xi + 1)\big) \\ -\big(\xi * (\eta - 1) * (\eta + 1) * (\xi - 1)\big) \\ (\eta - 1) * (\eta + 1) * (\xi - 1) * (\xi + 1) \end{Bmatrix} \qquad \text{(AI.2.3)}$$

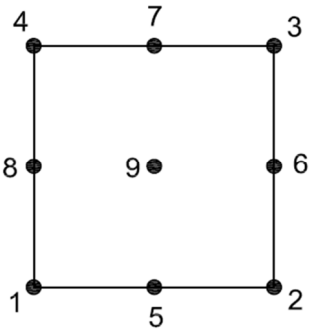


Figura A I0.5: Cuadrilátero de Segundo Orden Con Nueve Nodos

A I.3 ELEMENTOS HEXAEDROS

HEXAEDRO DE PRIMER ORDEN (8 NODOS)

$$\boldsymbol{N}(\boldsymbol{\xi},\boldsymbol{\eta},\boldsymbol{\zeta}) = \begin{Bmatrix} N_1(\xi,\eta,\zeta) \\ N_2(\xi,\eta,\zeta) \\ N_3(\xi,\eta,\zeta) \\ N_4(\xi,\eta,\zeta) \\ N_5(\xi,\eta,\zeta) \\ N_6(\xi,\eta,\zeta) \\ N_7(\xi,\eta,\zeta) \\ N_8(\xi,\eta,\zeta) \end{Bmatrix} = \frac{1}{8} \begin{Bmatrix} (1-\xi)*(1-\eta)*(1-\zeta) \\ (1+\xi)*(1-\eta)*(1-\zeta) \\ (1+\xi)*(1+\eta)*(1-\zeta) \\ (1-\xi)*(1+\eta)*(1-\zeta) \\ (1-\xi)*(1-\eta)*(1+\zeta) \\ (1+\xi)*(1-\eta)*(1+\zeta) \\ (1+\xi)*(1+\eta)*(1+\zeta) \\ (1-\xi)*(1+\eta)*(1+\zeta) \end{Bmatrix} \qquad \text{(AI.3.1)}$$

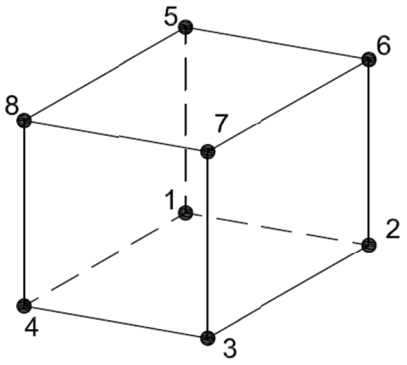


Figura A I.6: Hexaedro de Primer Orden con Ocho Nodos

HEXAEDRO DE SEGUNDO ORDEN INCOMPLETO (20 NODOS)

$$\boldsymbol{N(\xi,\eta,\zeta)} = \begin{Bmatrix} N_1(\xi,\eta,\zeta) \\ N_2(\xi,\eta,\zeta) \\ N_3(\xi,\eta,\zeta) \\ N_4(\xi,\eta,\zeta) \\ N_5(\xi,\eta,\zeta) \\ N_6(\xi,\eta,\zeta) \\ N_7(\xi,\eta,\zeta) \\ N_8(\xi,\eta,\zeta) \\ N_9(\xi,\eta,\zeta) \\ N_{10}(\xi,\eta,\zeta) \\ N_{11}(\xi,\eta,\zeta) \\ N_{12}(\xi,\eta,\zeta) \\ N_{13}(\xi,\eta,\zeta) \\ N_{14}(\xi,\eta,\zeta) \\ N_{15}(\xi,\eta,\zeta) \\ N_{16}(\xi,\eta,\zeta) \\ N_{17}(\xi,\eta,\zeta) \\ N_{18}(\xi,\eta,\zeta) \\ N_{19}(\xi,\eta,\zeta) \\ N_{20}(\xi,\eta,\zeta) \end{Bmatrix} = \frac{1}{8} \begin{Bmatrix} (1-\xi)*(1-\eta)*(1-\zeta)*(-2-\xi-\eta-\zeta) \\ (1+\xi)*(1-\eta)*(1-\zeta)*(-2+\xi-\eta-\zeta) \\ (1+\xi)*(1+\eta)*(1-\zeta)*(-2+\xi+\eta-\zeta) \\ (1-\xi)*(1+\eta)*(1-\zeta)*(-2-\xi+\eta-\zeta) \\ (1-\xi)*(1-\eta)*(1+\zeta)*(-2-\xi-\eta+\zeta) \\ (1+\xi)*(1-\eta)*(1+\zeta)*(-2+\xi-\eta+\zeta) \\ (1+\xi)*(1+\eta)*(1+\zeta)*(-2+\xi+\eta+\zeta) \\ (1-\xi)*(1+\eta)*(1+\zeta)*(-2-\xi+\eta+\zeta) \\ 2*(1-\xi^2)*(1-\eta)*(1-\zeta) \\ 2*(1+\xi)*(1-\eta^2)*(1-\zeta) \\ 2*(1-\xi^2)*(1+\eta)*(1-\zeta) \\ 2*(1-\xi)*(1-\eta^2)*(1-\zeta) \\ 2*(1-\xi^2)*(1-\eta)*(1+\zeta) \\ 2*(1+\xi)*(1-\eta^2)*(1+\zeta) \\ 2*(1-\xi^2)*(1+\eta)*(1+\zeta) \\ 2*(1-\xi)*(1-\eta^2)*(1+\zeta) \\ 2*(1-\xi)*(1-\eta)*(1-\zeta^2) \\ 2*(1+\xi)*(1-\eta)*(1-\zeta^2) \\ 2*(1+\xi)*(1+\eta)*(1-\zeta^2) \\ 2*(1-\xi)*(1+\eta)*(1-\zeta^2) \end{Bmatrix} \quad \text{(AI.3.2)}$$

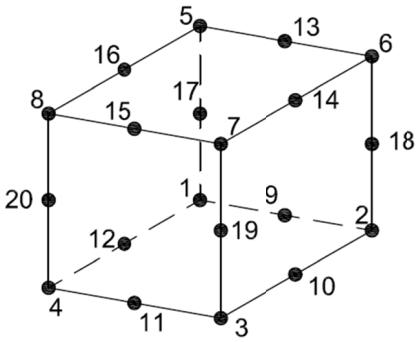


Figura A I.7: Hexaedro de Segundo Orden Incompleto con Veinte Nodos

HEXAEDRO DE SEGUNDO ORDEN COMPLETO (27 NODOS)

$$\boldsymbol{N(\xi,\eta,\zeta)} = \begin{Bmatrix} N_1(\xi,\eta,\zeta) \\ N_2(\xi,\eta,\zeta) \\ N_3(\xi,\eta,\zeta) \\ N_4(\xi,\eta,\zeta) \\ N_5(\xi,\eta,\zeta) \\ N_6(\xi,\eta,\zeta) \\ N_7(\xi,\eta,\zeta) \\ N_8(\xi,\eta,\zeta) \\ N_9(\xi,\eta,\zeta) \\ N_{10}(\xi,\eta,\zeta) \\ N_{11}(\xi,\eta,\zeta) \\ N_{12}(\xi,\eta,\zeta) \\ N_{13}(\xi,\eta,\zeta) \\ N_{14}(\xi,\eta,\zeta) \\ N_{15}(\xi,\eta,\zeta) \\ N_{16}(\xi,\eta,\zeta) \\ N_{17}(\xi,\eta,\zeta) \\ N_{18}(\xi,\eta,\zeta) \\ N_{19}(\xi,\eta,\zeta) \\ N_{20}(\xi,\eta,\zeta) \\ N_{21}(\xi,\eta,\zeta) \\ N_{22}(\xi,\eta,\zeta) \\ N_{23}(\xi,\eta,\zeta) \\ N_{24}(\xi,\eta,\zeta) \\ N_{25}(\xi,\eta,\zeta) \\ N_{26}(\xi,\eta,\zeta) \\ N_{27}(\xi,\eta,\zeta) \end{Bmatrix} = \frac{1}{4} \begin{Bmatrix} \xi*\eta*\zeta*(\xi-1)*(\eta-1)*(\zeta-1)*{}^{1}/_{2} \\ \xi*\eta*\zeta*(\xi+1)*(\eta-1)*(\zeta-1)*{}^{1}/_{2} \\ \xi*\eta*\zeta*(\xi+1)*(\eta+1)*(\zeta-1)*{}^{1}/_{2} \\ \xi*\eta*\zeta*(\xi-1)*(\eta+1)*(\zeta-1)*{}^{1}/_{2} \\ \xi*\eta*\zeta*(\xi-1)*(\eta-1)*(\zeta+1)*{}^{1}/_{2} \\ \xi*\eta*\zeta*(\xi+1)*(\eta-1)*(\zeta+1)*{}^{1}/_{2} \\ \xi*\eta*\zeta*(\xi+1)*(\eta+1)*(\zeta+1)*{}^{1}/_{2} \\ \xi*\eta*\zeta*(\xi-1)*(\eta+1)*(\zeta+1)*{}^{1}/_{2} \\ \eta*\zeta*(1-\xi^2)*(\eta-1)*(\zeta-1) \\ \xi*\zeta*(\xi+1)*(1-\eta^2)*(\zeta-1) \\ \eta*\zeta*(1-\xi^2)*(\eta+1)*(\zeta-1) \\ \xi*\zeta*(\xi-1)*(1-\eta^2)*(\zeta-1) \\ \eta*\zeta*(1-\xi^2)*(\eta-1)*(\zeta+1) \\ \xi*\zeta*(\xi+1)*(1-\eta^2)*(\zeta+1) \\ \eta*\zeta*(1-\xi^2)*(\eta+1)*(\zeta+1) \\ \xi*\zeta*(\xi-1)*(1-\eta^2)*(\zeta+1) \\ \xi*\eta*(\xi-1)*(\eta-1)*(1-\zeta^2) \\ \xi*\eta*(\xi+1)*(\eta-1)*(1-\zeta^2) \\ \xi*\eta*(\xi+1)*(\eta+1)*(1-\zeta^2) \\ \xi*\eta*(\xi-1)*(\eta+1)*(1-\zeta^2) \\ 2*\zeta*(1-\xi^2)*(1-\eta^2)*(\zeta-1) \\ 2*\zeta*(1-\xi^2)*(1-\eta^2)*(\zeta+1) \\ 2*\eta*(1-\xi^2)*(\eta-1)*(1-\zeta^2) \\ 2*\eta*(1-\xi^2)*(\eta+1)*(1-\zeta^2) \\ 2*\xi*(\xi-1)*(1-\eta^2)*(1-\zeta^2) \\ 2*\xi*(\xi+1)*(1-\eta^2)*(1-\zeta^2) \\ 2*(1-\xi^2)*(1-\eta^2)*(1-\zeta^2) \end{Bmatrix} \quad \text{(AI.3.3)}$$

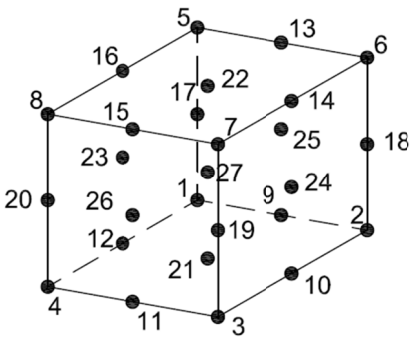
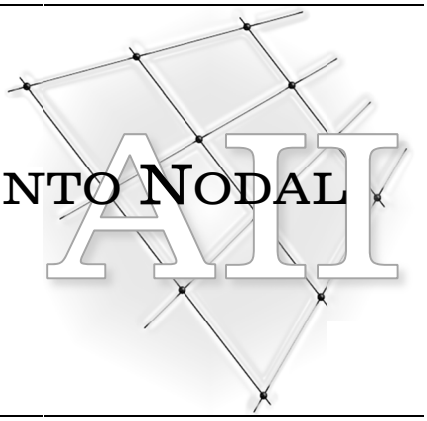


Figura A I.8: Hexaedro de Segundo Orden con Veintisiete Nodos

Anexo II: Ordenamiento Nodal



Todo software de FEM, en cualquier etapa de aplicación del método, posee un sistema de ordenamiento nodal propio que se respeta constante en sus códigos. Es por esto que, como se explicó en la sección IV.2, es necesario entender correctamente el sistema de secuencia con el que se trabaja para evitar la propagación de errores.

En este tercer anexo se incluyen los diagramas completos de algunos elementos junto con sus tablas de compatibilidades. Las cuales corresponden al traslado desde el ordenamiento utilizado por Gmsh al ordenamiento propuesto por T. Hughes, que fue tomado como sistema de cabecera. Se presentan a continuación elementos cuadriláteros y hexaedros, de primer y segundo orden, incluyendo las versiones incompletas.

A II.1 Elementos Cuadriláteros

Diagramas de Ordenamiento

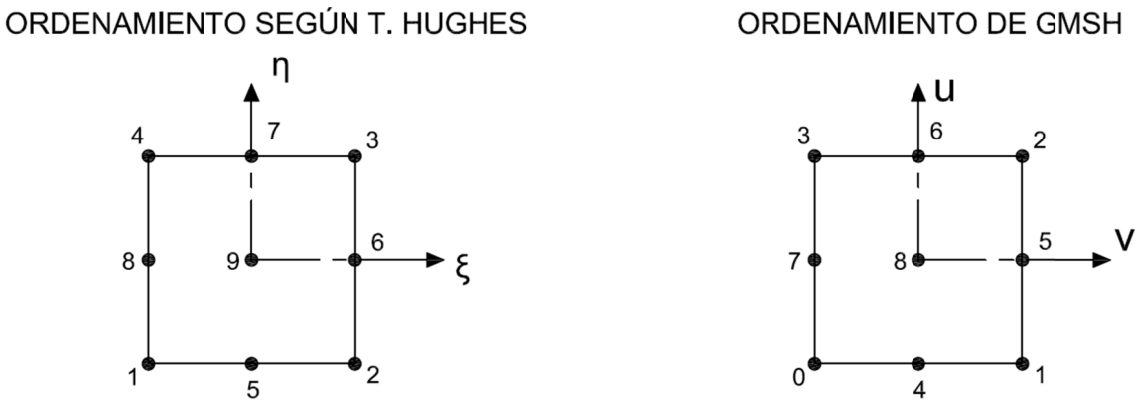


Figura A II.1: Diagrama de Ordenamiento para Elementos Cuadriláteros

TABLA DE COMPATIBILIDAD

ORDENAMIENTOS	GMSH	HUGHES
Cuadrilátero de Segundo Orden Completo (9 nodos) / Cuadrilátero de Segundo Orden Incompleto (8 nodos) / Cuadrilátero de Primer Orden (4 nodos)	0	1
	1	2
	2	3
	3	4
	4	5
	5	6
	6	7
	7	8
	8	9

Figura A II.2: Tabla de Compatibilidad para Elementos Cuadriláteros

A II.2 ELEMENTOS HEXAEDROS

DIAGRAMAS DE ORDENAMIENTO

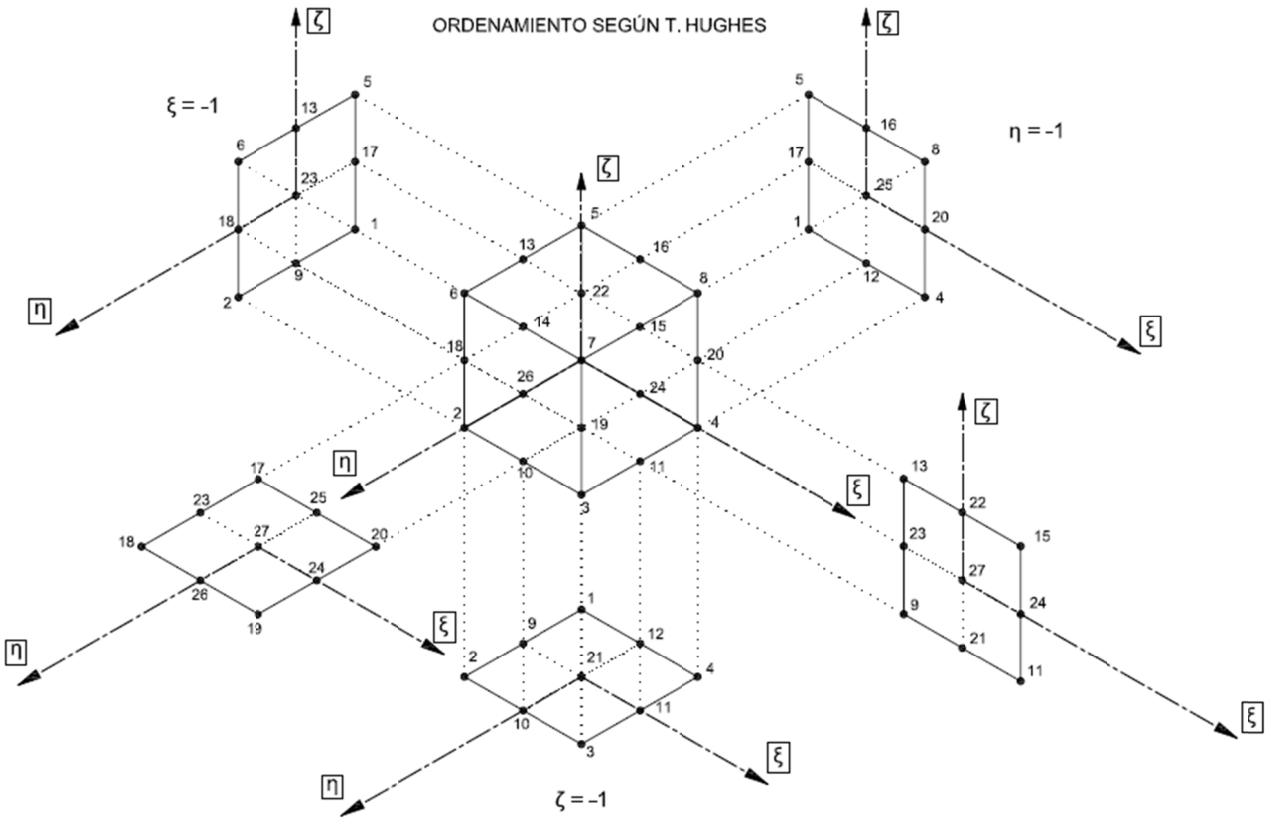


Figura A II.3: Diagrama de Ordenamiento para Elementos Hexaedros según T. Hughes

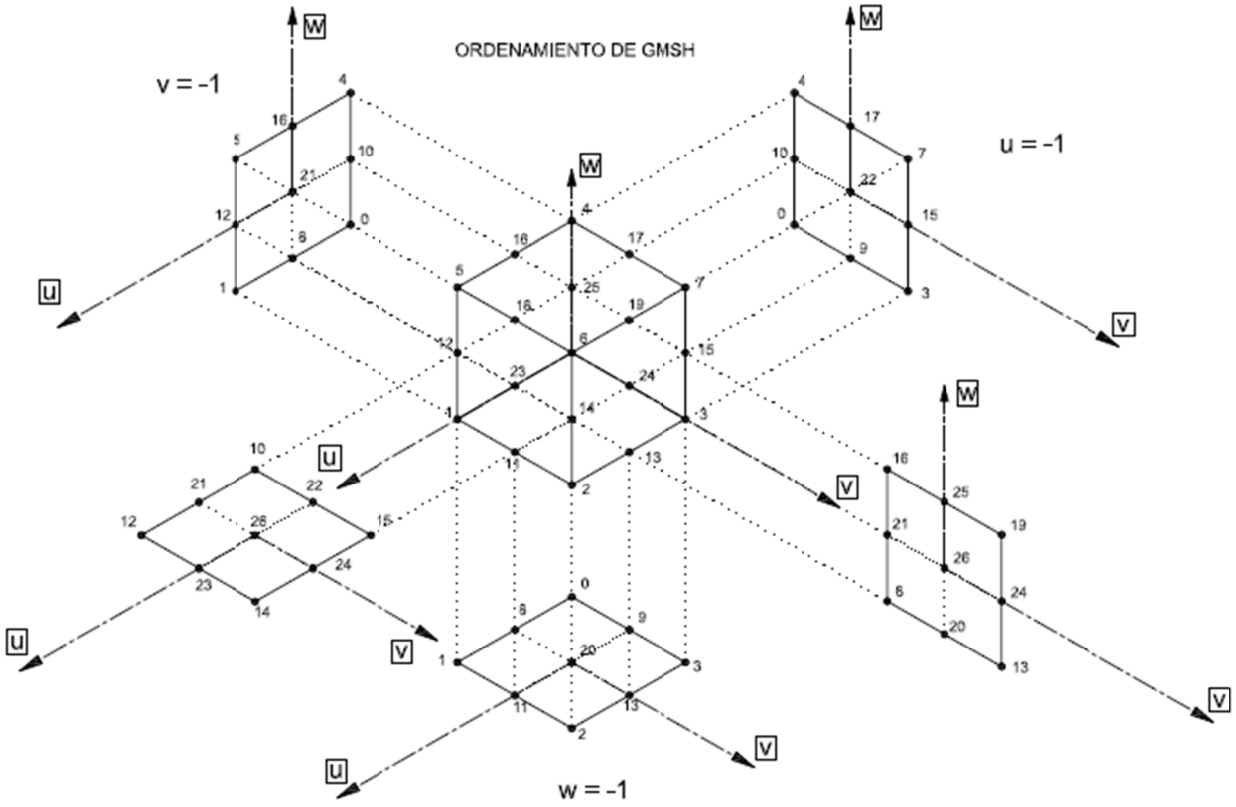


Figura A II.4: Diagrama de Ordenamiento para Elementos Hexaedros según GMSH

TABLA DE COMPATIBILIDAD

ORDENAMIENTOS	GMSH	HUGHES
Hexaedro de Primer Orden (8 nodos)	0	1
	1	2
	2	3
	3	4
	4	5
	5	6
	6	7
	7	8
	8	9
Hexaedro de Segundo Orden Incompleto (20 nodos)	9	12
	10	17
	11	10
	12	18
	13	11
	14	19
	15	20
	16	13
	17	16
	18	14
	19	15
Hexaedro de Segundo Orden Completo (27 nodos)	20	21
	21	23
	22	25
	23	26
	24	24
	25	22
	26	27

Figura A II.5: Tabla de Compatibilidad para Elementos Hexaedros

Anexo III: Lista de Códigos de Programa

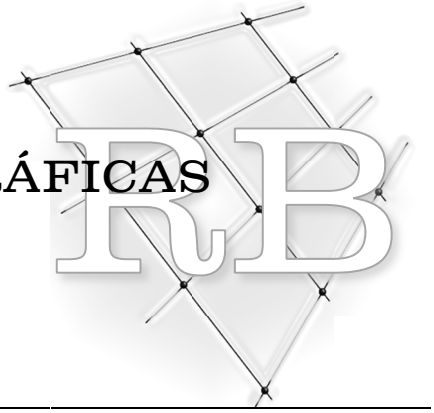
Todos los algoritmos implementados a lo largo del trabajo se encuentran codificados en lenguaje de MATLAB. A continuación se presenta una lista que indica el nombre del archivo (".m") que contiene el código, la sección del presente trabajo en la que se trata dicho código y una breve descripción. Esta lista se condice con los archivos que integran el Anexo Digital, en la carpeta "Códigos".

Nombre del Archivo	Sección	Descripción
"NodesArray.m"	IV.3.1	Genera el Arreglo de Nodos y Coordenadas a partir del archivo de una malla.
"IENArray.m"	IV.3.2	Genera el Arreglo IEN a partir del archivo de una malla.
"IENArray_Reduction.m"	IV.3.2	Incorpora la Reducción de Elementos al Arreglo IEN.
"IENReduction.m"	IV.3.2 VI.2.3	Elimina los elementos nulos del Arreglo IEN.
"IndexTrans.m"	IV.2 AII	Traduce los índices del ordenamiento nodal de GMSH al propuesto por T. Hughes.
"PlaneShapeFunctionsLib.m"	IV.4.2	Biblioteca de Funciones de Forma para elementos planos.
"PlaneShapeFunctionsLib_Reduction.m"	IV.4.2 VI.2.3	Incorpora la Reducción de Elementos a la Biblioteca de Funciones de Forma.
"SpaceShapeFunctionsLib.m"	IV.4.2	Biblioteca de Funciones de Forma para elementos espaciales.
"SpaceShapeFunctionsLib_Reduction.m"	IV.4.2 VI.2.3	Incorpora la Reducción de Elementos a la Biblioteca de Funciones de Forma.
"DifN.m"	IV.4.3	Genera la Biblioteca de Funciones de Forma Derivadas.

"FullLocation_3D.m"	V.2 VI.2	Programa de Vinculación con el Algoritmo de Lista y el Algoritmo de Newton-Raphson.
"FullLocation_SingleRandom_3D.m"	V.3.1 VI.3	Ídem anterior, utilizando el Algoritmo de Lista Aleatoria Simple.
"FullLocation_MultiRandom_3D.m"	V.3.2 VI.2	Ídem anterior, utilizando el Algoritmo de Lista Aleatoria Múltiple.
"FullLocation_MassCentre_3D.m"	V.4 VI.2	Programa de Vinculación con el Algoritmo de Centros de Masa y el Algoritmo de Newton-Raphson.
"FullLocation_MassCentre_3D_Coords.m"	V.4 VI.3	Programa de Vinculación con el Algoritmo de Centros de Masa y el Algoritmo de Coordenadas de Área.
"FullLocation_ESUP_3D.m"	V.5 VI.2	Programa de Vinculación con el Algoritmo ESUP y el Algoritmo de Newton-Raphson.
"FullLocation_ESUP_3D_Coords.m"	V.5 VI.3	Programa de Vinculación con el Algoritmo ESUP y el Algoritmo de Coordenadas de Área.
"FullLocation_ESUP_MassCentre_3D.m"	V.6 VI.2	Programa de Vinculación con el Algoritmo ESUP Ordenado por Centros de Masa y el Algoritmo de Newton-Raphson.
"FullLocation_ESUP_MassCentre_3D_Coords.m"	V.6 VI.3	Programa de Vinculación con el Algoritmo ESUP Ordenado por Centros de Masa y el Algoritmo de Coordenadas de Área.

Tabla A III.1: Listado de Códigos

Referencias Bibliográficas



[1] Thomas J. R. Hughes, "*The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*," Prentice-Hall Inc., New Jersey, USA, 1987.

[2] Romesh C. Batra, "*An Introduction to the Finite Element Method*," Department of Engineering Science & Mechanics, Virginia Polytechnic Institute & State University, Blacksburg, USA, 2006.

[3] Pascal Getreuer, "Writing Fast MATLAB® Code," 2008.

[4] Reinald Löhner, "Applied CFD Techniques: An Introduction Based on Finite Element Methods," John Wiley & Sons, New Jersey, USA, 2008.

[5] Stacey A. Combes, "*Wing Flexibility and Design for Animal Flight*," PhD Thesis, University Of Washington, USA, 2002.

[6] C. Geuzaine, J. F. Remacle, "*Gmsh Reference Manual*," Free Software Foundation Inc., USA, 2013.

[7] Amar Khennane, "*Introduction to Finite Element Analysis Using MATLAB and Abaqus*", CRC Press Taylor & Francis Group LLC, USA, 2013.

[8] Pablo Amster, "*Topological Methods in the Study of Boundary Value Problems*", Springer Science+Business Media, New York, USA, 2014.

[9] Eugenio Oñate, "*Structural analysis with the Finite Element Method – Linear Statics Vol. 1*", Artes Gráficas Torres S. L., CIMNE, Barcelona, España, 2009.

[10] A.Kaveh, "*Computational Structural Analysis and Finite Element Methods*", Springer International Publishing, Switzerland, 2014.