UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

"APLICACIÓN DE LA TRIANGULACIÓN DE DELAUNAY AL MÉTODO DE LOS ELEMENTOS FINITOS"

T E S I S

PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A

JUAN URIEL QUEZADA CURIEL

ASESOR: MAT. RAMÍREZ FLORES LUÍS

MÉXICO 2012

FES Aragón

ÍNDICE

INTRODUCCIÓN

La necesidad de medir con una mayor aproximación los esfuerzos y deformaciones en el mundo físico; tales como, choques automovilísticos, estudios anatómicos, Fisiología del Deporte, Biología Morfológica, Ingeniería Civil, Sismología, Aeronáutica, Bioingeniería, Sismología, Mecánica Industrial, Graficación por Computadora, videojuegos y otros; hacen de una gran utilidad el Método de los Elementos Finitos (MEF), ya que provee un comportamiento previo; así como de simulaciones en la conducta de materiales o energía.

El MEF es un método numérico que sustituye con gran eficiencia a otros métodos analíticos; que en la práctica, son extremadamente complejos. El MEF, básicamente consiste en obtener información cinemática de un objeto con comportamiento analógico mediante técnicas analíticas

Éste necesita como primer paso discretizar el elemento a analizar; bajo este dominio; cada elemento con cierta libertad se le denomina: Nodo. En el análisis estructural de dos dimensiones sería un mayado, en tres hablaríamos de un "Esqueleto alámbrico" formando tetraedros, de tal manera que el centro de ésta tesis será este tema.

La triangulación de Delaunay es el procedimiento más óptimo para generar estas estructuras mencionadas anteriormente; pues la estabilidad de sus triángulos (en 2D por ejemplo) la hace

superar a cualquier otra forma de triangulación; ya que, tienen sus triángulos a ser lo más equiláteros posibles.

El problema radica en que la generación de la triangulación de Delaunay se basa en procedimientos combinatorios extensos y búsquedas ciegas; y a pesar de la existencia de métodos que mejoran esto, tienden a ser precisos en sus triangulaciones haciendo el proceso desgastante.

La propuesta optada en esta tesis es una "Semitriangulación" bastante aproximada a la generada con un método preciso; es decir, un algoritmo que se aproxime con *n* número de iteraciones a la triangulación de Delaunay perfeccionándose en cada iteración.

Un aporte es la Triangulación de Delanuay mediante Esferas de Cobertura en *n* dimensiones que en nuestro caso será de tres; puesto que, tomaremos un modelo estructural (un avión Harrier) y se deformará con el MEF. Esto se ha logrado incorporando la función para hacer hiperesferas mediante Mínimos Cuadrados y Álgebra Geométrica.

Para ello se tiene que comenzar por proporcionar un ambiente básico de la triangulación de Delaunay y su dualidad con el Diagrama de Voronoi; así como también los conceptos de la geometría computacional; por otra parte se da cuenta la importancia de las aplicaciones de las cercanías-proximidades y sus principales enfoques de estudio.

PLANTEAMIENTO DEL PROBLEMA

¿Se puede optimizar la discretización de una estructura en 3D con un resultado aproximado a la triangulación de Delaunay para el Método de los Elementos Finitos?

OBJETIVO GENERAL

Proponer un algoritmo eficiente y práctico con el que se pueda discretizar un objeto en 3D basado en la triangulación de Delaunay para ser aplicado al Método de los Elementos Finitos.

OBJETIVOS ESPECÍFICOS

- Realizar un software de triangulación de Delaunay en 2D.
- Realizar un algoritmo para generar esferas de manera eficiente.
- Generación de un software que aplique las deformaciones a un modelo 3D.
- Proponer un algoritmo que involucre la triangularización de manera más eficiente.

ANTECEDENTES

Introduciremos algunos conceptos de primitivas geométricas que necesitaremos.

Existen diversos sistemas geométricos que pueden expresar algoritmos geométricos: Geometría Afín, Geometría Euclidiana y Geometría Proyectiva por ejemplo. Empezaremos por definir algunos conceptos básicos de la Geometría; la llamada: *Geometría Afín.* Por último menciono un procedimiento sencillo; pero útil, para generar hiperesferas y así poder trabajar las esferashiperesferas de cobertura.

CAPÍTULO UNO

GEOMETRÍA COMPUTACIONAL

1.1 Aplicaciones

Podría decirse que esta disciplina es el nexo de unión entre la geometría y el mundo de la computación. Existen muchos campos de la ciencia de la computación que resuelven problemas de naturaleza geométrica. Estas incluyen Computación Gráfica, Visión por Computadora y Procesamiento de Imágenes, Robótica, Dinámica de Fluidos computacionales, Sistemas de Información por gráficos, sólo por nombrar algunos. La Geometría Computacional resuelve diversos problemas desde una perspectiva geométrica (Mount, 2002). Esta disciplina es relativamente joven de ahí su desconocimiento y escasa difusión.

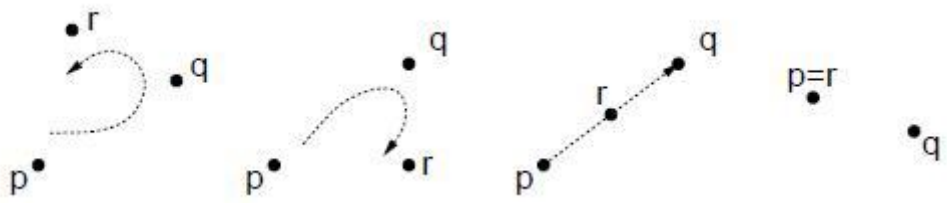
Gran parte de la historia de la Geometría Computacional se ha centrado en problemas espaciales para 2 dimensiones y poco en 3; otro factor es que presupone trabajar con un número constante y reducido de dichas dimensiones (10 o menos), pues la complejidad computacional aumenta exponencialmente (Burkardt, 2002).

La investigación en dicha área ha fomentado el cálculo correcto y robusto de primitivas geométricas basado en fundamentos matemáticos sólidos.

1.2 Orientaciones

Orientación: Con el fin de tomar decisiones discretas, necesitamos una operación relacional que simule las operaciones (<, >, =) de los reales en aritmética para trabajar con puntos. Existe una relación entre las (d + 1) tuplas de puntos en una *d* espacial y que toma el nombre de orientación de la relación binaria en 1-espacio.

Dada una tercia ordenada de puntos (p,q,r) en el plano, decimos que estas tienen una *orientación positiva* si definen una orientación triangular en sentido contrario al reloj, *orientación negativa* viceversa y *orientación cero* si son colineales o iguales. Notemos que la orientación depende del orden en que son dados los puntos y se altera con la aplicación de una transformación afín (Trians, 2003).



Positivo negativo cero

Formalmente, la orientación es denotada con el signo del determinante de los puntos en coordenadas homogéneas, esto es anteponiendo un 1 a cada coordenada:

$$\text{Orient}(p,q,r) = \det \begin{pmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{pmatrix}.$$

Esto puede ser generalizado a cualquier orden $(d+1)$ *- tuplas de puntos en d espacio.*

Áreas y ángulos: La orientación del determinante junto con la norma euclidiana, puede ser usado para computar los ángulos en el plano. El área del triángulo puede ser determinada si dividimos esta cantidad entre 2. En una dimensión *d*, y *d* + 1 puntos. En general el área puede ser delimitada por la división de éste determinante por (*d*!). Una vez sabiendo el área de un triángulo podemos computar el área de un polígono expresándolo como una suma de áreas.

El seno de un ángulo $\theta = |p-q||r-q|$ *Orient(q,p,r)*

1.3 Cerco convexo

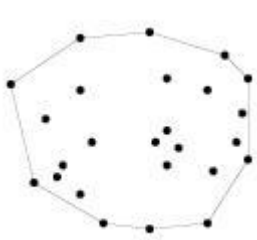
Un gran número de razones son de importancia para obtener un cerco convexo de un conjunto de puntos. Esto puede ser usado para aproximar formas más complejas. Por ejemplo, el cerco convexo de un polígono en el plano o poliedro en 3-D es el cerco convexo de sus vértices.

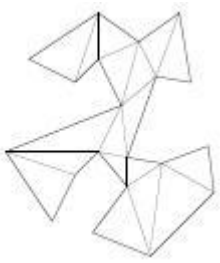
Un conjunto *S* es convexo si dado cualquier punto *p*, *q* ∈ *S*, en cualquier combinación convexa de *p* y *q* esta en *S;* o equivalentemente, la línea del segmento $\overline{pq}$ pertenece al conjunto *S*.

Un cerco convexo de S se define como la intersección de todos los conjuntos convexos que contiene *S*, o más intuitivamente el conjunto convexo más pequeño que contiene a *S*.

Una definición equivalente de cerco convexo es el conjunto de puntos que pueden ser expresados como combinaciones convexas de los puntos en *S*. Recordar que una combinación convexa de tres o más puntos es una combinación afín de puntos en que los coeficientes suman 1 y todos éstos están en un intervalo de [0,1] (P. Cignonini, 1997).

La convexidad es una propiedad importante de la geometría. Se produce esta; si para dos puntos cual quiera en un conjunto, la línea de segmento que los une se encuentra dentro del mismo conjunto. Uno de los primeros problemas de los campos de la computación geométrica, es determinar la forma convexa más pequeña, llamada *cerco convexo* que envuelve a un conjunto de puntos.





Cerco Convexo Triangulación de polígonos

El problema de generar un cerco convexo es: Dado un conjunto N de puntos (en el plano), generar un polígono convexo cerrado, enumerando sus vértices con una orientación positiva (que puede ser negativa también). Esto podemos lograrlo tomando una pareja cualquiera y con cada punto de los restantes aplicarle *Orient()*; es decir, saber si éste último punto se encuentra a la derecha o a la izquierda de la línea conformada de los dos puntos tomados. Este ciclo de tomar todos los pares junto con todos los puntos toman un costo de *O(n3)*.

1.3.1 Algoritmo de Graham

En la primera fase de este algoritmo se ordenan los puntos de acuerdo a su ángulo. Tomamos el punto con la ordenada más baja y se etiqueta como punto de referencia angular; también podemos hacerlo tomando un punto interior al cierre o cerco convexo, esto es escogiendo tres puntos no alineados y generando su baricentro.

En seguida, se toman tercias de puntos de la lista ordenada generada para aplicarle la función *Orient()* o área signada, con el fin de descalificar a los puntos que no son vértices del cerco convexo. El orden computacional de este algoritmo es de $O(n \log n)$ (Peter Su, 1996).

La intersección es uno de los problemas más elementales es la determinación de dos conjuntos

de objetos que se cruzan unos con otro a menudo se reduce a determinar cuales son los pares de entidades primitivas que lo hacen (ejemplo segmento de líneas).

CAPÍTULO DOS

Diagramas de Voronoi y Triángulos de Delaunay.

Los diagramas antes mencionados son estructuras que su objetivo fundamental es la de almacenar información referente a la proximidad entre puntos (en *n* dimensiones); de tal forma que, dado un punto podemos encontrar el punto más cercano a éste en una nube de puntos.

2.1 Diagramas de Voronoi

Sea P = $\{p_1,p_2,\ldots,p_1\}$ un conjunto de puntos en el plano (o en cualquier dimensión espacial), que llamaremos *sitios*. Definimos $\mathcal{V}(p_i)$, una célula de Voronoi para p_i, establecemos un conjunto de puntos q en el plano que están más cerca (utilizando la distancia euclidiana) de p_i, que de cualquier otro sitio . Esto es, la célula de Voronoi para p_i y está definido como:

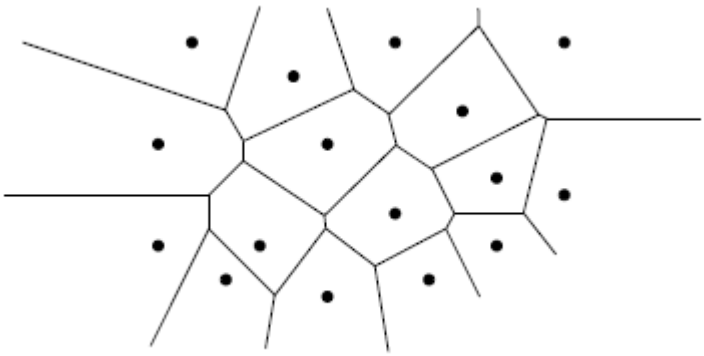
$$\mathcal{V}(p_i) = \{q \mid |p_iq| < |p_jq|, \forall j \neq i\}$$

Otra forma de definir $\mathcal{V}(p_i)$ es en términos de intersección de semiplanos. Dado dos sitios p_i y p_j, el conjunto de puntos que están estrictamente cerca de p_i más que a p_j es justamente un semiplano *abierto* cuya línea que lo encierra es un bisector perpendicular entre p_i y p_j. Denotemos este semiplano como $h(p_i,p_j)$. Fácilmente se puede

observar que el punto q se encuentra en $V(p_i)$; si y solamente si, q se encuentra dentro de la intersección de $h(p_i, p_j)$ para toda $j \neq i$ (Zimmer, 2005). En otras palabras:

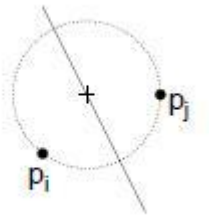
$V(p_i) = \cap_{j \neq i} h(p_i, p_j)$.

La intersección de semiplanos es un polígono convexo (posiblemente no acotado). Así que, un diagrama de Voronoi consiste en una colección de segmentos de líneas, que pueden ser no acotadas, ya sea en uno o en ambos lados.



2.1.1 Propiedades del Diagrama de Voronoi

Aristas de Voronoi: Cada arista en un vértice de Diagrama de Voronoi es equidistante desde sus dos vecinos más cercanos p_i y p_j. Así que, podemos trazar un círculo para cada punto que forman las aristas.



Grado: Si existen *n* sitios concéntricos en un mismo vértice de Voronoi; se dice que, es de el diagrama de Voronoi es de orden *n*

Cerco convexo: Un polígono está perfectamente delimitado por sus aristas si el sitio que contiene cae sobre un cerco convexo.

Tamaño: Si *n* denota el número de sitios, entonces podemos decir que el Diagrama de Voronoi es un grafo plano con exactamente *n* lados. Esto sigue la regla de Euler que dice los vértices de Voronoi son a lo sumo $2n-5$ y el número de vértices es a lo sumo $3n-6$.

2.1.2 Aplicaciones

Búsqueda del vecino más cercano: Una de las más importantes estructuras de datos en geometría computacional es la solución de búsquedas de vecinos cercanos. Dado un conjunto de puntos *P*, y dado un punto a buscar *q*, determinar los puntos más cercanos de *P* a *q*. Esto puede ser resuelto computando primero del diagrama de Voronoi y luego localizando la célula del diagrama que contenga *q*.

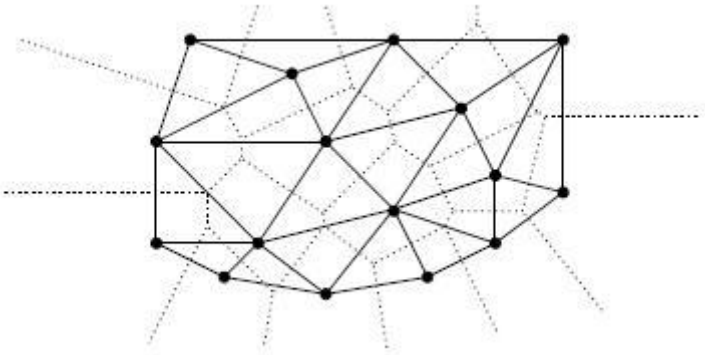
Facilidad de ubicación: Si deseamos establecer un negocio en un determinado lugar geográfico, ¿Cómo sabremos exactamente cual es la mejor opción de hacerlo? Podríamos colocarlo lo más lejano a los demás existentes pero perderíamos clientela. Los diagramas de Voronoi nos ayudarían a analizar este problema.

Vecinos e interpolación: Dado un conjunto de puntos con alturas medibles sobre algún terreno geométrico. Cada punto tiene (*x, y*) coordenadas y un valor de altura. Podríamos interpolar el valor de altura de un punto de búsqueda que no tengamos su valor. Una de las formas de hacer esto es la llamada *interpolación de los vecinos naturales*, esta basada en el cómputo de los vecinos Voronoi.

2.2 Triángulos de Delaunay

2.2.1 Conceptos básicos

Podemos decir que el grafo dual correspondiente al diagrama de Voronoi es el triángulo de Delaunay (DT). Los vértices de este grafo dual son en sí mismos los sitios. Así que, si el grado del diagrama de Voronoi es de tres el resultado será una triangulación formando así los Triángulos de Delaunay.



Triángulos de Delaunay de un conjunto de puntos (líneas sólidas) y el diagrama de Voronoi (líneas punteadas) (Shewchuk, 1999).

Los triángulos de Delaunay tienen interesantes propiedades que son consecuencia del diagrama de Voronoi.

Cerco convexo: La acotación del perímetro exterior de los triángulos de Delaunay es el acotamiento del cerco convexo del conjunto de puntos.

Circunferencia circunscrita: La circunferencia circunscrita de cualquier triángulo que compone la triangulación de Delaunay esta vacío (no contiene sitios *P*).

Círculo vacío: Dos sitios p_i y p_j son conectados mediante una arista en la triangulación de Delaunay, si y sólo si hay un círculo vacio pasando a través de p_i y p_j.

Par más cercano: Los pares más cercanos de los sitios en *P* son vecinos en la triangulación de Delaunay

Es común asumir que los sitios se encuentran en una posición general; es decir que, a cada si estamos en dos dimensiones podremos tener tres sitios; en otro caso se habla de grafos de Delaunay.

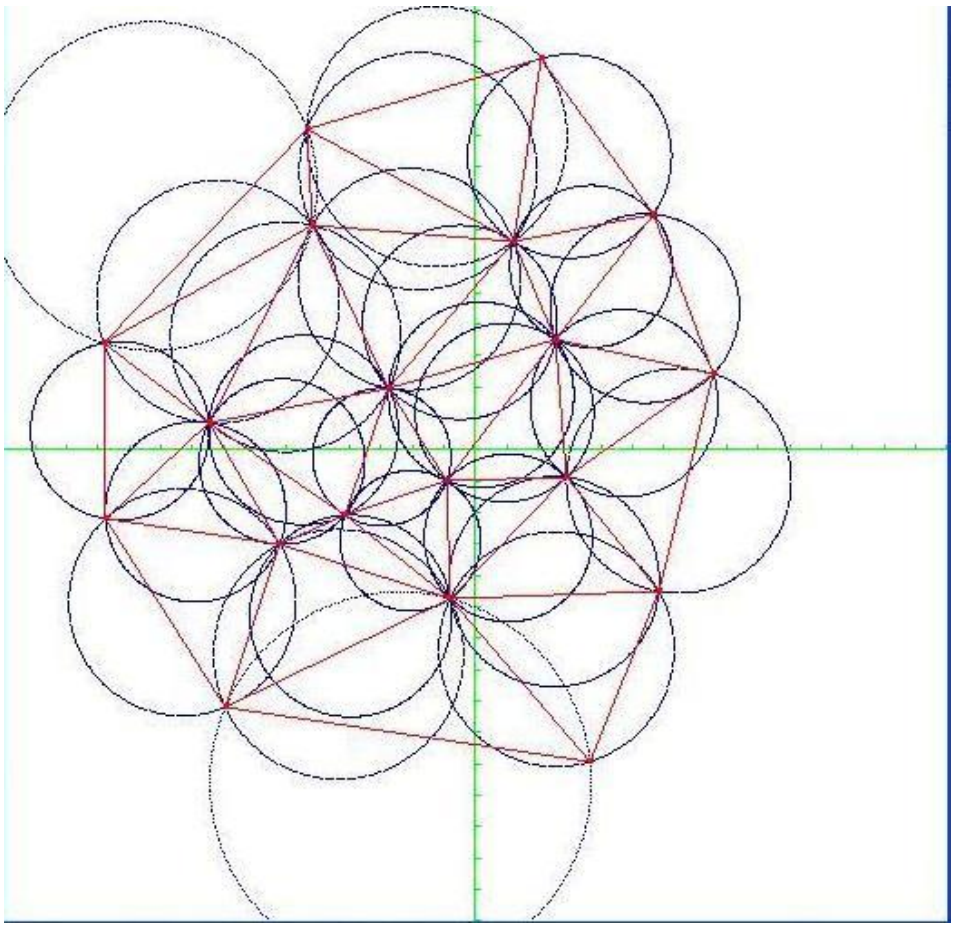
Dado un conjunto de puntos P con *n* sitios donde hay *h* sitios sobre la cerco convexo; por la fórmula de Euler, (Marsahll Bern, 98) la triangulación de Delaunay tiene $2n-2-h$ triángulos, y $3n-3-h$ lados; estas fórmulas sólo trabajan en 2 dimensiones, en cuatro dimensiones oscila entre $O(n)$ hasta $O(n^2)$. En dimensiones mayores, el número de simplex (generalización de un triángulo en n dimensiones) oscila entre $O(n^{\frac{d}{2}})$ (Burkardt, 2002).

2.2.2 Construcción incremental.

La construcción incremental se realiza cuando cada punto es añadido sin la necesidad de generar las combinaciones de todos los puntos.

La siguiente figura muestra la triangulación de Delaunay en 2D hecha en el software realizado por el autor de esta tesis.

Cada punto es ubicado con el cursor y el programa genera la triangulación, cabe señalar aquí que este programa fue realizado con una búsqueda ciega; es decir, el peor algoritmos $O(2)$.



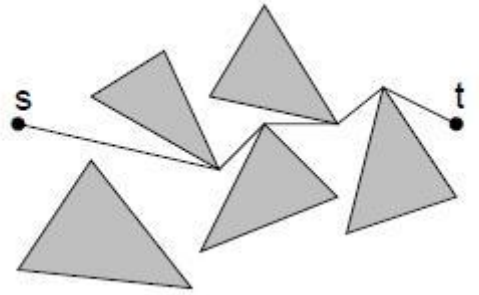
CAPÍTULO TRES.

PROXIMIDAD, LOCALIDAD Y ESTRUCTURAS ESPACIALES DE DATOS.

Uno de los dos puntos que motivaron esta tesis fue encontrar un algoritmo que genere la triangulación de Delaunay evitando la complejidad de los algoritmos existentes y manejando una simplicidad; este será tema de un próximo capítulo, donde se pondrá el algoritmo en el que se basó esta tesis.

3.1 Un ejemplo clásico de búsqueda

Un problema común es el de encontrar la ruta más corta. Dado un conjunto de objetos poligonales en el plano, encontrar la ruta más corta; eludiendo los polígonos durante el trayecto y empezando desde un punto propuesto hacia otro como el objetivo (Mount, 2002).



Ruta más corta desde el punto S al punto t

3.2 Localidad

El término de "localidad" se refiere a la posición relativa de un punto a una subdivisión geométrica u objetos geométricos discontinuos. El ejemplo más conocido es el problema de la localización de un punto, en que una subdivisión espacial dentro de áreas discontinuas contienen el punto buscado.

El también llamado "El problema de la ruta más corta" es establecido por polígonos con la función de ser obstáculos en un plano. Se trata de evitar a estos polígonos y encontrar a un punto también dado con anterioridad; marcando así, una ruta que pueda ser medida y comparada con otras como la más corta. Aunque es posible encontrar una solución fuera del dominio de los algoritmos geométricos como el algoritmo Dijkstra, parece que al resolver el problema en su dominio geométrico sería posible proponer soluciones más eficientes.

La medida de la calidad de un algoritmo en la geometría computacional ha sido tradicionalmente

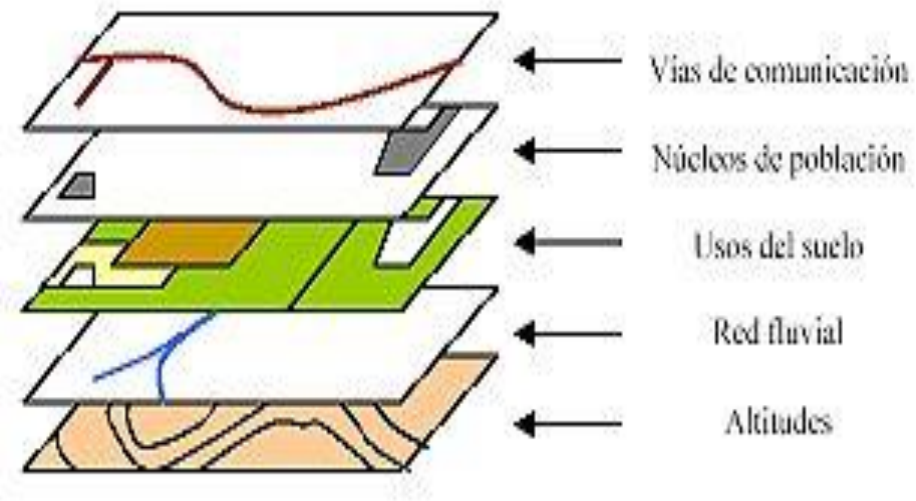
representada la asintótica que refleja el "peor de sus casos". Por lo tanto, un algoritmo que se ejecuta en O(n) de tiempo es mejor que uno se ejecuta en O (n log n) de tiempo; y este a su vez, es mejor que uno que se ejecuta en O(n2) de tiempo. (Este problema en particular puede resolverse en $O(n^2 \log n)$ por un algoritmo bastante sencillo y en O(n log n) por un algoritmos relativamente complejo. En algunos casos el tiempo medio de ejecución es el que se considera.

3.3 Datos Espaciales

Los datos espaciales consisten en objetos espaciales compuestos por puntos, líneas regiones, rectángulos, superficies, volúmenes e incluso datos de alta dimensión algunos ejemplos son: Ciudades, ríos, carreteras, estados, campos agrícolas, montañas y partes de sistemas CAD, etc. Los ejemplos de las propiedades espaciales incluyen la medida de un río dado o los limites de una ciudad, etc. A veces es también deseable conocer los atributos o información no espacial tales como temperaturas, densidades, etc. Las bases de datos espaciales facilitan la carga computacional y la eficiencia del procesamiento computacional (Buchmann A. Günther, 1990).

Con respecto a las bases de datos espaciales mencionadas en el párrafo que antecede, estas son generadas y almacenadas en las estructuras de datos espaciales; ya que al almacenarse dentro de los datos espaciales, se puede simplificar su información, también puede obtenerse con mayor

eficiencia una búsqueda de un objeto espacial. Estas bases de datos que conforman las estructuras espaciales pueden trabajar con diversos algoritmos especializados en búsqueda-inserción.



INFORMACIÓN DEL MUNDO REAL REALIZADO EN CAPAS TEMATICAS

Un dato espacial es una variable asociada a una localización del espacio. Normalmente se utilizan datos vectoriales, los cuales pueden ser expresados mediante tres tipos de objetos espaciales: Puntos, líneas y polígonos.

De esta forma la información sobre puntos, líneas y polígonos se almacena como una colección de coordenadas (x, y). Las características lineales, pueden almacenarse como un conjunto de puntos de coordenadas (x, y). Las características poligonales, pueden almacenarse como un circuito cerrado de coordenadas. La otra forma de expresar

datos espaciales es mediante rasterización, la cual, a través de una malla que permite asociar datos a una imagen; es decir, se pueden relacionar paquetes de información a los píxeles de una imagen digitalizada.

Los datos espaciales además se caracterizan por su naturaleza georreferenciada y multidireccional. La primera se refiere que la posición relativa o absoluta de cualquier elemento sobre el espacio contiene información valiosa, pues la localización debe considerarse explícitamente en cualquier análisis. Por multidireccional se entiende a que existen relaciones complejas no lineales, es decir que un elemento cualquiera se relaciona con su vecino y además con regiones lejanas, por lo que la relación entre todos los elementos no es unidireccional. Es decir, todos los elementos se relacionan entre sí, pero existe una relación más profunda entre los elementos más cercanos.

3.4 Métodos de Acceso Espacial

Para evitar la revisión exhaustiva de los datos en una base de datos, se crean índices que reducen el número de elementos a visitar en la base de datos en un procesamiento de consulta. La clásica indexación por B-tree no es aplicable en el caso espacial donde no existe un orden único de los valores de claves. Es por este motivo que existen tres categorías de métodos de acceso

espacial, las PAM (Point Access Method), R-Tree, las SAM (Spatial Access Method), los cuales se utilizan de acuerdo al tipo de dato en el que está la base de datos espacial ya sea raster o vectorial. Aunque se han creado los benchmarks que comparan diferentes métodos, los resultados no son concluyentes, pero se recomienda utilizar cualquiera de ellos. Un índice R-tree aproxima cada geometría en un único rectángulo que la acota minimizando los espacios llamado MBR (Minimal Bounding Rectangle) y organiza una colección de objetos espaciales en una jerárquica donde las hojas contienen punteros a los datos y los nodos intermedios contienen el rectángulo mínimo que contiene a sus sub-hojas. Todas las hojas aparecen al mismo nivel. Cada entrada a una hoja es una tupla (R,O), donde R es el MBR y O es el objeto. Cada nodo intermedio es un tupla (R,P), donde R es el MBR que contiene los rectángulos hijos apuntados por P.

3.5 Lenguajes de Consulta Espacial

Las bases de datos espaciales no tienen un conjunto de operadores que sirvan como elementos básicos para la evaluación de consultas ya que estas manejan un volumen extremadamente grande de objetos complejos no ordenados en una dimensión. Es por esto que existen algoritmos complejos para evaluar predicados espaciales. Las consultas son realizadas generalmente en SSQL (Spatial SQL), el cual introduce, mediante extensiones, los distintos conceptos del álgebra ROSE dentro del

lenguaje SQL estándar, es decir, utiliza las cláusulas SELECT-FROM-WHERE para las tres operaciones en el álgebra relacional (proyección algebraica, producto cartesiano y selección) (Liria, 2002). Las tres categorías fundamentales de consultas en un sistema de información espacial son:

En el lenguaje SSQL, el ejemplo del segundo punto se escribiría de la siguiente forma.

SELECT poblacion FROM ciudades WHERE nombre= "Valdivia"

El otro tipo de consultas, para los datos obtenidos mediante rasterización, es llamado PSQL(Pictoral SQL) donde cada objeto espacial se extiende mediante un atributo *loc* (localización) el cual es referenciado en la cláusula SELECT para una salida gráfica y una cláusula específica para tratar relaciones espaciales. También se destaca en los lenguajes de modelado de la información espacial a GML que es una estructura para almacenar y compartir datos geográficos. Es una codificación del modelo geométrico de rasgo simple del OGC (Open Geospatial Consortium simple feature) usando XML. Un rasgo geográfico (geographic feature) es definido por el OGC como "una abstracción del fenómeno del mundo real, si éste está asociado con una posición relativa a la Tierra". Por tanto, es posible hacer una representación del mundo real con un conjunto de rasgos. La especificación de un rasgo viene dada

por sus propiedades, las que pueden pensarse definidas como un triple (nombre, tipo, valor). Si este rasgo es geográfico entonces la propiedad tendrá un valor geométrico. Por tanto, un rasgo simple del OGC es aquel cuya propiedad geométrica está restringida a una geometría simple en la que sus coordenadas estén definidas en dos dimensiones y en el caso de existir una curva, ésta es sujeta a una interpolación lineal.

CÁPITULO CUATRO

INTRODUCCIÓN AL MÉTODO DE ELEMENTOS FINITOS

4.1 Introducción

El método de los elementos finitos (MEF) es un método numérico para la solución de problemas ingenieriles donde se necesita hacer modelos complejos con geometrías complejas; problemas de estructuras y cargas no distribuidas, determinación de propiedades de materiales, mecánica de fluidos, problemas de conducción de calor que de otra forma con soluciones analíticas y ecuaciones diferenciales sería muy costoso.

El MEF determina el comportamiento de una estructura ante las cargas sustituyendo la solución continua exacta. Para esto, se

discretiza la estructura; es decir, se convierte en elementos no diferenciales o *elementos finitos*, interconectados entre sí a través de un determinado número de nodos.

Después de analizar cada elemento por separado se recompone la estructura restableciendo la estructura y la compatibilidad de desplazamientos en los nodos lo que da lugar a ecuaciones algebraicas. La resolución de este sistema de ecuaciones permite hallar los desplazamientos de los nodos y, a partir de ellos, las restantes incógnitas de la estructura. Es un método aproximado cuyo grado de aproximación aumenta con el número de elementos en que se divida la estructura. (Manuel Vázquez, 2001) y (Hurtado).

Estos autores mencionados en la anterior línea proponen la realización a detalle junto con los conceptos de elasticidad, matemáticos y energéticos.

Las fases de método son:

1. División en elementos finitos.
2. Vector de desplazamientos del elemento.
3. Matriz de rigidez del elemento.
4. Matriz completa de rigidez de la estructura
5. Respuesta de la estructura.

La división de elementos finitos o discretización de la estructura se realiza utilizando programas de cómputo especiales llamados *preprocesadores*; éste será el objetivo de la investigación, la aplicación de los triángulos de Delaunay como método de discretización que es la primera fase. La literatura mencionada no menciona sobre esto,

más bien se deja el trabajo a métodos matemáticos diversos; sin embargo, los triángulos de Delaunay por su mayor estructura equilátera de sus triángulos es ideal.

4.2 Elementos tetraédricos

4.2.1 Funciones de desplazamientos

El método de los elementos finitos generaliza sus conceptos a cualquier dimensión; pero en especial es resulta práctico en 3D aplicado a ingeniería civil y de construcción entre otras áreas. En nuestro caso serán tetraedros.

Considerando un elemento tetraédrico de *cuatro nodos*, que tiene un nodo en cada vértice y los desplazamientos nodales u_1, v_1, w_1, u_2, v_2, w_2, u_3, v_3, w_3 Fig.(4.1).

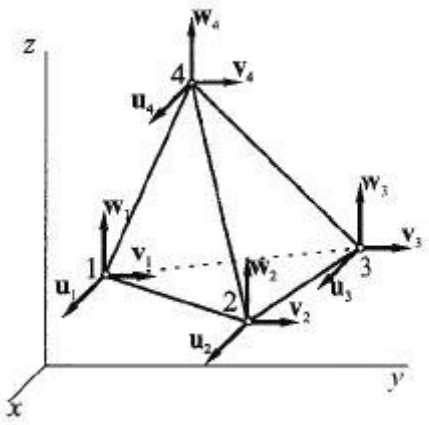


Fig. (4.1)

El vector de desplazamientos o la interpolación de los desplazamientos son:

$$u(x,y,z) = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z,$$

$$v(x,y,z) = \alpha_5 + \alpha_6 x + \alpha_7 y + \alpha_8 z,$$

$$z(x,y,z) = \alpha_9 + \alpha_{10} x + \alpha_{11} y + \alpha_{12} z.$$

Que puede transformarse en:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} =$$

$$\begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & N_4 & 0 & 0 \\ 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & N_4 & 0 \\ 0 & 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & N_4 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ w_1 \\ \cdot \\ \cdot \\ \cdot \\ u_4 \\ w_4 \\ w_4 \end{bmatrix}$$

Dónde

$$N_i = {}^{1}/_{6v} \left(a_i + b_i x + c_i y + c_i z\right)$$

siendo v el volumen del elemento:

$$6v = det \begin{bmatrix} 1 & x_1 & y_1 & z_4 \\ 1 & x_2 & y_2 & z_4 \\ 1 & x_3 & y_2 & z_4 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix}$$

$$a_i = det \begin{vmatrix} x_j & y_j & z_j \\ x_k & y_k & z_k \\ x_l & y_l & z_l \end{vmatrix} \quad b_i = -\, det \begin{vmatrix} 1 & y_j & z_j \\ 1 & y_k & z_k \\ 1 & y_l & z_l \end{vmatrix}$$

$$c_i = det \begin{vmatrix} x_j & 1 & z_j \\ x_k & 1 & z_k \\ x_l & 1 & z_l \end{vmatrix} \quad d_i = det \begin{vmatrix} x_j & y_j & 1 \\ x_k & y_k & 1 \\ x_l & y_l & 1 \end{vmatrix}$$

Al tomar en cuenta la relación entre deformación y desplazamiento, dada por

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \varepsilon_{xy} \\ \varepsilon_{xz} \\ \varepsilon_{yz} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

Los términos de la matriz de deformación B son constantes que dependen de las coordenadas x, y, z de los cuatro nodos. Se concluye que la matriz cinemática es:

$$B = (B_1 \quad B_1 \quad B_1 \quad B_1)$$

Donde

$$B_i = \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_i}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_i}{\partial z} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} & 0 \\ \frac{\partial N_i}{\partial z} & 0 & \frac{\partial}{\partial x} \\ 0 & \frac{\partial N_i}{\partial z} & \frac{\partial N_i}{\partial y} \end{bmatrix} = \frac{1}{6V} \begin{bmatrix} b_i & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & d_i \\ c_i & b_i & 0 \\ 0 & d_i & c_i \\ d_i & 0 & b_i \end{bmatrix}$$

CÁPITULO CINCO

GENERACIÓN DE ESFERAS DE COBERTURA MEDIANTE MÍNIMOS CUADRADOS

La solución por regresión de mínimos cuadrados es bastante eficiente; sin embargo, como se puede ver en el apartado del apéndice cuando no es lineal la complejidad aumenta considerablemente.

Una posible forma de generar una esfera usando Álgebra Geométrica es utilizar el producto "cuña" (Apéndice V); pero esto se deja como tema propuesto a investigar.

El algoritmo de ajuste de una esfera a puntos está basado en la recopilación que hizo Dietmar Hildenbrand (Hildenbrand, 2005), (Bindel, Lecture 17, Week 7, 2009).

La idea básica es obtener por mínimos cuadrados; dados *n* puntos, la esfera que los encierra con el mínimo error.

Dietmar y Hildenbrant obtienen los eigenvectores y eigenvalores por la función que tiene MATLAB como se verá en el próximo cuadro; sin embargo, yo hago mención a su realización completa, de forma "manual" y más adelante muestro su estructura.

Este algoritmo que muestro a continuación posé su fuerza en la generación de hiperesferas; (esferas en *n* dimensiones). El proceso es como se puede apreciar sencillo si se cuenta con MATLAB o un software especializado en matemáticas; sin embargo, si no es así se debe de tener al alcance un kit de bibliotecas matemáticas al alcance para ser incorporadas; vectores por ejemplo (estos se encuentran en la biblioteca STL -Standard Template Library-, la cual ya es un standard de C++), como de matrices y necesariamente de eigenvectores y eigenvalores; yo me encaminé por generarlas de forma personalizada. Si el caso del lector de esta tesis es ese entonces bastará con que eche un vistazo al algoritmo y se salte las consideraciones que propongo.

5.1 Ajustando una esfera a puntos.

La aproximación de esferas como vectores conduce a un método sencillo para el ajuste de una "esfera óptima".

Si medimos la distancia entre un punto X_i y una esfera S puede ser definida en Álgebra Geométrica Conformal mediante el producto punto:

$$X_i \cdot S = \left(P = \boldsymbol{x} + \frac{1}{2}x^2 e\infty + e0\right) \cdot \left(s = P + \frac{1}{2}r^2 e\infty\right) \quad (5.1)$$

Aplicando la ecuación del apéndice E3:

$$X_i \cdot S = p_i \cdot s - s_4 - \frac{1}{2}s_5 p_i^2$$

O de otra forma:

$$X_i \cdot S = \sum_{j=1}^{5} w_{i,j} s_j \quad (5.2)$$

Donde

$$w_{i,j} = \begin{cases} p_{i,k} & : \ k \in \{1,2,3\} \\ -1 & : \ k \in 4 \\ -\frac{1}{2}p_i^2 & : \ k \in 5 \end{cases}$$

Si tomamos el Mínimo Cuadrado de las distancias entre un punto y una esfera.

$$min \sum_{i=1}^{n} (x_i \cdot S)^2 \quad (5.1)$$

Con el fin de obtener el error mínimo podemos cambiarlo a su forma bilineal.

$$min\,(s^T \cdot Bs)^2 \quad (5.2)$$

Donde

$$s^T = (s_1, s_2, s_3, s_4, s_5)$$

$$\boldsymbol{B} = \begin{pmatrix} b_{1,1} & \cdots & b_{1,5} \\ \vdots & \ddots & \vdots \\ b_{5,1} & \cdots & b_{5,5} \end{pmatrix}$$

5.1.1 Código de Ajuste de Hiperesferas en MATLAB

```
function [RadioCua,Centro] = circ(puntos)
```

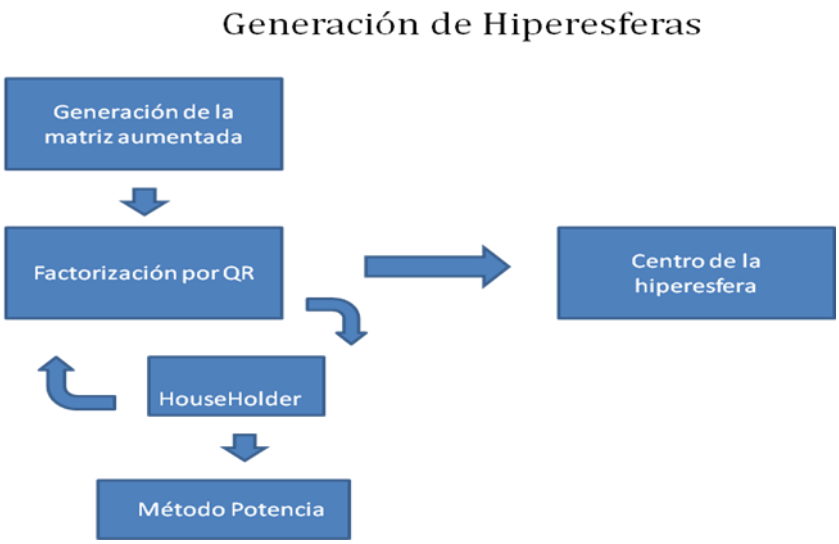
```
  W = puntos;

  [m,n] = size(W);

  for i=1:m,
     W(i,n+1)=-1.0;
     W(i,n+2)=-0.5*(sum( puntos(i,1:n).*
     puntos(i,1:n) ));
  end

  B=W'*W;

  [u,v]= eig(B);
  indices = diag(v);
  [val, ind]= min(indices);

  s = u(:,ind);
  s = (1/(s(n+2)))*s;
  Centro = s(1:n)

RadioCua = dot(Centro,Centro)-2*s(n+1);
```

5.2 Consideraciones

- No podemos obtener los eigenvectores y el eigenvalor por el método de potencia, ni por el de Gram-Schmidt ya que de entrada tenemos una matriz inconsistente; esto es debido a que la última columna es una combinación lineal de las demás.
- Para evitar la solución trivial; evitamos la factorización QR de la matriz con el método Gram-Schmidt y procedemos con HouseHolder (Bindel, Lectura 18, Week 7, 2009) (Podría ser también utilizando la descomposición del valor singular).
- Se recomienda implementar métodos numéricos utilizando el algoritmo QR para obtener la matriz final donde logramos los eigenvectores y eingenvalores.

5.3 Estructura del programa para hiperesferas.

Este esquema muestra un sencillo procedimiento; basta apuntar que una vez obtenidos el eigenvalor y el eigenvector mínimo (es el centro de la hiperesfera pues obtuvo el error mínimo) mediante la ecuación (5.2), es necesario un despeje de la fórmula de la esfera para obtener el radio.



5.3.1 Llamadas al programa hiperesferas

Tomando como ejemplo una esfera en 3D:

```
#include "puntos.h"
#include "Matriz.h"
#include "HiperEsf.h"

//  Se generan 3 puntos:

int ind1,ind2,ind3,ind4;
Punto Pto1(Dim), Pto2(Dim), Pto3(Dim), Pto4(Dim);
Pto1.SetVal(0,3); Pto1.SetVal(1,4); Pto1.SetVal(2,6);
Pto2.SetVal(0,8); Pto2.SetVal(1,2); Pto2.SetVal(2,1);
Pto3.SetVal(0,10); Pto3.SetVal(1,9); Pto3.SetVal(2,2);

Punto sPtos[] = {Pto1, Pto2, Pto3};
Punto Centro(2);                  // El parametro 2 es para
                              //establecer la dimensión
```

```
long double radio=0.0;
Matriz* M = new Matriz(sPtos, 3, 3);  // Se genera uma matrix 3x3
                            // parametros: renglon, columna

HiperEsf* HipEs = new HiperEsf(M);
HipEs->MakeHiperEsf(Centro,radio);       //Este método obtiene los
                                    //eigenvecotres y el eigenvalor
```

CORRIDA:

Este programa tardó 66.9 milisegundos.

CAPÍTULO SEIS

ALGORITMO PROPUESTO

En este capítulo reuniré los elementos anteriores analizados proyectando la forma de estos en la aplicación de los triángulos de Delaunay en el

Método de los Elementos Finitos; empezaré por describir algunas tareas que fueron indispensables para poder integrar los temas mencionados como un convertidor de formato de diseño 3D mqo a OpenGl y así como, se mencionará la importancia de las estructuras de datos en función del algoritmo de división espacial, la importancia de encontrar el punto más cercano y por último se añadirá el código de la aplicación tomando como ejemplo la deformación a un avión Harrier mediante MEF.

6.1 Importador y visualizador del formato 3D mqo

El formato mqo es un tipo de formato para representar objetos tridimensionales. Se tuvo que generar un importador-visualizador para este tipo de archivos e integrarlo en el código general.

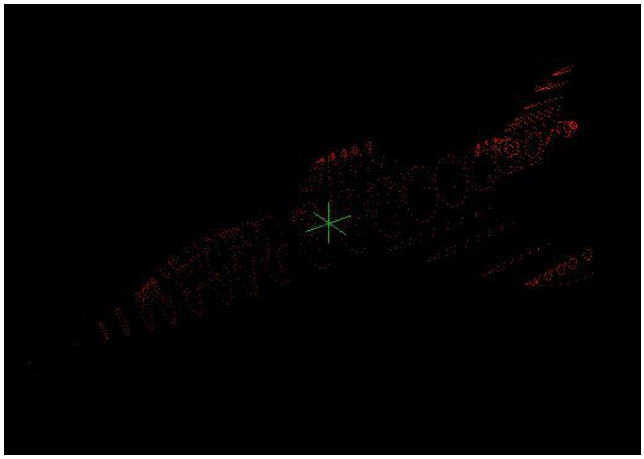
El siguiente código trabaja importando un parte del avión y al final se integra en un solo formato de texto plano.

```
#include <stdio.h>
#include <string.h>

int main()
{
    FILE *fp1, *fp2;
    char palabra[100];
       char palabraAnter[100];
       char Object[30];
       char Comando[100];
       int c;
       int Grabar=0;
       int contLlave=0, coor=0;
       int i=0;

       strcpy(Comando,"glVertex3f(");

       fp1 = fopen("Harrier.mqo", "r");
    fp2 = fopen("Salida.txt", "w");
```

```
            fprintf(fp2, "#include <glut.h>\n");
            fprintf(fp2, "void Carga(){ \n");
            fprintf(fp2, "glColor3f(1.0f, 0.0f, 0.0f); \n");
            fprintf(fp2, "glBegin(GL_POINTS); \n");

        do
        {
            c = fscanf(fp1, "%s", palabra);

                if(strcmp("\"TF\"",palabra) == 0){
                        strcpy(Object,"TF");
                    printf("Object es %s \n",Object);
                }
                if(strcmp("\"SL\"",palabra) == 0){
                        strcpy(Object,"SL");
                    printf("Object es %s \n",Object);
                }
                if(strcmp("\"Rumpf\"",palabra) == 0){
                        strcpy(Object,"Rumpf");
                    printf("Object es %s \n",Object);
                }
                if(strcmp("\"Haube\"",palabra) == 0){
                        strcpy(Object,"Haube");
                    printf("Object es %s \n",Object);
                }
                if(strcmp("\"Heck\"",palabra) == 0){
                        strcpy(Object,"Heck");
                    printf("Object es %s \n",Object);
                }

    ////////////////////////////////////////////////////
                if(strcmp("{",palabra)==0 && strcmp(Object,"Haube")==0)
                        contLlave++;

                if(contLlave==2)
                        Gravar=1;

                if(Grabar){
                        if(strcmp(palabra,"{")!=0 &&
    strcmp(palabra,"}")!=0){
                                coor++;
                                if(coor==1){
                                        strcat(Comando,palabra);
                                }
                                if(coor==2){
                                        strcat(Comando,", ");
                                        strcat(Comando, palabra);
                                }
                                if(coor==3){
                                        strcat(Comando," ,");
                                        strcat(Comando,palabra);
                                        strcat(Comando," );\n");
                                        fprintf(fp2, Comando);
                                        strcpy(Comando,"glVertex3f(");
                                        coor=0;
                                }
                        }
                        if(strcmp(palabra,"}")==0){
                               contLlave = 0;
                               Grabar=0;
                        }
```

```
            }

        }
        while (c != EOF);

           fprintf(fp2, "glEnd();\n} ");
        fclose(fp1);
        fclose(fp2);


        return 0;
}
```

Produciendo la siguiente salida:



6.2.2 Hiperesferas

La hiperesferas se representan como un conjunto de conjuntos; es decir, todas las hiperesferas que tienen en común a un punto serán agrupadas en un primer conjunto y de esa forma hasta que cada punto tenga su propio conjunto (hiperesferas circundantes a él).

Donde *N* es el número del hipercubo que conforma

Pto 1	…	…	…
Pto 1	….	…	….

Pto 1	….	…	….
Pto 1	…	…	….

6.3 Trabajos Futuros

Para perfeccionar este algoritmo he propuesto mejoras en la estructura de datos para hacer más eficiente la búsqueda:

6.3.1 Generación de Hipercubos Virtuales

Se necesitará conocer que punto de la estructura de datos es más cercano al punto que se va a integrar.

Con la finalidad de evadir el recorrido ciego a la llegada de un nuevo punto con los demás que se encuentran en la estructura de datos, y obtener así la proximidad a su vecino más cercano, se optaría por dividir el conjunto de puntos en hipercubos virtuales que contienen un conjunto de puntos más pequeño que el original; siendo una tabla de *nCelulas* de renglones y PTOxESF es la longuitud de todos los vectores y será arbitrario pues este valor depende del número de la subdivisión del universo de puntos.

La coordenada para el nuevo punto en el dominio de este hipercubo virtual es:

$$N = L^2 y + Lx + z\ ;$$

Donde *N* es uno de los *n* hipercubos conformados dentro del dominio.

L es la longitud del hipercubo. Este dato es arbitrario y dependiente de la precisión; es decir, entre más divisiones (hipercubos) tenga el hipercubo global será menor el tiempo de búsqueda para el nuevo punto a integrar.

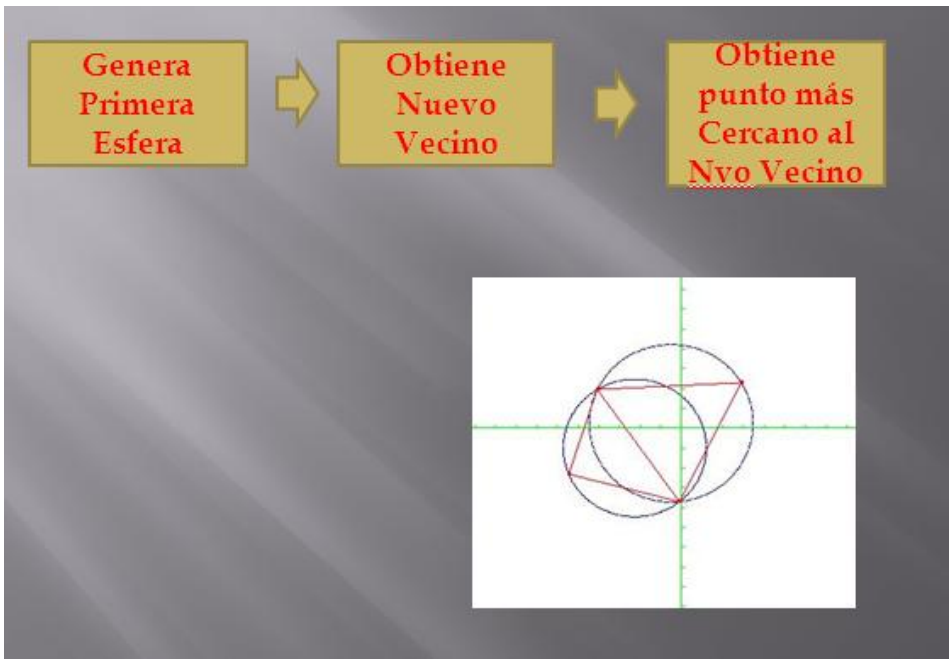
Estructura de HiperCubos:

Cuadrante	Pto 1	Pto 2	Pto …..	PTOSxESF
1				.
2				.
.				.
.				.
nCelulas				PTOSxESF

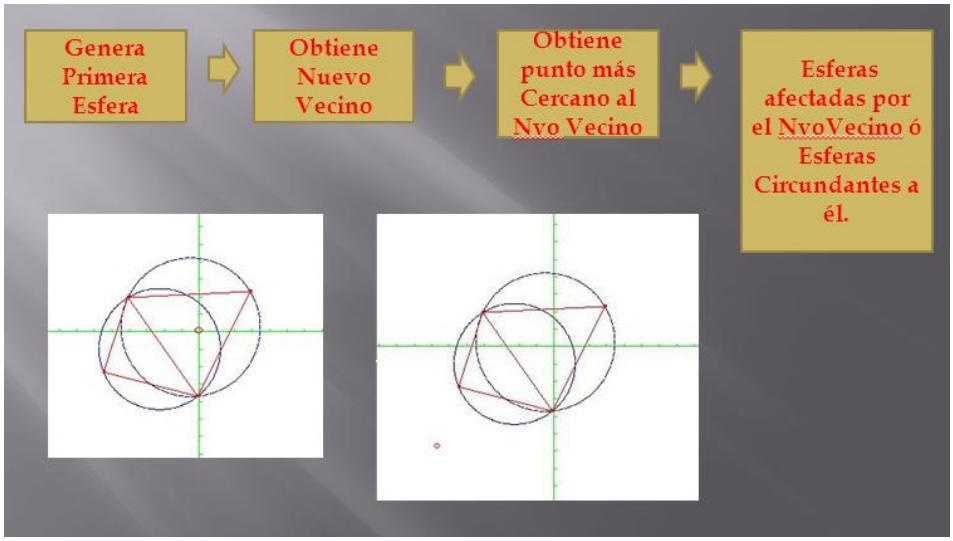
6.4 Algoritmo SemiDelaunay

El algoritmo general es:

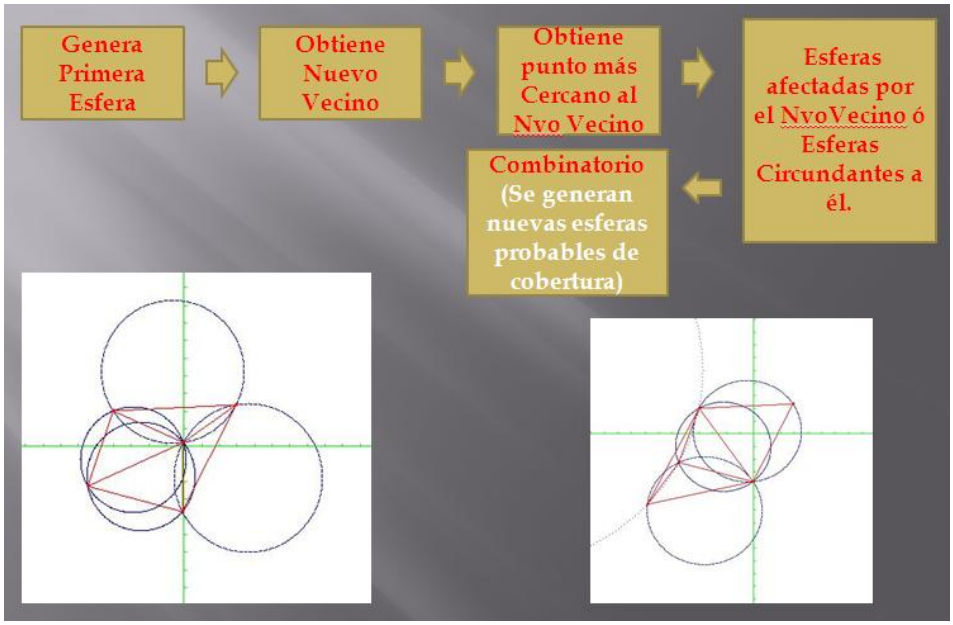
Estos primeros pasos son graficados con el sortware realizado para 2 Dimensiones:



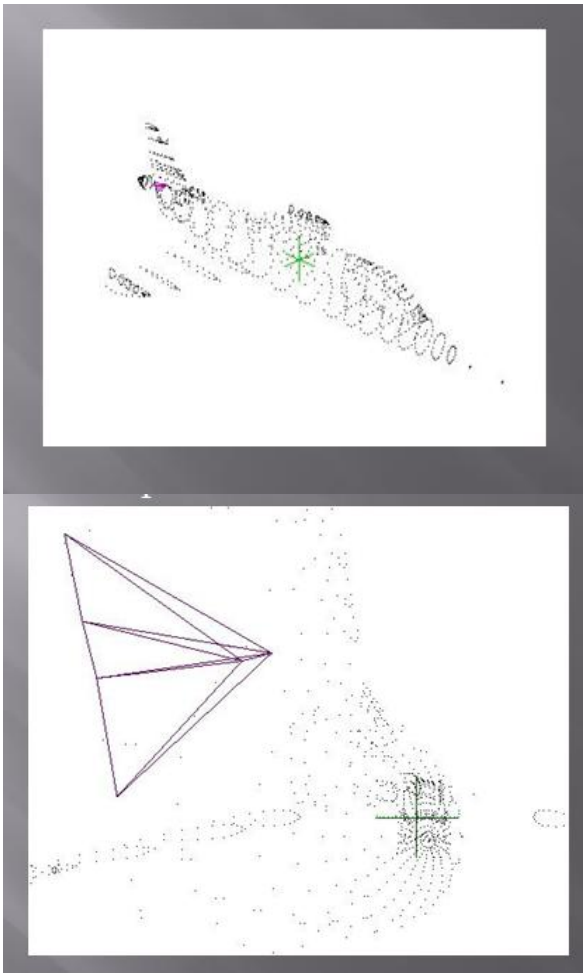
En esta gráfica podemos observar como el nuevo punto puede caer dentro o fuera de alguna esfera, si cae fuera se toma el punto más cercano a este y se prueba si forman esferas de Delaunay. De lo contrario se forman con el Cerco Convexo que lo encierra.



Para finalizar; los posibles resultados con las esferas de SemiDelaunay:



La siguiente gráfica muestra el comienzo de la triangulación en un objeto en 3D:



CONCLUSIONES

La cercanía a la triangulación de Delaunay se ha localizado a muy pocas iteraciones; 2 o 3. Esto varía con diversas variables:

La saturación de las hiperesferas. Una saturación excesiva es inecesaria pues generará triángulos de Delaunay bastante obtusos.

Puntos huérfanos. Es decir; puntos que no pertenecieron a ninguna hiperefera; esto puede

ser aceptable para una triangulación que sirva sólo de análisis general sin detalles extensos. En caso de que no se desee tener puntos huérfanos se puede implementar un ciclo extra de retroceso para incorporarlos al diagrama.

Este algoritmo puede suplir a los métodos que generan el Diagrama de Delaunay; sin embargo se deben de tener presente el nivel de precisión del diagrama para así, establecer la saturación de la triangulación.

BIBLIOGRAFÍA

Anton, H. (1999). *Introducción al Álgebra Lineal.* Mexico: Limusa Noriega Editores.

Bindel, F. (2009). Lectura 18, Week 7. *Matrix Computations.*

Bindel, F. (2009). Lecture 17, Week 7. *Matrix Computations.*

Buchmann A. Günther, S. T. (1990). *Desing and Implementation of Large Spatial Databases. Lecture Notes in Computer Scince .*

Burkardt, J. (2002). *The number of Nodes in a Delaunay Mesh Refinement.*

Canale, S. C. (1985). *Métodos numéricos para ingenieros con aplicaciones en computadoras personales.* Mc Graw Hill.

Hildenbrand, D. (2005). Geometric Computing in Computer Graphics using Conformal Geometric Algebra. *Geometric Algebras and Applications colloquium*, (pág. 10).

Hurtado, J. E. *Introducción al análisis estructural por elementos finitos.* Manizales.

Joseph Edward Shigley, C. R. (1995). *Diseño en Ingeniería Mecánica.* México: Mc Graw Hill.

Leo Dorst, D. F. (2009). *Geometric Algebra for Computer Science.* Morgan Kaufmann Publishers.

Liria, A. L. (2002). *Algoritmos para el Procesamiento de Consultas Espaciales utilizando R-trees. La Consulta de los Pares Más Cercanos y su Aplicación en Bases de Datos Espaciales.* Almeria.

Manuel Vázquez, E. L. (2001). *El método de los elementos finitos aplicado al análisis estructural.* Madrid: Noela .

Marsahll Bern, D. E. (98). *Mesh Generation and optimal triangulation.* California.

Mount, D. M. (2002). Computationay Geometry. Maryland, United States.

P. Cignonini, C. M. (1997). *DeWall: A fast Divide & Conquer Delaunay Triangulation Algorithm in Ed.* Italy.

Peter Su, R. L. (1996). *A Comparison of Sequential Delaunay Triangulation Algorithms.*

Shewchuk, J. R. (1999). *Lecture Note on Delaunay Mesh Generation.* Berkeley.

Trians, J. (2003). *Geometría para la informática gráfica y CAD.* Barcelona: Alfaomega grupo editor, S.A.

Zimmer, H. (2005). *Voronoi and Delaunay Techniques.*

APÉNDICE

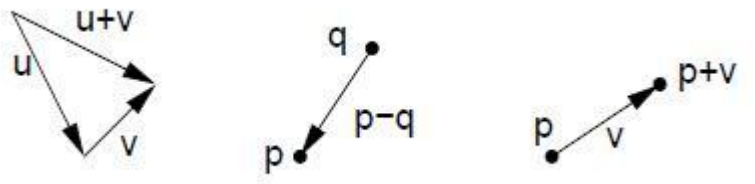
A GEOMETRÍA AFÍN

En geometría computacional pude manipularse algunos objetos geométricos; tales como, puntos, rectas, planos, polígonos, curvas y superficies; intuitivamente podemos ver estos objetos como conjunto de puntos; a tales objetos necesitaremos aplicarles alguna transformación; es por eso que conviene establecer un marco formal.

La geometría afín consiste en un conjunto de escalares, un conjunto de puntos y un conjunto de vectores. Los puntos son usados para especificar una posición, los vectores son usados para especificar dirección y magnitud, pero no tienen posición fija en el espacio. (Esto contrasta con el álgebra lineal donde no hay una distinción real entre puntos y vectores).

Las siguiente operaciones pueden ser realizadas por los escalares, puntos y vectores donde S es un escalar, V es un vector genérico y V un punto genérico.

$S \cdot V$	$\rightarrow$	V	multiplicación escalar-vector.
$V+V$	$\rightarrow$	V	Suma de vectores.
$P-P$	$\rightarrow$	V	Resta de vectores.
$P+V$	$\rightarrow$	P	Suma de punto-vector.



Estas operaciones pueden derivarse en otras más. Por ejemplo, podemos definir la resta de dos vectores, la división de un vector por un escalar y la definición de el *vector cero* y que no tiene magnitud.

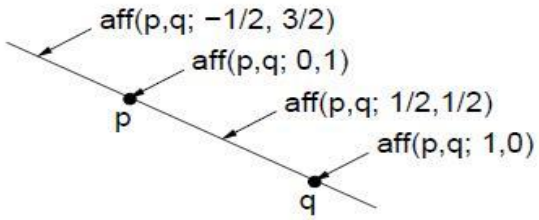
Notemos que no es posible multiplicar un punto por un escalar o sumar dos puntos. Pero existe una operación especial que combina estos dos elementos llamado *combinación afín*. Dado 2 puntos p_0 y p_1 y dos escalares α_0 y α_1, tal que $\alpha_0 + \alpha_1 = 1$, podemos definir la siguiente combinación afín.

$$aff\ (p_0, p_1; \alpha_0, \alpha_1) = \alpha_0 p_0 + \alpha_1 p_1 = p_0 + \alpha_1 (p_1 - p_0).$$

Vemos que el término medio no es legal debido a las bases de operaciones establecidas con anterioridad, esto es en realidad un promedio ponderado entre dos puntos. Una observación importante es que si $p_0 \neq p_1$, entonces el punto $aff(p_0, p_1; \alpha_0, \alpha_1)$ se encuentra entre p_0 y p_1. Como α_1 varia desde $-\infty$ a $+\infty$, esto hace que pueda encontrarse sobre todos los puntos de la línea.

En especial el caso donde $0 < \alpha_0, \alpha_1 \leq 1$, $aff(p_0, p_1; \alpha_0, \alpha_1)$ es un punto que subdivide el segmento de línea p_0p_1 en dos segmentos de tamaño proporcional a α_0 y $< \alpha_1$. La operación resultante es llamada *combinación convexa* si 0

$\leq \alpha_0$ y $\alpha_0 \leq 1$ y el conjunto de todas las combinaciones convexas es el segmento p_0p_1.



Es fácil extender ambas combinaciones a tres puntos (no colineales) $\alpha_0p_0 + \alpha_1p_1 + \alpha_2p_2 = 1$.

aff $(p_0, p_1, p_2; \alpha_0, \alpha_1, \alpha_2) = \alpha_0p_0 + \alpha_1p_1 + \alpha_2p_2 = p_0 + \alpha_1(p_1 - p_0) + \alpha_2(p_2 - p_0)$.

El conjunto de todas las combinaciones afines de tres puntos (no colineales) genera un plano. El conjunto de todas las combinaciones convexas de tres puntos generan todos los puntos del triángulo definido por los estos tres puntos. Estas formas son llamadas *cerradura afín* y *cerradura convexa* de los puntos, respectivamente.

B GEOMETRÍA GEOMÉTRICA

En geometría afín aún no se había hablado de distancias o ángulos. La geometría euclidiana es una extensión de la geometría afín que incluye una operación adicional llamada *producto interno*, que mapea dos vectores reales (no puntos) a un real no negativo. Un ejemplo importante de un producto interno es el *producto punto*, definido como sigue. Supongamos que el vector $\vec{u}$ es representado en dimensión d es representado por el (no homogéneo) vector (u1, u2, . . . , u_d). Entonces definimos.

$$\vec{u} \cdot \vec{v} = \sum_{i=0}^{d-1} u_i . v_i$$

Magnitud de un vector es $\vec{v} = \sqrt{\vec{v} . \vec{v}}$

Normalización: Dado cualquier vector no cero $\vec{v}$. Es un vector unitario en la misma dirección de $\vec{v}$.

$$\hat{v} = \vec{v}/|\vec{v}|$$

Distancia entre puntos: Se denota cualquier distancia dist(p,q) o |pq|como la longuitud que hay entre ellos, |p − q|.

Ángulo: El ángulo entre dos vectores no cero $\vec{u}$ y $\vec{v}$ (que van desde 0 hasta π) es:

$$\text{ang}(\vec{u}, \vec{v}) = cos^{-1}\left(\frac{\hat{u} \cdot \hat{v}}{|\vec{u}|\,|\vec{v}|}\right) = \cos^{-1}(\hat{u} \cdot \hat{v})$$

C REGRESIÓN CON MÍNMOS CUADRADOS

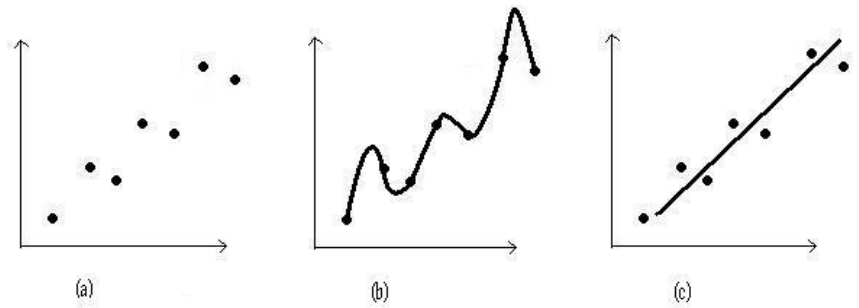
Cuando se asocia un error sustancial con los datos, la interpolación polinomial es inapropiada y puede llevar a resultados no satisfactorios cuando se usa para predecir valores intermedios. Los datos experimentales a menudo son de este tipo. Por ejemplo, en la figura 1a se muestran siete datos obtenidos experimentalmente que muestran una variación significativa. Si se ajusta un polinomio interpolante de sexto orden a estos datos (Fig 1b), pasará exactamente por todos los puntos. Sin embargo, debido a la variabilidad de los datos, la curva oscila ampliamente en los intervalos entre puntos.

Una estrategia más apropiada en estos casos es la de obtener una función aproximada que ajuste "adecuadamente" el comportamiento o la tendencia general de los datos, sin coincidir necesariamente con cada punto en particular. La figura 1c muestra una línea recta que puede usarse en la caracterización de la tendencia de los datos sin pasar sobre ningún punto en particular.

Una manera de determinar la línea de la figura 1c es inspeccionar visualmente los datos graficados y luego trazar la "mejor" línea a través de los puntos. Aunque este enfoque recurre al sentido común y es válido para cálculos "a simple vista" es deficiente ya que es arbitrario.

La manera de quitar esa subjetividad es considerar un criterio que cuantifique la suficiencia del ajuste. Una forma de hacerlo es

obtener una curva que minimice la diferencia entre los datos y la curva.



a) Muestra de datos con un error significativo. b) Ajuste polinomial con oscilaciones que violan el rango de los datos, c) se obtienen resultados más satisfactorios usando el ajuste de mínimos cuadrados (Canale, 1985).

C.1 REGRESIÓN LINEAL

El ejemplo más simple de una aproximación por mínimos cuadrados es el ajuste de una línea recta a un conjunto de parejas de datos observadas: (x_1, y_1), (x_2, y_2), (x_n, y_n). La expresión matemática de una línea recta es:

$$y = a_0 + a_1 x + E$$

En donde a_0 y a_1 son coeficientes que representan la intersección con el eje de las abscisas y la pendiente, respectivamente y E es el error o residuo entre el modelo y las observaciones, reordenando:

$$E = y - a_0 - a_1 x$$

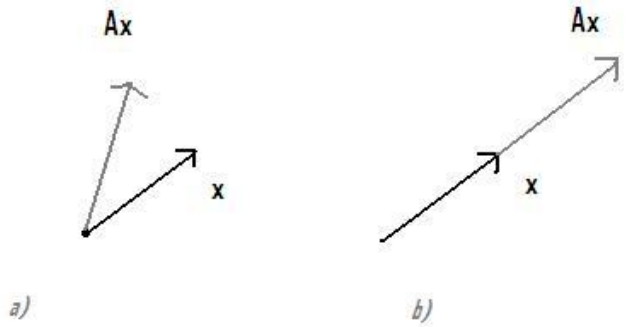
Por lo tanto, el error o residuo es la diferencia entre el valor real de y y el valor aproximado.

Una estrategia es la de minimizar la suma de los cuadrados de los residuos, S_r:

$$S = \sum_{i=1}^{n} E_i^2 = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)^2$$

D EIGENVECTORES Y EINGENVALORES

Si *A* es una matriz *n x n* y **x** es un vector en R^n, entonces no hay ninguna relación geométrica general entre el vector **x** y el vector *A***x** (figura a). Sin embargo, a menudo existen ciertos vectores **x** diferentes de cero tales que **x** y *A***x** son múltiplos escalares entre sí (figura b). Estos vectores surgen de manera natural en el estudio de vibraciones, sistemas eléctricos, genética, reacciones químicas, mecánica cuántica, esfuerzo mecánico, economía y geometría.

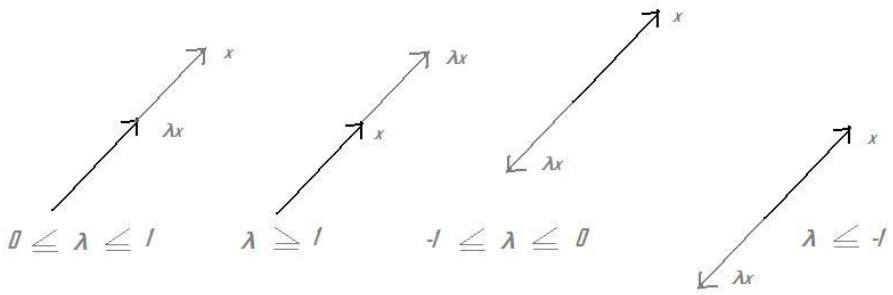


Definición. Si A es una matriz n X n, entonces un vector x diferente de cero en R n se denomina eigenvector de A si Ax es un múltiplo escalar de x; es decir,

$Ax = \lambda x$

para algún escalar λ. El escalar λ se denomina eigenvalor de A, y se dice que x es un eigenvector de A correspondiente a λ (Anton, 1999).

En R2 y R3, la multiplicación por A mapea cada eigenvector x de A (en caso de haber alguno) sobre la misma recta que pasa por el origen que x. Dependiendo del signo y la magnitud del eigenvector λ correspondiente a **x**,el operador lineal $A\mathbf{x} = \lambda\mathbf{x}$ hace que **x** se comprima o alargue por un factor λ, con un cambio de dirección en caso de que λ sea negativo (figura 2).



E ÁLGEBRA GEOOMÉTRICA CONFORMAL.

El Álgebra Geométrica Conformal en 5D que es una extensión del Álgebra Geométrica Proyectiva en 4D. En esta Álgebra las esferas y los círculos son representados como objetos; para representar círculos podemos intersectar dos esferas mediante una operación algebraica .

Las esferas, puntos, planos y demás figuras geométricas son representados como vectores; el producto punto "·" es útil para su medición, como veremos; también se utiliza el producto cuña "∧" para la formación de formas (Leo Dorst, 2009). Dos formas duales de representación son:

Entidad	**Representación 1**	**Representación 2**
Punto	$P = \mathbf{x} + \frac{1}{2} x^2 e\infty + e0$	
Esfera	$s = P + \frac{1}{2} r^2 e\infty$	*s = x1 ^x2 ^x3 ^x4*
Plano	$\pi = n + d e\infty$	*p = x1 ^x2 ^x3 ^e1*
Círculo	$Z = S_1 \wedge S_2$	*z = x1 ^x2 ^x3*
Línea	$L = \pi_1 \wedge \pi_2$	*l = x1 ^x2 ^e1*

La representación de un vector en Álgebra Conformal Geométrica 3D, puede ser escrito con los vectores base e_1, e_1 y e_1 como:

$$S = s_1e_1 + s_2e_2 + s_3e_3 + s_4e_\infty + s_5e_0$$

(E1)

Donde:

e_0 representa el origen en 3D.

e_0 representa el origen el punto en el infinito.

El producto punto de vectores de un vector P y un vector S es:

$$P \cdot S = (p + p_4 e_\infty + p_5 e_0) \cdot (s + s_4 e_\infty + s_5 e_0)$$

(E2)

Simplificando, obtenemos:

$$P \cdot S = p \cdot s - p_5 s_4 - p_4 s_5$$

(E3)

De aquí la distancia entre dos puntos:

$$P \cdot S = -\frac{1}{2}(s - p)^2 \qquad \text{(E4)}$$

F MECÁNICA

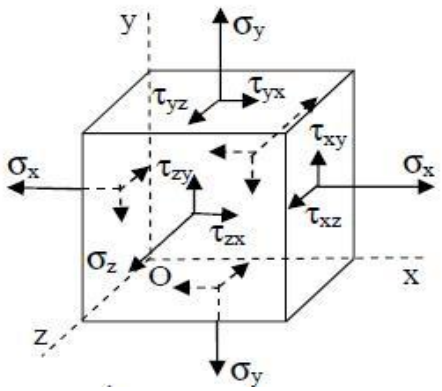


Figura 1

Componentes del esfuerzo

En la figura 1 se observa un elemento del estado general de esfuerzo tridimensional y se muestran tres esfuerzos normales, $\sigma_x, \sigma_y \, y \, \sigma_z$, todos positivos; y seis esfuerzos cortantes, $\tau_{xy}, \tau_{yx}, \tau_{yz}, \tau_{zy}, \tau_{zx}, \tau_{xz}$, también positivos. El elemento está en equilibrio estático y, por lo tanto, $\tau_{xy} = \tau_{yx}$, $\tau_{yz} = \tau_{zy}$, $\tau_{zx} = \tau_{xz}$

Los esfuerzos normales dirigidos hacia afuera del elemento se consideran positivos y son de tensión. Los esfuerzos cortantes que actúan sobre una cara positiva de un elemento son positivos, si se ejercen en la dirección positiva de un eje de referencia; éste es el caso de la figura 1. El primer subíndice de una componente de esfuerzo cortante es la coordenada normal (o perpendicular) a la cara del elemento. La componente de esfuerzo cortante es paralela al eje del segundo subíndice. Como el elemento que se presenta está en equilibrio estático, las caras negativas de dicho elemento tendrán esfuerzos cortantes que actúan en dirección opuesta, pero también se les considera positivos.

Lo que se dijo es el convenio clásico de los signos del esfuerzo (Joseph Edward Shigley, 1995).