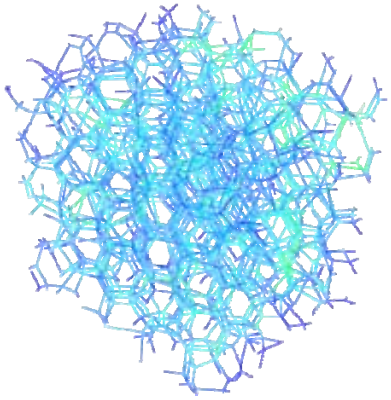
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ingeniería

Investigación "in silico" de las propiedades mecánicas del hueso trabecular



TESIS

QUE PARA OBTENER EL TÍTULO DE

INGENIERO MECÁNICO

PRESENTAN

ELLIOT ISMAEL BUSTILLO CASAS

HUMBERTO RAMIRO LÓPEZ CERVANTES

DIRECTOR DE TESIS

DR. RAFAEL SCHOUWENAARS FRANSSENS

México, Cd. Universitaria 2008

“Puede ser un héroe lo mismo el que triunfa que el que sucumbe,
pero jamás el que abandona el combate”
Thomas Carlyle

"El éxito parece estar relacionado con la acción.
Las personas de éxito son activas. Cometen errores pero no se rinden."
Conrad Hilton

AGRADECIMIENTOS

Humberto Ramiro López Cervantes:

En cada momento, con cada paso y decisión que tomamos vamos formando nuestro futuro, sin embargo, en esta formación no nos encontramos solos, existen personas que con su presencia hacen que cada momento sea único, que cada paso deje una huella imborrable en nuestra personalidad y que nos enseñan a asumir con responsabilidad cada decisión. A cada una de estas personas les agradezco por estar conmigo en el camino a este primer paso en el andar de mi vida. A quienes más agradezco es a mis padres, a mi madre porque con la fuerza y determinación de tus manos me enseñaste que nunca debo rendirme, que no debo detenerme hasta alcanzar lo que me propongo, que debo trabajar con honestidad y respeto, te agradezco todas las noches en vela que pasaste por acompañarme y por confiar en mi, a mi padre porque me enseñaste el valor de la disciplina, a asumir la responsabilidad de mis actos y que el trabajo duro es el camino para la superación, pero sobre todo les agradezco el amor, cariño y comprensión que siempre me demostraron, gracias papás.

A mi tutor, el Dr. Rafael Schouwenaars por las enseñanzas que a lo largo de este tiempo fueron determinantes en mi formación, el tiempo, dedicación y conocimientos fundamentales para el desarrollo de este trabajo, pero sobre todo le agradezco la confianza y apoyo que ha puesto en mí.

A la Universidad Nacional Autónoma de México por concederme la mejor instrucción durante estos años.

A la Facultad de Ingeniería por la formación proporcionada a lo largo de estos años.

A mis hermanos, José Luis porque me mostraste el valor e importancia que tiene amar tu profesión, Nohemi por explicarme a través de tu vida lo importante que es divertirse y disfrutar cada etapa de la vida, les agradezco por estar y compartir esta vida conmigo.

Con el afán de superarse y buscar una mejor calidad de vida parte de mi familia se ha separado, sin embargo, siempre he sentido su cariño, apoyo y confianza, por ello agradezco a mis tíos Florencio, Francisco, Mateo, Lázaro, Daniel, Gerardo y a mi primo Rubén por los cortos pero inolvidables y divertidos momentos que compartimos en nuestro pueblo San Mateo. A mi tía Rosalía por sus consejos, sus palabras de aliento y ante todo su cariño.

A mis primas Marina, Alejandra y Guadalupe por las muchas tareas que me ayudaron a hacer en mis primeros años de estudio, gracias por su paciencia. A mi tía Inocencia por los consejos y cariño que siempre me ha demostrado.

A mis mejores amigos, Othón por el apoyo que siempre me brindaste, por escucharme y aconsejarme en los momentos más difíciles, por que fuiste pieza clave en el desarrollo de este trabajo, Coronel por las competencias, por los consejos y por enseñarme la importancia de tener la mente fría en las decisiones importantes, pero sobre todo les agradezco su amistad, gracias.

A mis amigos Adriana, Sonia, Guerrero, Modelo, Elliot, Leo, Venancio, Chucho, Jesús V, Joel y Hugo por haber hecho de mi trayecto por la Facultad de Ingeniería una etapa muy feliz, divertida y satisfactoria.

A mis compañeros de la UDIATEM (Unidad de Investigación y Asistencia Técnica en Materiales) por permitirme formar parte del grupo. Un agradecimiento especial al M.I Edgar Isaac Ramirez por su tiempo, conocimientos y paciencia brindados a lo largo de este trabajo.

A cada persona que conocí aun cuando haya sido por poco tiempo, que me ha enseñado algo. Y a todos que por algún descuido no los agregué, gracias.

AGRADECIMIENTOS

Bustillo Casas Elliot Ismael:

A Dios por la identidad y propósito que ha dado a mi vida.

A mis padres por el apoyo, disciplina y respaldo brindado a lo largo de todos estos años, por su esfuerzo y sacrificio para ofrecerme mejores oportunidades, pero sobre todo por el amor que les motiva a darlo todo.

A la UNAM y a la Facultad de Ingeniería por haberme brindado la oportunidad de tener una educación de calidad en una de las mejores instituciones educativas a nivel mundial.

A mi director de tesis el Dr. Rafael Schouwenaars Franssens por su tiempo, dirección y conocimientos que hicieron posible la elaboración de esta tesis.

Al M.I. Edgar Isaac Ramírez Díaz por los conocimientos y equipo aportados para la mejora de este trabajo.

A Ammy por su amor, intercesión y sabiduría que han hecho de mí una mejor persona, por esos momentos inolvidables, las palabras de aliento en momentos difíciles, por creer en mí, pero sobre todo por mostrarme el camino.

A mis hermanas por su compañía y aportaciones a mi vida.

A todos mis amigos por hacer de mi trayecto por la Universidad una etapa agradable y en especial a Humberto por invitarme a formar parte de este proyecto.

ÍNDICE

PRÓLOGO

En los últimos años la necesidad de contar con información precisa acerca de las propiedades mecánicas del hueso surge conjuntamente con la de diseñar prótesis, desarrollar materiales que lo puedan sustituir, evaluar el deterioro con la edad y los padecimientos así como para definir procedimientos terapéuticos. Para lograr esto es indispensable poder modelar el comportamiento mecánico óseo, es por ello que se ha desarrollado este trabajo que busca aportar a la reciente investigación alrededor de este tema.

El modelado del comportamiento mecánico del hueso humano es sumamente complejo, si se parte del hecho de que estructuralmente podemos hablar de dos tipos de hueso, el trabecular y el compacto, siendo estos diferentes en cuanto a su comportamiento mecánico y funcionalidad, sin dejar de lado que la estructura y propiedades del hueso como unidad se ven afectadas por la edad, sexo, patologías del sujeto, etc.

Siguiendo la línea de autores como *Gibson (1985), Baupre y Hayes (1985), Silva y Gibson (1997), Ramírez (2007)* entre otros, quienes se enfocaron en el modelado del hueso trabecular debido a que este tipo es el punto central en las fallas presentes en los huesos, dado que si bien la resistencia de la estructura depende fundamentalmente del hueso cortical, la experiencia muestra que las fracturas se generan a partir del hueso trabecular. Por lo anterior el objetivo principal de este trabajo es desarrollar la metodología para la construcción de un modelo numérico que se aproxime a la geometría del hueso trabecular. Asimismo, se simula mediante paquetería de elemento finito ensayos de compresión y deslizamiento[1] a estructuras virtuales de hueso trabecular, determinando de esta forma las constantes elásticas de dichas estructuras.

Los capítulos que abordan este trabajo de tesis son los siguientes.

El primer capítulo se denomina "Anatomía y mecánica del hueso" y pretende exponer los conocimientos básicos sobre la clase de hueso que existen, su morfología y características generales, haciendo énfasis en su estructura ya que tiene influencia importante en sus propiedades mecánicas. Es decir, se establece una idea muy clara del material que se está estudiando.

[1] Prueba mecánica en la cual a un cubo se le aplica un desplazamiento a la cara superior de éste en dirección paralela a la misma y se le restringe el movimiento a la base en la dirección de la solicitación.

El segundo capítulo "Diagramas Voronoi" se dedica a describir los aspectos generales de este concepto geométrico, tal como la definición formal, aplicaciones, algoritmo de generación y propiedades del mismo. Con esto se busca presentar al lector una visión general de la herramienta conceptual usada para aproximarse a la mesoestructura del hueso trabecular.

El capítulo tres, nombrado "Construcción del diagrama de Voronoi 3D" presenta el programa desarrollado en la paquetería MATLAB® que permite generar el diagrama de Voronoi en el espacio tridimensional a partir del cual se obtiene probetas cúbicas virtuales, lo que se busca en éste es mostrar como se realizó el programa, para lo cual se presenta de manera detallada los comandos, la forma y lógica utilizada. Esto se hace con el afán de que este trabajo sirva de referencia para futuros trabajos.

El capítulo "Estructura del archivo de *inp* para ABAQUS®" tiene por objetivo describir la estructura y comandos utilizados para generar un archivo de entrada *inp* para la paquetería de análisis por elementos finitos ABAQUS®, creado para simular sobre las probetas virtuales generadas en MATLAB®, ensayos de compresión y deslizamiento.

El capítulo cinco "Propiedades mecánicas para las probetas virtuales" tiene por objetivo presentar las propiedades mecánicas determinadas para los arreglos de Voronoi delimitados en un cubo, dichas propiedades determinadas a través de los ensayos de compresión y deslizamiento simulados en ABAQUS®. Se busca mostrar la forma en que se determinaron dichas propiedades.

El último capítulo toma como base los resultados de las propiedades mecánicas determinadas para las probetas virtuales y presenta un análisis de dichos resultados, así como las conclusiones a las que se pudieron llegar sobre los modelos, su aproximación al comportamiento mecánico del hueso trabecular y del método de generación de dichos modelos.

I. ANATOMÍA Y MECÁNICA DEL HUESO

I.1 Introducción

El hueso es un tipo de tejido conectivo especializado, característico por su rigidez y dureza como producto de las sales impregnadas en su estructura agregándole además que es un material auto reparable capaz de adaptar su masa, forma y propiedades a los requerimientos mecánicos y metabólicos.

El esqueleto en el ser humano lo conforman alrededor de 208 huesos formados por tejido óseo, cartílagos, médula ósea y el periostio. Estos huesos se pueden clasificar en:

- Huesos largos: huesos duros y densos que brindan resistencia, estructura y movilidad, por ejemplo el fémur.
- Huesos cortos: con mediciones de largo, ancho y alto aproximadamente iguales, como los de las manos y pies
- Huesos planos: están compuestos de una capa de hueso esponjoso entre dos capas delgadas de hueso compacto. Su forma es aplanada, no redondeada, por ejemplo los huesos del cráneo y las costillas

Las principales funciones del hueso son; *protección de órganos internos, almacenamiento, soporte, movimiento y regulación iónica*. Éstas lo hacen de suma importancia para estudios biomecánicos. Es por ello que en este trabajo se estudia el comportamiento mecánico del hueso con base en su estructura.

I.2 Estructura

La estructura de un hueso largo se compone de diferentes tipos de tejido (Figura I.1). La *diáfisis* es la parte alargada del hueso, la *epífisis* se encuentra en los extremos o terminaciones del hueso, la *metáfisis* es la unión de la diáfisis con las epífisis, el *cartílago articular* es una fina capa de cartílago hialino que recubre la epífisis donde el hueso se articula con otro hueso (reduce la fricción y absorbe los impactos y vibraciones), el *periostio* es la membrana que rodea la superficie del hueso no cubierta por cartílago (esencial en el crecimiento óseo, en su reparación y en su nutrición), la *cavidad medular* es un espacio cilíndrico situado en la parte central de la diáfisis, dicha cavidad está tapizada por el *endostio*, una membrana que contiene las células osteoprogenitoras.

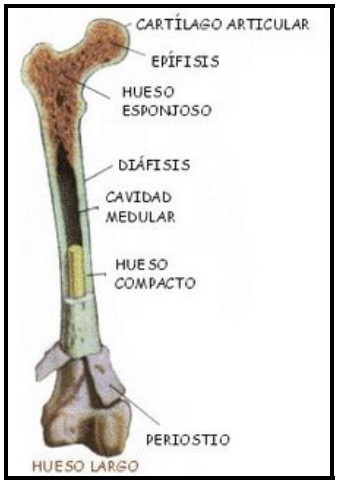


Figura I.1 Esquema de un hueso largo
(http://cienciasnaturales-bio.blogspot.com)

I.3 Tejido óseo

Como otros tejidos conjuntivos, el hueso o tejido óseo está constituido por una matriz en la que se encuentran células dispersas. La matriz está constituida por 25% de agua, 25% de proteínas y 50% de sales minerales. Más del 90% de ella corresponde a fibrillas de colágeno 1 organizadas en laminillas de unos 5 μm de grosor (Figura I.2) y el 10% restante lo constituyen células, vasos sanguíneos y prolongaciones celulares.

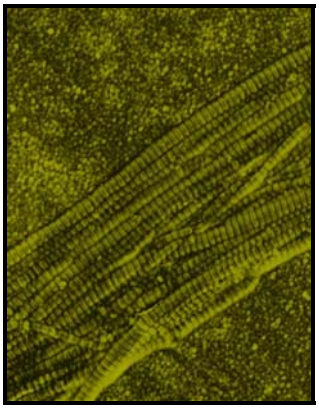


Figura I.2 Fibrillas de Colágeno
(http://www.electronicafacil.net/ciencia/Article13251.html).

Las células que se pueden encontrar en la matriz ósea son:

a) *Células osteoprogenitoras*: células no especializadas derivadas del mesénquima, que es el tejido del que derivan todos los tejidos conectivos. Se encuentran en la capa interna del periostio, en el endostio y en los canales del hueso que contienen los vasos sanguíneos. A partir de ellas se generan los osteoblastos y los osteocitos
b) *Osteoblastos*: son células que forman el tejido óseo pero que han perdido la capacidad de dividirse por mitosis. Segregan colágeno y otros materiales

utilizados para la construcción del hueso. Se encuentran en las superficies óseas y a medida que segregan los materiales de la matriz ósea, esta los va envolviendo, convirtiéndolos en osteocitos.

c) *Osteocitos*: son células óseas maduras derivadas de los osteoblastos que constituyen la mayor parte del tejido óseo. Al igual que los osteoblastos han perdido la capacidad de dividirse. Los osteocitos no segregan materiales de la matriz ósea y su función es la mantener las actividades celulares del tejido óseo como el intercambio de nutrientes y productos de desecho.

d) *Osteoclastos*: son células derivadas de monocitos circulantes que se asientan sobre la superficie del hueso y proceden a la destrucción de la matriz ósea (resorción ósea).

Las sales minerales más abundantes son la hidroxiapatita (fosfato tricálcico) y carbonato cálcico. En menores cantidades hay hidróxido de magnesio, cloruro y sulfato magnésicos. Estas sales minerales se depositan por cristalización en el entramado formado por las fibras de colágeno, durante el proceso de calcificación o mineralización. (*http://www.nlm.nih.gov/medlineplus*)

I.4 Clasificación

El hueso no es totalmente sólido sino que tiene pequeños poros entre sus componentes, formando pequeños canales por donde circulan los vasos sanguíneos encargados del intercambio de nutrientes. En función del tamaño de estos poros, el hueso se clasifica en compacto y esponjoso, también conocido como hueso trabecular o poroso, aunque en ambos casos, se trata de la misma composición.

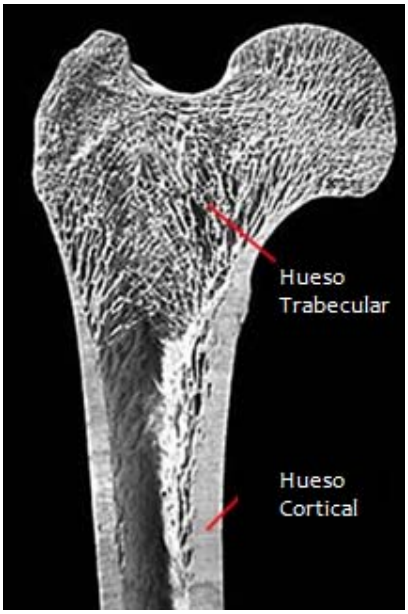


Figura I.3 Hueso trabecular/Hueso cortical
(Fotografía de Paul Crompton ©Escuela de Medicina, Universidad de Wales)

I.4.1 Hueso compacto

El hueso compacto, también conocido como *cortical*, constituye la mayor parte de la diáfisis de los huesos largos así como de la parte externa de todos los huesos del cuerpo. El hueso compacto es el que se encarga de brindar soporte y protección.

Dicha estructura presenta un 10% de porosidad. Tiene una estructura de láminas o anillos concéntricos alrededor de canales centrales llamados canales de Havers (miden de 30 a 70 μm de diámetro) que se extienden longitudinalmente. Los canales de Havers están conectados con otros canales llamados canales de Volkmann que perforan el periostio. Ambos canales son utilizados por los vasos sanguíneos, linfáticos y nervios para extenderse por el hueso.

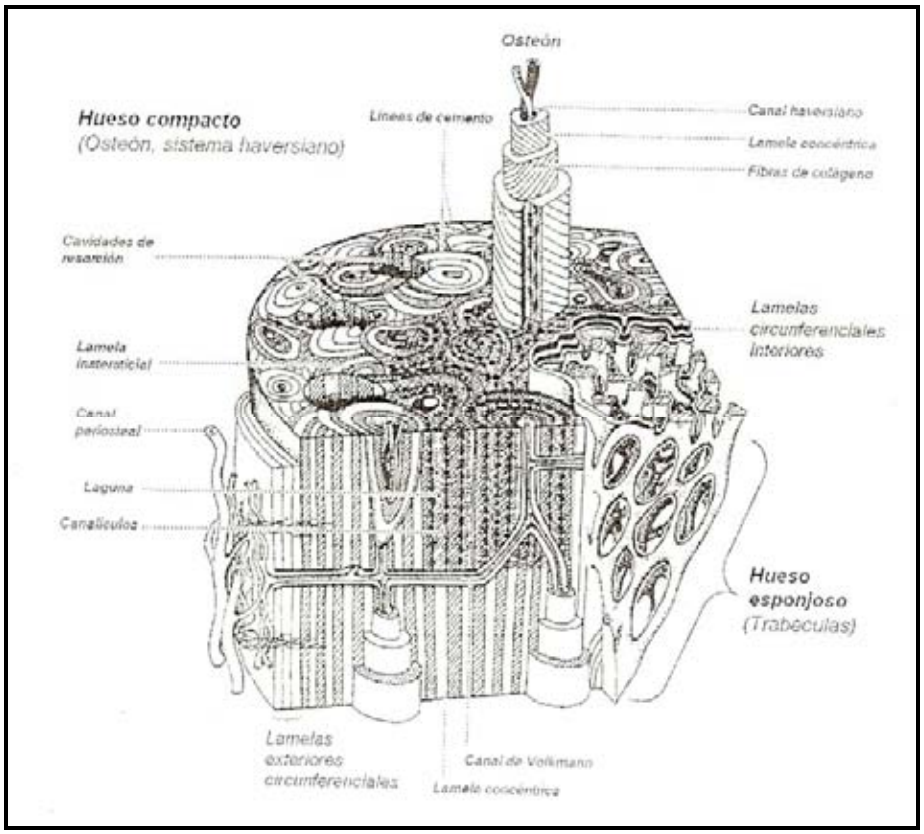


Figura I.3 Diagrama de las estructuras óseas en el hueso cortical (Weiss, 1988)

Entre las láminas concéntricas de matriz mineralizada hay pequeños orificios o lacunae donde se encuentran los osteocitos. Para que estas células puedan intercambiar nutrientes con el líquido intersticial, cada lacunae dispone de una serie de canalículos por donde se extienden prolongaciones de los osteocitos. Los canalículos están conectados entre sí y, eventualmente a los canales de Havers.

I.4.2 Hueso esponjoso

A diferencia del hueso compacto, el hueso esponjoso no contiene osteones, sino que las láminas intersticiales están dispuestas de forma irregular formando unas placas llamadas trabéculas. Estas placas tienen un espesor menor a 0.2 mm y forman una estructura esponjosa. En el caso de trabéculas de espesor mayor a 0.2 mm suelen aparecer estructuras similares a los osteones.

Tiene la misma composición que el hueso compacto pero con una porosidad que varía entre 50 y 90%. Dentro de las trabéculas están los osteocitos que yacen en sus orificios o lacunae con canalículos que irradian desde las mismas. En este caso, los vasos sanguíneos penetran directamente en el hueso esponjoso y permiten el intercambio de nutrientes con los osteocitos.

El hueso esponjoso es el principal constituyente de las epífisis de los huesos largos, sin embargo, mientras que en un hueso largo se puede encontrar hasta un 90% de hueso cortical y solo 10% de hueso trabecular, en otros como las vertebras la proporción es de 62% y 38% respectivamente.

El hueso trabecular tiene una velocidad de restitución 10 veces mayor (es decir, la pérdida y formación de hueso) que el hueso cortical: en un adulto sano, cerca del 25% del hueso trabecular se degrada y es reemplazado anualmente, en comparación al 3% en el hueso cortical. Esto se debe a que el hueso trabecular tiene una mayor superficie y un contacto más cercano con la médula, aumentando su vulnerabilidad a los cambios en el micro-ambiente óseo. Esta velocidad de restitución mayor significa que los huesos compuestos principalmente de hueso tipo trabecular (cadera y vértebras) pueden ser áreas de alto riesgo para fracturas en la vejez.

I.5 Propiedades mecánicas del hueso

El tejido óseo posee dos características importantes desde el punto de vista mecánico:

1. Por una parte se puede considerar un material compuesto resultado de la unión de dos fases: una orgánica y otra inorgánica. Esta estructura proporciona una gran resistencia que se ve aumentada por la disposición alternante de la hidroxiapatita en las fibras de colágeno.
2. Su porosidad variable en función de la zona del hueso analizada lo hace un material anisotrópico. La anisotropía implica propiedades mecánicas diferentes dependiendo de la dirección de las solicitaciones ejercidas sobre él. Esta porosidad y la orientación de las trabéculas contribuyen a este efecto.

El tejido óseo soporta mejor las cargas de compresión que las de tracción. Sin embargo, la inserción de ligamentos y la acción de los músculos transforman las cargas de tracción en cargas de compresión.

La presión ejercida sobre un hueso incide directamente sobre las trabéculas. La presencia de unas membranas formadas de colágeno y calcio en la superficie de las trabéculas permite distribuir en forma homogénea y en todas direcciones la carga recibida. Las membranas localizadas debajo del cartílago (subcondral) son especialmente importantes ya que absorben y distribuyen las cargas provenientes de la articulación. Dichas cargas se repartirán a través del hueso esponjoso hacia el hueso cortical (R. Ballesteros, 2004).

El hueso posee un comportamiento viscoelástico, recuperándose de la deformación con una cierta lentitud si no se rebasa el límite elástico. Al aumentar la velocidad de aplicación de la fuerza, el hueso se deforma menos y se fractura antes.

I.5.1 PROPIEDADES MECÁNICAS DEL HUESO ESPONJOSO

Debido a que el presente trabajo se enfoca al análisis del hueso esponjoso, se dedicará esta sección para describir de manera particular las propiedades mecánicas de este tipo de hueso.

En los huesos largos, el hueso esponjoso es el que permite absorber la energía de impacto que se produce en las articulaciones distribuyendo las cargas mecánicas hacia el hueso cortical. La estructura tubular del hueso cortical le permite soportar la compresión, tracción, flexión y torsión o la combinación de ellas, reduciendo posibles fracturas. El hueso esponjoso se encuentra de igual manera en los cuerpos vertebrales, en donde su principal función es la de absorber las cargas (E. Ramírez, 2007).

La resistencia del hueso esponjoso es un factor determinante para la fractura del hueso, ya que es en esta zona donde se observa de manera más clara el efecto de enfermedades como la osteoporosis. No obstante, sus propiedades elásticas permiten determinar las condiciones de carga en las regiones de falla del hueso esponjoso, además, determinan el comportamiento mecánico del hueso esponjoso bajo condiciones comunes de actividad. Los experimentos muestran que durante tales actividades, el comportamiento del hueso puede ser considerado dentro del rango lineal.

La gráfica esfuerzo-deformación del hueso puede ser dividida en dos regiones: la región de deformación elástica y la región de deformación permanente (Turner, 1993). Dentro de la región elástica, la deformación en el hueso aumenta con el incremento de la carga y regresa a su forma original al dejar de aplicar la carga. Los efectos viscosos durante su deformación son debidos a fluidos en la matriz del hueso que causan pérdidas en la energía elástica, pero, de no considerar una aproximación matemática que trate al hueso elásticamente, el análisis de esfuerzos resultaría sumamente complicado.

La pendiente de la curva esfuerzo-deformación dentro de la región elástica del hueso define el módulo de Young, el cual representa la rigidez intrínseca del material compuesto. En el hueso esponjoso en cambio, cada una de las trabéculas del material tiene su propio módulo y en conjunto forman una estructura que tiene su propia rigidez. Por lo tanto, en el hueso esponjoso se puede hablar de *módulo material*; para indicar la rigidez de cada trabécula, y de *módulo estructural;* para indicar la rigidez del conjunto de trabéculas.

En el hueso esponjoso se considera un material anisotrópico ya que existe una dirección preferencial en las trabéculas causando que el módulo de Young varíe en función de la dirección en la que se analice. El módulo de Young también varía dependiendo de la región que se examine, convirtiéndolo así, en un material no homogéneo. Experimentalmente se han registrado valores para el modulo de elasticidad del hueso esponjoso desde 0.76 hasta 20 GPa (Keaveny, 1993).

A diferencia del hueso cortical, el hueso esponjoso, como ya se ha mencionado, posee una estructura irregular basada en una red de vigas (trabéculas) orientadas de manera aleatoria debido a su crecimiento natural y a la habilidad de adaptación del hueso a las cargas a las que está sujeto. La aplicación de la teoría de celdas de Voronoi se considera una opción factible para generar una aproximación de esta geometría y obtener así una estructura que presente un comportamiento elástico estadísticamente similar al del hueso esponjoso. Dada la importancia de este concepto, el siguiente capitulo se dedica a la descripción de los aspectos generales de esta herramienta geométrica.

II. DIAGRAMA DE VORONOI

II.1 Introducción

Los diagramas de Voronoi fueron por primera vez discutidos por Peter Lejeune-Dirichlet en 1850, pero fue hasta mucho después en 1908 que estos diagramas fueron descritos por Georgy F. Voronoi.

El diagrama de Voronoi se define como una estructura geométrica que representa información de proximidad acerca de un conjunto de puntos u objetos, es decir, el diagrama de Voronoi de un conjunto de objetos geométricos es una partición del espacio en celdas, cada una de las cuales contiene una colindancia con sus puntos más cercanos. Se puede decir entonces que un diagrama de Voronoi consiste en la subdivisión del plano en regiones tales que todos los puntos de esa región están más cerca del nodo al que está asociada que de ningún otro.

Los diagramas de Voronoi permiten dividir una región en polígonos que dependen de la configuración de los puntos muestreados. Si los datos son regularmente espaciados en red, entonces se producirá un arreglo de polígonos cuadrados. Si los puntos están irregularmente espaciados se produce un arreglo de áreas poligonales irregulares, como se observa en la figura II.1

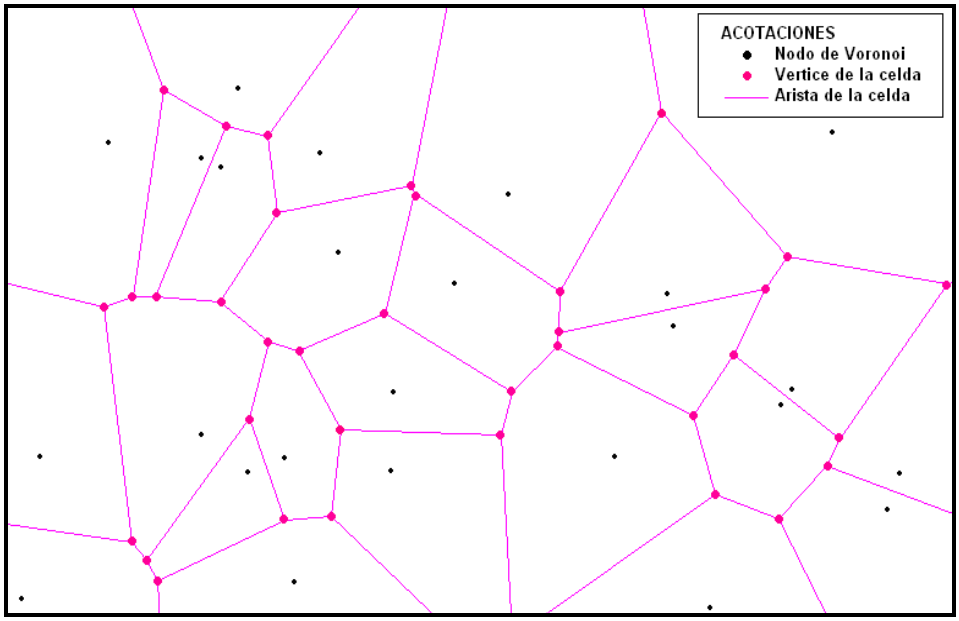


Figura II.1 Diagrama de Voronoi de un muestreo de puntos aleatorios en dos dimensiones creado en la paquetería *DELONE (Desarrollado por Alfredo Vegas Acitores. Sin publicar)*

Estas estructuras geométricas tienen numerosas aplicaciones en diversas áreas que van desde la arqueología hasta la navegación robótica. Por ejemplo, los diagramas de Voronoi son esenciales en la solución del problema de la comunicación inalámbrica. Dado un conjunto de subestaciones. ¿Cómo pueden ser pre-procesadas de modo que para cualquier llamada celular la estación más cercana lleve la llamada? El diagrama de Voronoi da respuesta a esta pregunta con base a que su principal cualidad es la información de proximidad que proporciona. Sin embargo el área de aplicación que es de interés para este trabajo es la biomecánica, dado que en las investigaciones de los últimos años se han tenido resultados satisfactorios, donde los diagramas de Voronoi han sido el método de generación de la arquitectura trabecular del hueso (*Silva y Gibson (1997), Ramírez (2007)*), donde se aposto por generar la geometría de primer nivel para aproximar el comportamiento mecánico de un modelo numérico al comportamiento mecánico del hueso trabecular. Esta aplicación se ha llevado a cabo en dos dimensiones, que como se menciono se obtuvieron buenos resultados, en base a ello este trabajo tiene como objetivo extenderlo al espacio tridimensional, con el objeto de aproximarse en mayor proporción al comportamiento del hueso trabecular.

Un aspecto que vale la pena subrayar, es que la similitud que existe entre la arquitectura del hueso trabecular y los diagramas de Voronoi no es la única existente en la naturaleza, hay varios ejemplos donde dichos diagramas tienen cierta aproximación con algunas geometrías propias de la biosfera, en la figura II.2 se muestran como las marcas del caparazón de una tortuga siguen aproximadamente las líneas de lo que es el diagrama de Voronoi.

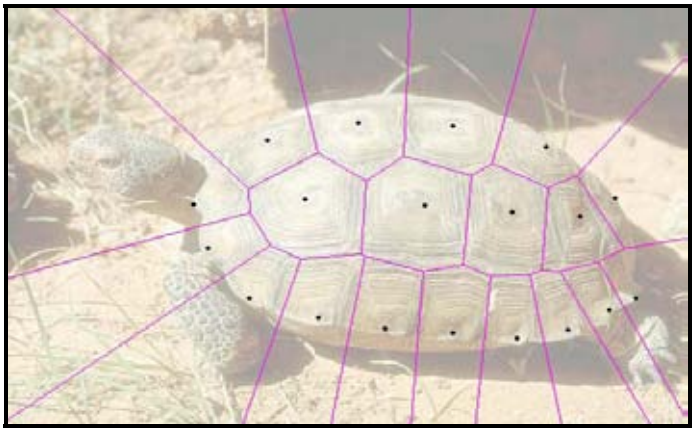


Figura II.2 Celdas de Voronoi en la naturaleza. Imagen creada en la paquetería *DELONE (Desarrollado por Alfredo Vegas Acitores. Sin publicar)*

En base a la importancia que el diagrama de Voronoi representa para este trabajo, se presenta a continuación el proceso de generación de este diagrama en el espacio bidimensional, dado la facilidad que el plano presenta para la explicación de dicha generación.

II.2 Construcción del diagrama de Voronoi

Con el objeto de dejar explicito el proceso de construcción de las celdas del diagrama de Voronoi, en esta sección del trabajo llevará a cabo la metodología de construcción de dichas celdas a partir de un conjunto de puntos aleatorios. Dicho conjunto se muestra a continuación.

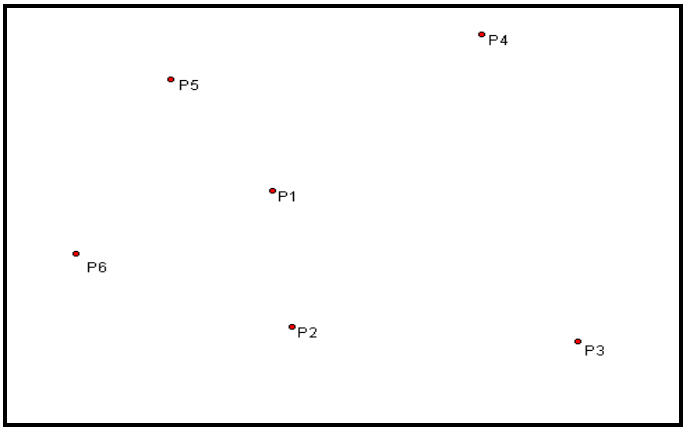


Figura II.3 Distribución de puntos aleatorios

Como principio para la construcción del diagrama es importante observar y determinar la colindancia que existe entre los puntos en base a los cuales se generaran las celdas correspondientes. Para el conjunto de puntos mostrados en la figura III.3, se observa que los vecinos más próximos para los siguientes puntos están dispuestos de la siguiente forma.

- P_1 vecinos: P_2, P_3, P_4, P_5 y P_6
- P_2 vecinos: P_1, P_3 y P_6
- P_3 vecinos: P_1, P_2 y P_4
- P_4 vecinos: P_1, P_3 y P_5
- P_5 vecinos: P_1, P_4 y P_6
- P_6 vecinos: P_1, P_2 y P_5

Con la determinación de la colindancia entre puntos, es posible comenzar con la generación de la celdas, para ésto se escoge cualquiera de los seis puntos, el orden en que se escoge el punto de inicio no altera el diagrama de Voronoi final. Dado esto, para comenzar la construcción de dicho diagrama, se escoge P_2 del cual se observa que tiene proximidad con P_1, P_3 y P_6. En primera instancia se escoge P_2 con cualesquiera los puntos con que tiene proximidad, en este caso se toma P_2 y P_3 con estos puntos se traza una recta y se determina también el punto medio mp_1 de dicha recta como se muestra la siguiente figura II.4

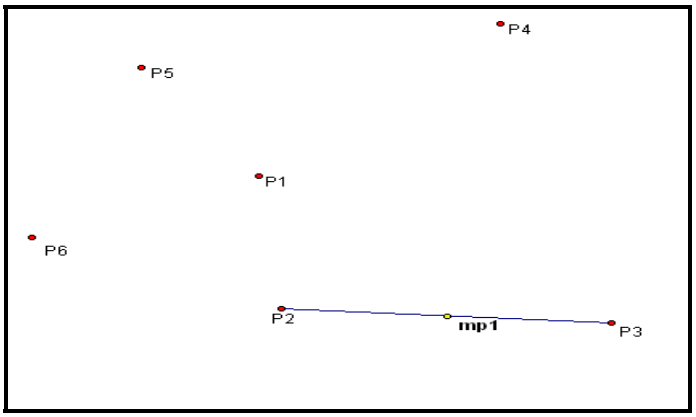


Figura II.4 Segmento de recta P_2P_3 y punto medio

A continuación se determinan también los segmentos de recta P_2P_1 y P_2P_6, estableciendo también los puntos medios de dichos segmentos mp_2 y mp_3 (figura II.5)

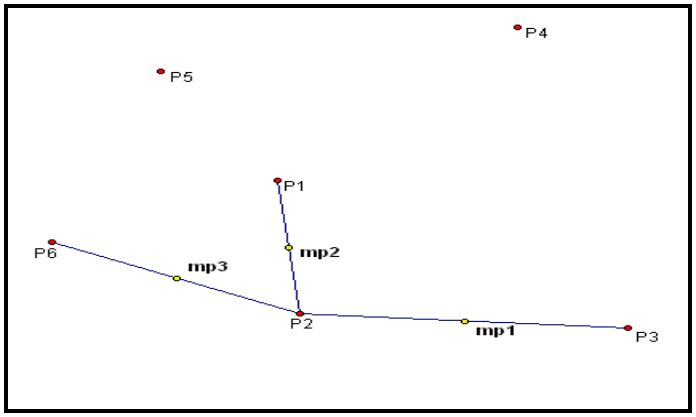


Figura II.5 Segmentos de recta y puntos medios

Con las rectas y los puntos medios determinados, se obtienen las rectas perpendiculares $perp_1$, $perp_2$ y $perp_3$ en el punto medio correspondiente a los tres segmentos de recta anteriormente dibujados, como se muestra a continuación.

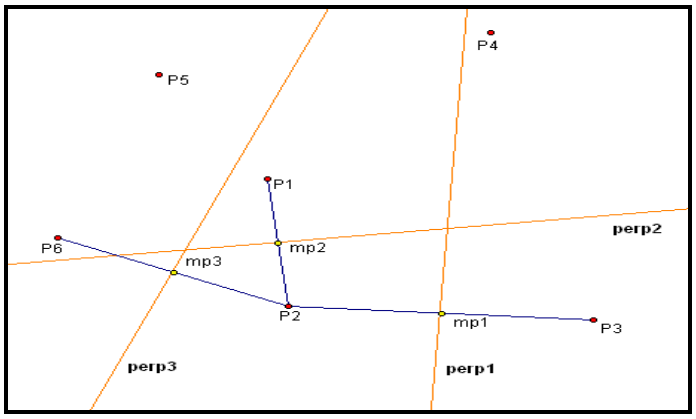


Figura II.6 Recta perpendiculares a las recta *mp* en los puntos medios

A partir de las tres rectas perpendiculares se observa que existe un polígono que rodea al punto P_2, eliminando los segmentos de recta sobrantes que no son parte del diagrama, se tiene delimitada la primera celda. (Figura II.7)

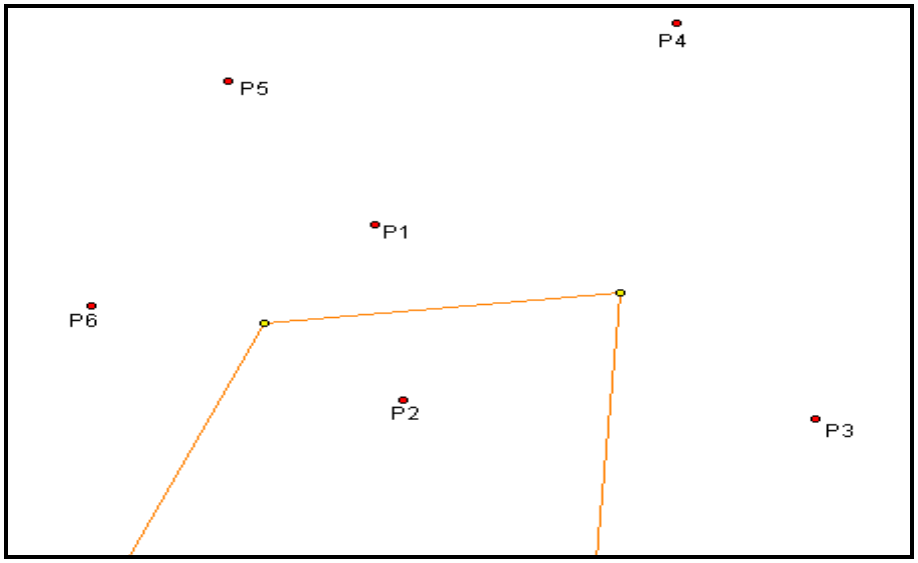


Figura II.7 Delimitación de la celda de Voronoi

Con la determinación de este polígono se termina con el trabajo en el punto P_2. Se pasa entonces con el punto P_6, siguiendo la metodología antes explicada, se determina la celda correspondiente a este punto, es decir se establecen los segmentos de recta P_6P_1 y P_6P_5 así como los puntos medios mp_4 *y* mp_5 correspondientes a cada segmento de recta (figura II.8), el segmento P_6P_2 no es necesario dado que ya se obtuvo el segmento de recta que separa a estos dos puntos.

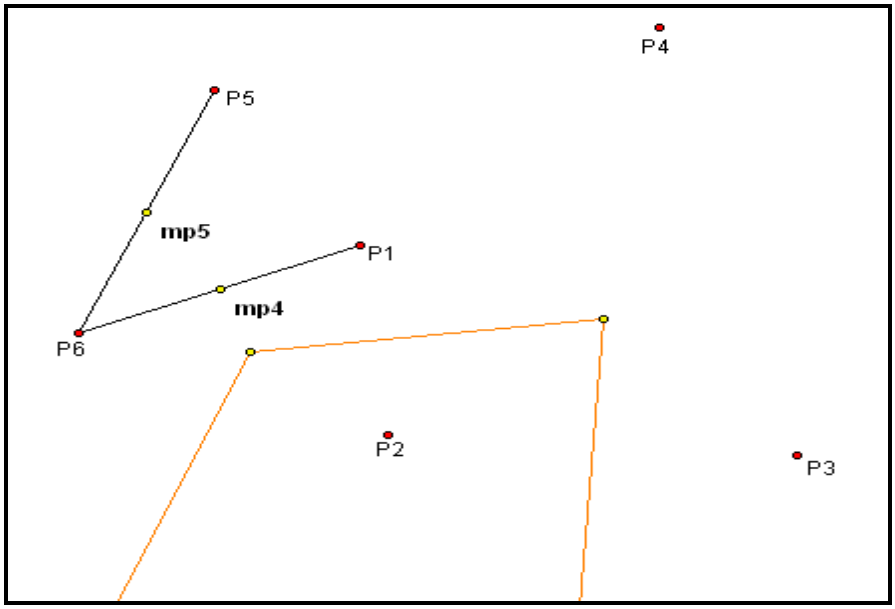


Figura II.8 Segmentos generadores de la celda de Voronoi

Siguiendo con el procedimiento antes explicado se obtienen las perpendiculares en el punto medio correspondiente, ocultando los elementos no necesarios en le diagrama es decir los segmentos de recta P_6P_1 y P_6P_5 así como los puntos medios mp_4 y mp_5.

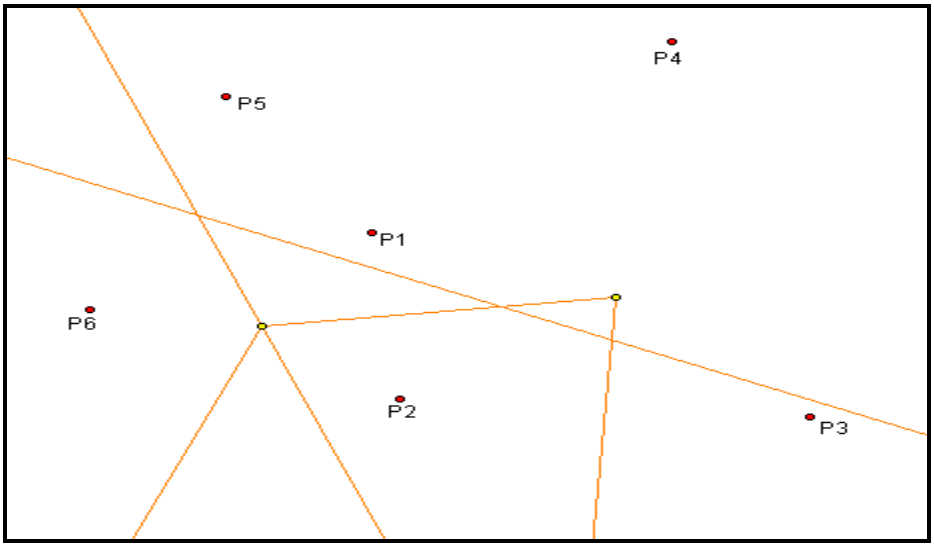


Figura II.9 Líneas perpendiculares a los segmentos *mp* en sus respectivos puntos medios

En la figura II.9 es posible observar nuevamente que con la generación de las rectas perpendiculares se forma un conjunto de rectas que rodean a P6, éstas forman la nueva celda de Voronoi, por lo que es necesario definir adecuadamente la celda correspondiente al nodo P6 eliminando los segmentos de recta sobrantes. (figura II.10)

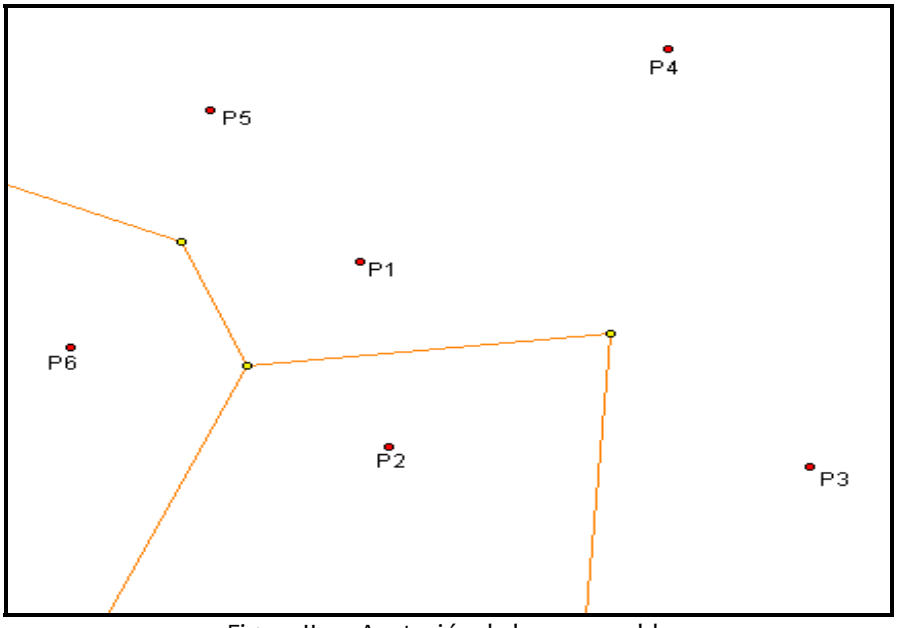


Figura II.10 Acotación de la nueva celda

Para los puntos P_5, P_4 y P_3 se sigue el mismo procedimiento, obteniendo segmentos de rectas con sus puntos vecinos, determinando puntos medios y las perpendiculares

correspondientes. De esta manera se obtiene el diagrama de Voronoi para el conjunto de puntos aleatorios.

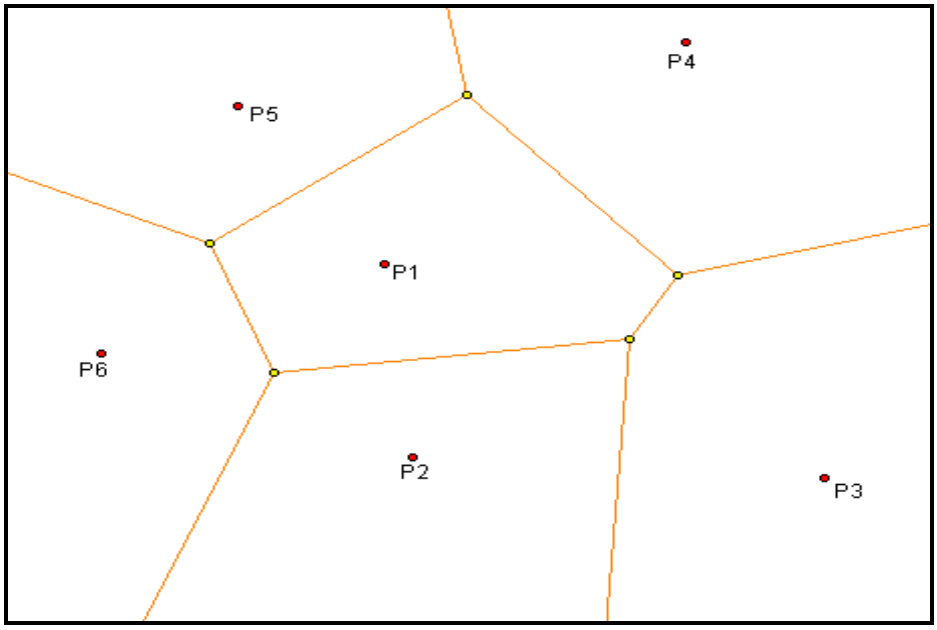


Figura II.1 Diagrama de Voronoi

Siguiendo con el procedimiento explicado se genera el diagrama de Voronoi para cualquier conjunto de puntos. En base a esta metodología es posible construir un algoritmo para implementar un programa capaz de generar dichas estructuras, sin embargo, por el tipo de procedimiento que sigue sería ineficiente y el tiempo de computo sería desperdiciado, por lo que existe una gran diversidad de programas y algoritmos que son capaces de generar los diagramas en forma eficiente dado que existe detrás de ellos trabajo de investigación arduo y de mucho tiempo en el que se a ampliado y desarrollado este tema, es por esto que el presente trabajo no pretende profundizar, solo dar los principios básicos.

Siguiendo la línea de presentar los principios básicos del tema se presentan a continuación las propiedades del diagrama de Voronoi.

II.3 Propiedades del diagrama de Voronoi

Se presentan a continuación las propiedades más representativas del diagrama de Voronoi. Es importante tener presente que estás propiedades y los temas desarrollados en este capitulo se especifican en el espacio bidimensional debido a la facilidad que presenta el plano para la comprensión del tema.

PROPIEDAD 1. Todas las regiones de Voronoi son convexas o infinitas.

PROPIEDAD 2. La unión de todas las regiones de Voronoi es el plano.

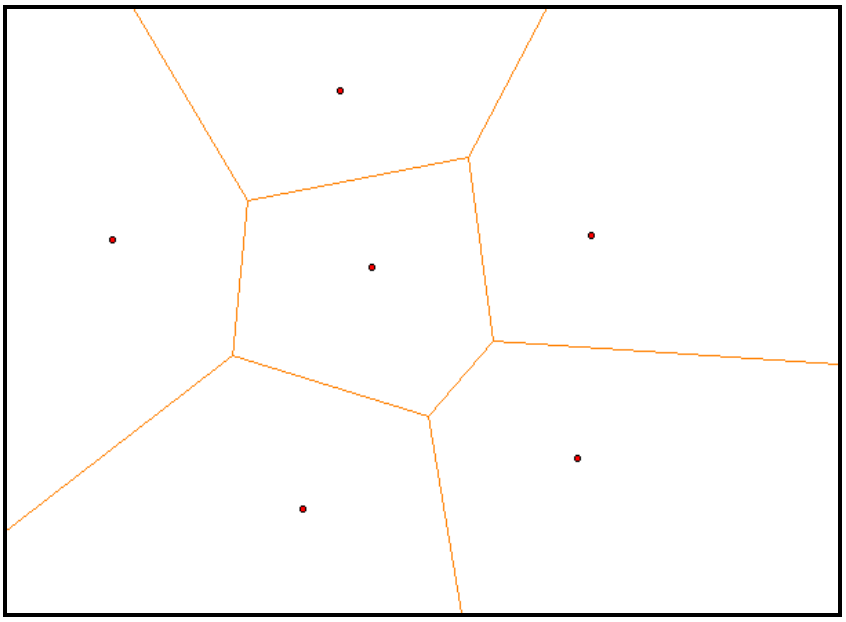


Figura III.12 Propiedad 1

PROPIEDAD 3. Si se toma como centro de un círculo cualquier punto del plano y se traza una circunferencia que pase por un solo nodo de Voronoi, dicho punto es interior a una región de Voronoi.

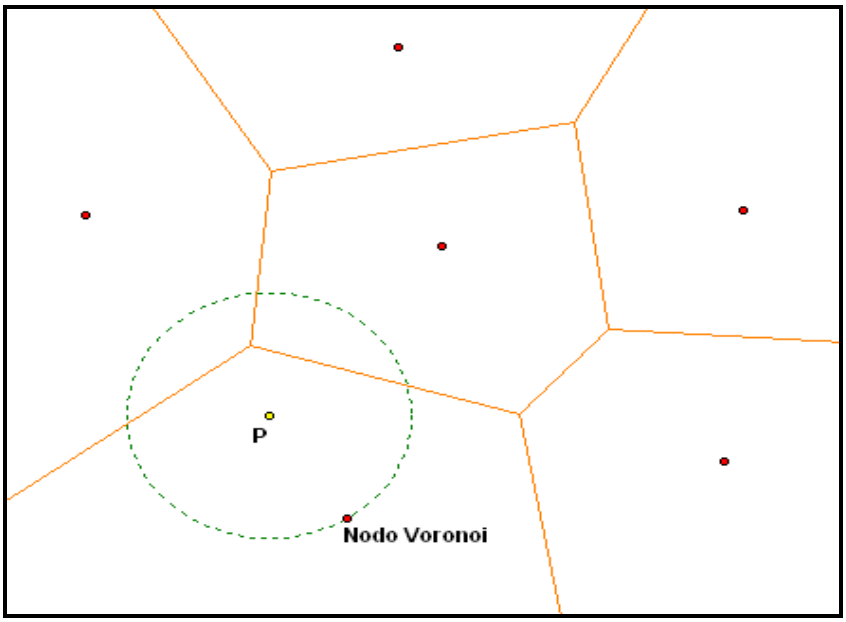


Figura II.13 Propiedad 3

PROPIEDAD 4. Si se toma como centro de un círculo cualquier punto del plano y se traza una circunferencia y está toca exactamente a dos nodos de Voronoi, separa exactamente a dos regiones de Voronoi, es decir, ese punto es parte de una arista de Voronoi.

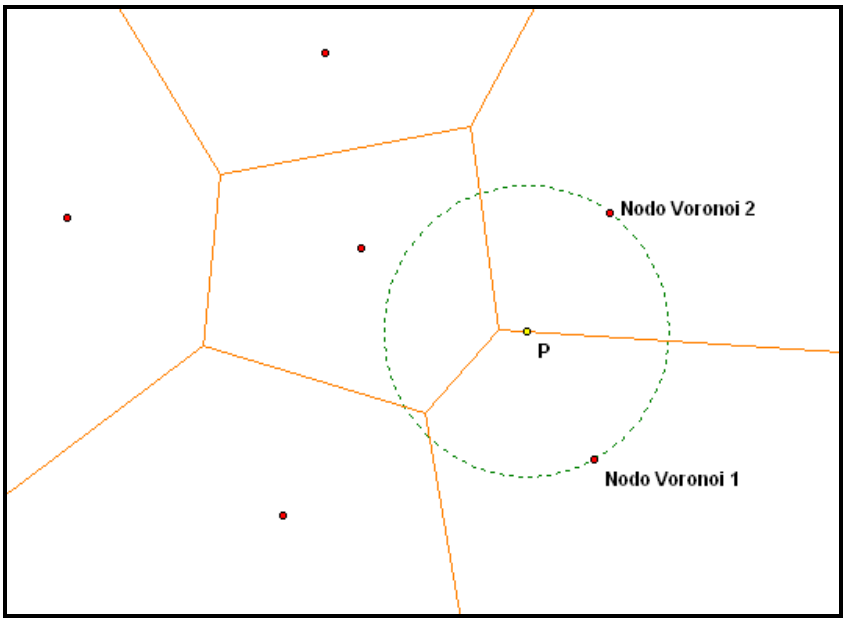


Figura II.14 Propiedad 4

PROPIEDAD 5. Si se puede dibujar una circunferencia que toque a tres nodos de Voronoi, es un vértice de Voronoi, esto es, la intersección de tres aristas de Voronoi.

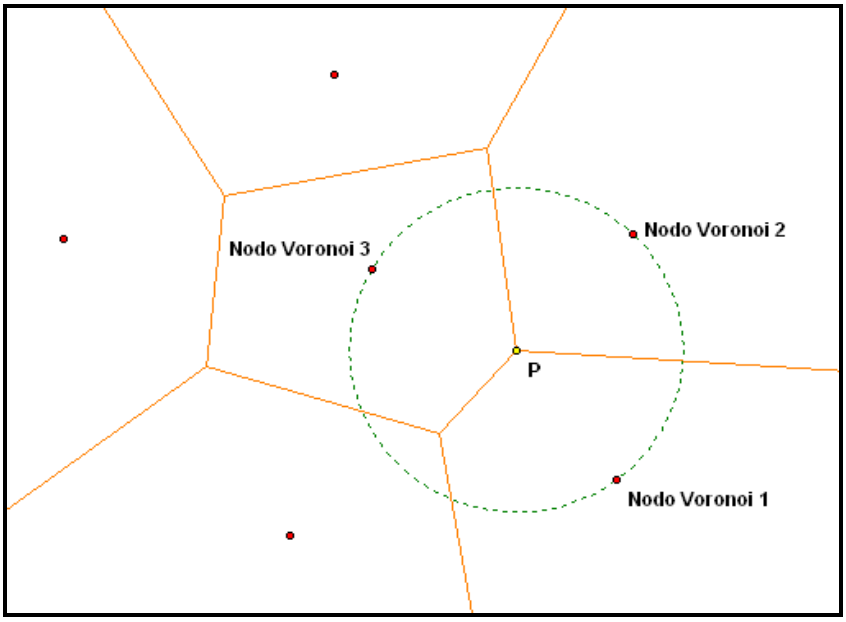


Figura II.15 Propiedad 5

PROPIEDAD 6. Para cada vértice de Voronoi, existe un único círculo centrado en dicho vértice y que pasa por tres nodos, dicho círculo es el mayor de todos los círculos que puede construirse sin que contenga dentro a otro, por lo tanto, si V es un vértice de Voronoi, P no es un nodo de Voronoi.

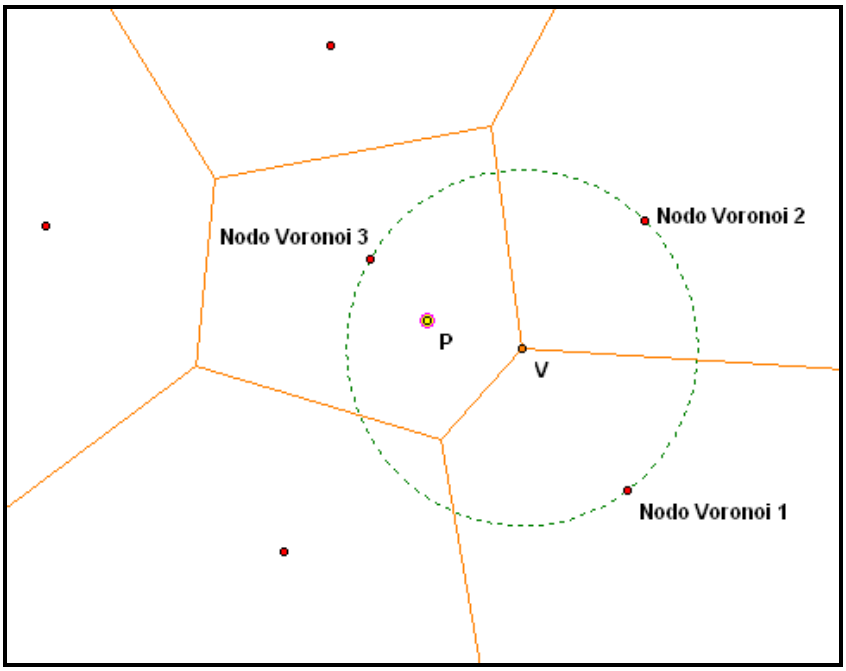


Figura II.16 Propiedad 6

III. CONSTRUCCIÓN DEL DIAGRAMA DE VORONOI 3D

El presente capítulo tiene por objetivo describir el programa desarrollado a lo largo de este trabajo, el cual es capaz de generar arreglos estructurales basados en el concepto de celdas de Voronoi, teniendo como datos de entrada una distribución de puntos y como datos de salida matrices que relacionan aristas y coordenadas de vértices de Voronoi, así como la estructura de un archivo *inp* necesario para llevar dichos arreglos estructurales al programa de análisis por elementos finitos ABAQUS®. La paquetería utilizada para el desarrollo del programa antes mencionado es MATLAB®, un programa matemático orientado al manejo de matrices y vectores con una gran variedad de funciones especializadas, de ahí la elección de este programa, ya que dentro de esta variedad se encuentra la función ***voronoin*** que determina los datos necesarios para graficar una estructura de Voronoi, sin embargo dichos datos no son los adecuados para poder exportarlos a un archivo *inp* de ABAQUS®, por lo que se requirió trabajar con estos para el formato del archivo *inp,* la modificación se realizó en un archivo *m* de MATLAB® que permite programar en un lenguaje sencillo basado en C.

Es importante recalcar que las funciones ***voronoin*** y ***convhulln*** utilizadas en el programa que se describirá a lo largo de este capítulo son funciones especializadas y creadas por MATLAB®, las cuales solo fueron aplicadas dada su gran funcionalidad y eficiencia, por otro lado el código generado para la manipulación de los datos es original creado especialmente para este trabajo.

A partir de anterior, el presente capítulo describirá el manejo proporcionado a las funciones ***voronoin*** y ***convhulln*** así como la adaptación dada a sus datos de salida, esto se presentará en los puntos III.1 y III.2 donde se describirá detalladamente las líneas de programación utilizadas con el fin de presentar las nociones básicas del manejo de la paquetería, ya que uno de los objetivos que se pretenden alcanzar es que en futuros trabajos el código pueda ser modificado, aún cuando no se conozca MATLAB®, para su mejora; sin embargo desde el punto III.3 ya no se hará una detallada presentación del código de programación ya que a partir de ese punto se manipula el diagrama de Voronoi y dicha manipulación es completamente susceptible de mejoras, por lo cual se presentan la lógica utilizada así como los datos de salida obtenidos. A pesar de que no se hará explícito el código, en el anexo se presentará éste, donde se indicarán las secciones que se explicarán a lo largo del capítulo.

III.1 Generación de los nodos o semillas de Voronoi

Como se mencionó en el capítulo dedicado al diagrama de Voronoi, el elemento fundamental para poder construir un diagrama de este tipo es la distribución de puntos (nodos) aleatoria u ordenada; debido a que MATLAB® maneja principalmente matrices y vectores, la función *voronoin* requiere esta distribución de puntos como una matriz de 3*x*n, donde n es el numero de puntos sobre los cuales se construirá el diagrama y cada renglón de la matriz son las coordenadas (*x, y, z)* de cada uno de estos puntos.

De lo anterior se define que el primer paso es la generación de los nodos o semillas de Voronoi, para los alcances de este trabajo la distribución se construyó de forma que los nodos siguieran el patrón de las estructuras cristalinas: cúbica simple, cúbica centrada en las caras y cúbica centrada en el cuerpo (figura III.1). Se escogen estas estructuras debido a que son los patrones más sencillos dentro de las estructuras cristalinas, en trabajos posteriores podría trabajarse con estructuras de mayor complejidad para observar las diferencias que presentan los distintos comportamientos.

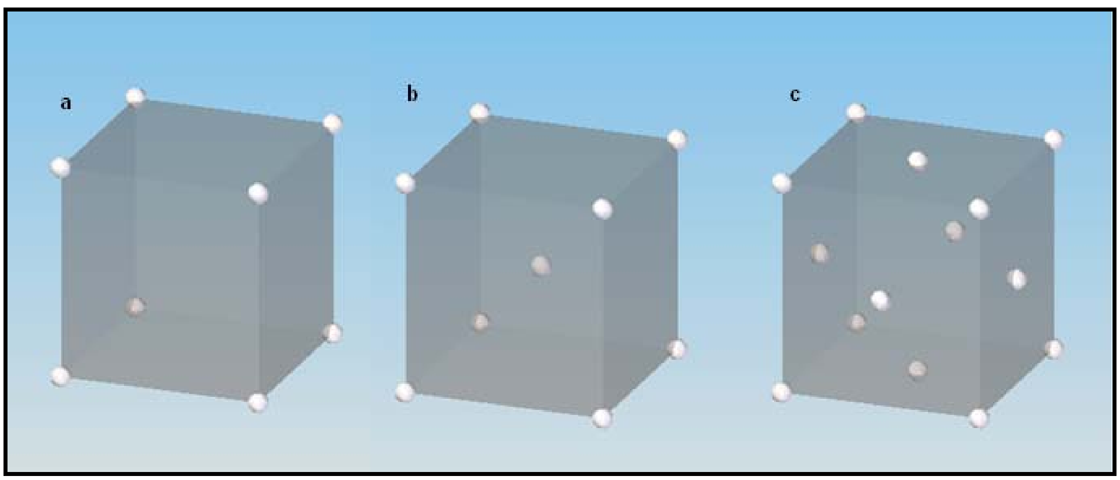


Figura III.1 Estructuras cristalinas: a) Cúbica simple b) Cúbica centrada en el cuerpo c) Cúbico centrado en las caras

Debido a que se generaron estos tres patrones, se tienen 3 diferentes secciones de generación de semillas de Voronoi en el programa. A continuación se presenta la sección de código dedicada a la generación de puntos con el patrón de la estructura cristalina cúbica simple.

```
%CÚBICA SIMPLE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
d= [0:1:5];
[x,y,z]=meshgrid (d,d,d);
B= [x(:),y(:),z(:)];
S= [B];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

La primera línea de programación define una matriz 6x1. En MATLAB® para poder definir una matriz se requiere encerrar a los elementos entre corchetes ([ ]) dentro de ellos las comas (,) o los espacios delimitan columnas y los punto y coma (;) renglones.

Ejemplo

A= [1, 2; 100 25] $A = \begin{pmatrix} 1 & 2 \\ 100 & 25 \end{pmatrix}$

Otra de las formas disponibles y es la utilizada en este caso dado el tipo de matriz, es definiendo dentro de los corchetes 3 números separados por dos puntos (:), el primer número define el primer elemento del renglón, el segundo el incremento entre los números y el tercero el ultimo elemento del renglón, si se quisiera definir otro renglón, al final del tercer número se coloca un punto y coma (;) definiendo de esta forma que se inicia un nuevo renglón, la única condición es que el numero de columnas sea compatible.

Ejemplo

A= [0:1:5; 0:2:10] $A = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 2 & 4 & 6 & 8 & 10 \end{pmatrix}$

La matriz *d* que se genera en esta sección, es la matriz base para generar la distribución de puntos.

d= [0:1:5] $d = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \end{pmatrix}$

La segunda línea del código, utiliza la función ***meshgrid***.

```
[X, Y, Z]=meshgrid (x, y, z)
```

Está función transforma el dominio especificado por los vectores *xyz* en los arreglos *XYZ*, generalmente utilizado para evaluar funciones de tres variables y generar gráficos de tres dimensiones, la aplicación que tiene para este trabajo es la de agrupar adecuadamente en las matrices *XYZ* los datos necesarios para generar un "*grid*" con una unidad de separación. Los renglones de la matriz *X* son copias del vector *x*, las columnas de la matriz de salida *Y* son copias del vector *y* y la matriz completa *Z* son copias del vector *z*. Para el caso de tres dimensiones, *meshgrid* genera 3 elementos que se conocen en MATLAB® como hipermatrices (figura III.2), es decir matrices de más de dos dimensiones que permiten almacenar con un único nombre distintas matrices del mismo tamaño.

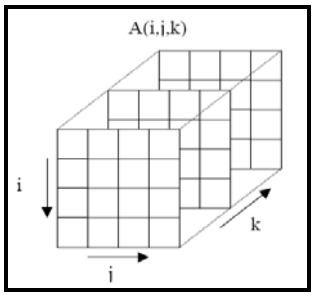


Figura III.2 Hipermatriz

Ejemplo

```
>> d= [0:1:1]
>> [x, y, z]= meshgrid (a, a, a)
x (:,:,1) =
     0     1
     0     1
x (:,:,2) =
     0     1
     0     1
y (:,:,1) =
     0     0
     1     1
y(:,:,2) =
     0     0
     1     1
z (:,:,1) =
     0     0
     0     0
z (:,:,2) =
     1     1
     1     1
```

La tercera línea del código, es la definición de una nueva matriz.

```
B= [x (:),y (:),z (:)]
```

La función que tiene la matriz B es la de agrupar en un solo arreglo los datos que la función ***meshgrid*** ordeno en tres matrices, de tal forma que cada renglón de la matriz B sean las coordenadas (*x, y, z)* de los puntos que se pretenden muestrear. Antes de definir a la matriz *B* se definen sus elementos, en esté caso los tres vectores columna ***x (:),y (:)*** *y* ***z (:)*** estos se obtuvieron a partir de las hipermatrices ***x y z*** de la siguiente forma.

La hipermatriz x está definida de la siguiente forma.

```
>> x
x(:,:,1) =

     0     1
     0     1
x(:,:,2) =
     0     1
     0     1
```

Para transformarla en un vector columna se declara lo siguiente.

```
>> x(:)
ans =
     0
     0
     1
     1
     0
     0
     1
     1
```

En este caso el operador "dos puntos" es utilizado para obtener el vector columna, al ordenarlas una detrás de otra. De la misma forma se hace para las hipermatrices **y** y **z**. Teniendo los vectores definidos, se separan por comas y se encierran entre corchetes con lo que se define una matriz.

```
>> B= [x(:),y(:),z(:)]
B =
     0     0     0
     0     1     0
     1     0     0
     1     1     0
     0     0     1
     0     1     1
     1     0     1
     1     1     1
```

Con esto se construyó un muestreo de puntos con el patrón cúbico simple, es decir solo los vértices de un cubo, para efectos del trabajo la distribución de puntos se creó en un volumen de 5*5*5 unidades. La representación grafica de esta distribución se muestra en la figura III.3.

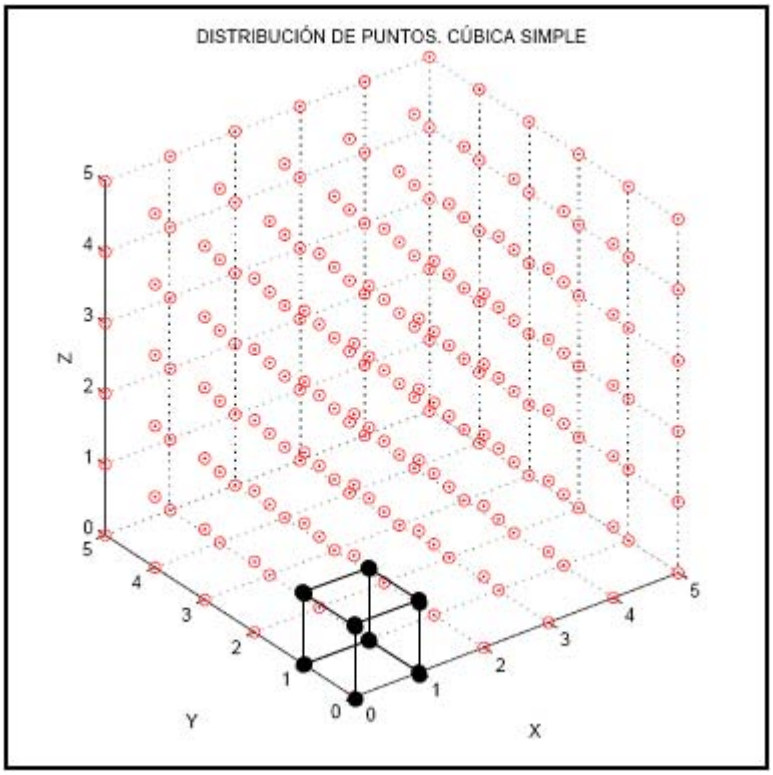


Figura III.3 Distribución de puntos con el patrón de la estructura cristalina cúbica simple

La línea cuatro de programación es solo asignar un nuevo nombre a la matriz B.

```
S= [B]
```

Está distribución es la base para generar los dos conjuntos con los que se pretende trabajar, es decir los patrones de las estructuras cúbica centrada en el cuerpo y cúbica centrada en las caras. Se comenzara por describir la distribución cúbica centrada en el cuerpo cuyo código se presenta a continuación.

```
%CÚBICA CENTRADO EN EL CUERPO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   1. d=[0:1:5];
   2. [x,y,z]=meshgrid(d,d,d);
   3. B= [x(:),y(:),z(:)];
   4. Y=B+0.5;
   5. S=[B; Y];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Como se puede observar, las líneas de programación que generaron esta distribución siguen los comandos que se utilizaron para la cúbica simple, únicamente se agregó la definición de dos matrices, una de ellas es la matriz Y con el fin de crear los nodos que se encuentran en el centro de las estructuras.

```
Y= B + 0.5
```

Esta línea indica que a todos los elementos de la matriz B (los puntos que siguen el patrón de la estructura cúbica simple) se le suman 0.5 con lo cual todos ellos se mueven en las direcciones *xyz* punto cinco de unidad, creando de esta forma los nodos centrales de cada cubo.

La línea cinco utiliza el operador punto y coma (;) para concatenar las matrices Y y B con lo cual se reúnen en una sola matriz todas las coordenadas de los puntos que siguen la estructura cúbica centrada en el cuerpo.

Con la manipulación de estos comandos se generó la distribución cúbica centrada en el cuerpo, cuya representación grafica se muestra en la figura III.4

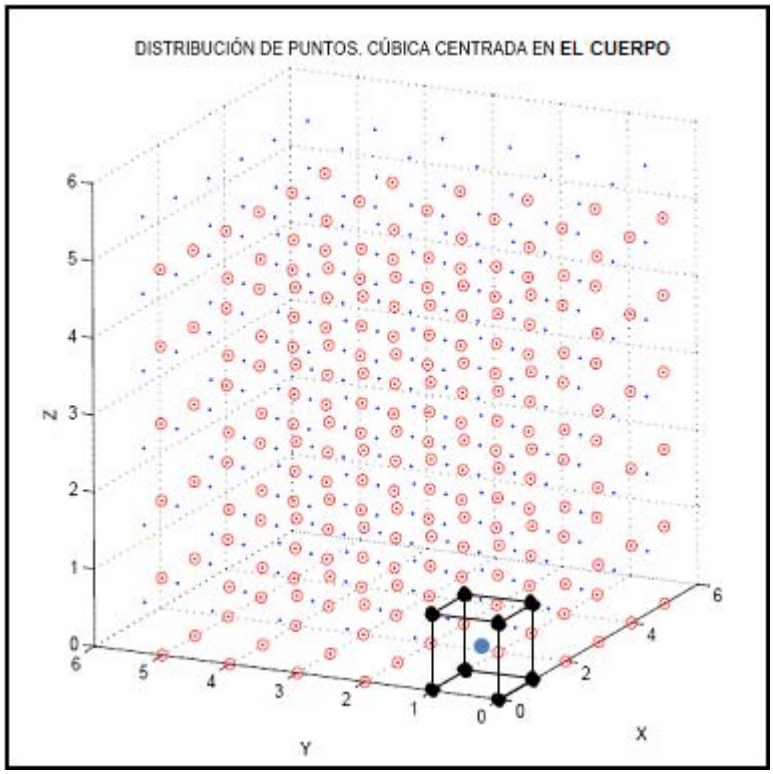


Figura III.4 Distribución de puntos con el patrón de la estructura cristalina cúbica centrada en el cuerpo.

Por último, la sección de generación de semillas para la distribución cúbica centrada en las caras, se construyó a partir de las siguientes líneas de programación; cabe mencionar que no se muestra el código completo debido al espacio que ocuparía, sin embargo se describen las partes de mayor importancia. El código que construye esta distribución de puntos, tiene como base la distribución cúbica simple, es por ello que aparecen los comandos importantes en las líneas de código de esa distribución, como lo es el ***meshgrid**, el operador punto y coma*, *el operador coma*, etc. definidos en las primeras tres líneas de programación de esta sección.

```
%CÚBICA CENTRADA EN LAS CARAS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
d= [0:1:5];
[x,y,z]= meshgrid (d,d,d);
B= [x(:),y(:),z(:)];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Para completar y seguir el patrón de estructura cúbica centrada en las caras se tiene, además de la base, tres partes importantes, en conjunto dedicadas a la construcción de las matrices *alphaXY, alphaYZ, alphaXZ.* La función de dichas matrices es generar las coordenadas de los nodos que se encuentran en las caras de las estructuras. La forma en que se generaron estos nodos, fue dividiéndolos dependiendo de la cara en la que se localicen, de esta forma se tienen los nodos que se encuentran en las cara paralelas al plano XY, los que se localizan en las caras paralelas al plano YZ y finalmente los nodos de las caras paralelas al plano XZ, los códigos para generar los 3 grupos de puntos siguen la misma lógica, a partir de esto se explica a continuación la generación del grupo XZ para la descripción de la lógica utilizada.

El código esta formado básicamente por tres *ciclos "for",* cada uno con la función de generar coordenadas de puntos. En MATLAB® al igual que en un lenguaje de programación imperativo el *ciclo for* es utilizado en programas donde se requieren procesos iterativos, la forma de definir un ciclo for en el archivo *m* es la siguiente:

```
for i=0:1:5
   cuerpo
    end
```

Los elementos que forman la estructura del ciclo son: la variable de control (i), la inicialización de la variable de control, el incremento de la variable en cada iteración, el valor final de la variable (estos tres últimos elementos separados por el operador dos puntos (:), en caso de no definir el incremento de la variable se toma por default 1), el cuerpo del ciclo (la instrucción que se hará en cada iteración) y el fin del ciclo que se declara con la palabra *end.*

```
%PLANO XZ
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cont4=1;
marca10=5; %SE DEFINE EL ANCHO DE LA CELDA
marca11= marca10-0.5;
for hh=0:1:marca10
    for gg=0.5:1:marca11
        for ff=0.5:1:marca11
            alphaXZ (cont4,:)= [gg,hh,ff];
            cont4=cont4+1;
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

El código anterior muestra que las variables de control de los ciclos "*for*" son por si mismas las coordenadas *(x, y, z)* dentro de la matriz alphaXZ. Las variables *gg (x)* y *ff (z)* tienen la característica de generar puntos a partir del valor de 0.5 en ***x*** y ***z*** con una distancia entre ellos de una unidad, lo que da por resultado un conjunto de puntos en el plano XZ representando de está forma los núcleos en las caras paralelas a dicho plano. Para generar esta distribución de puntos en todas las caras, la variable hh (y) se mueve una unidad a partir de cero, lo que da por resultado repetir la distribución de puntos que se genero en el plano XZ, la representación grafica de esto se muestra en la figura III.5

Dentro de las líneas de código, la definición de la matriz dentro de los ciclos "*for*" es un aspecto que vale la pena describir.

```
alphaXZ (cont4,:)= [gg,hh,ff];
cont4=cont4+1;
```

En estás líneas, como se mencionó anteriormente se define una matriz, la diferencia con esta definición es que después del nombre de la matriz se coloca un paréntesis dentro del cual se indica el número de renglón y columna al que pertenece el o los elementos de la matriz que se definen entre corchetes.

```
Matriz_A (renglón, columna) = [elemento]
```

El operador dos puntos (:) dentro de la declaración de una matriz define, dependiendo de la posición de esté, lo siguiente.

```
A (:,j) La j-ésima columna de A
A (i,:) El i-ésimo renglón de A
```

La propiedad de esté operador permitió la definición uno a uno de los puntos en cada iteración de los ciclos "*for*", variando el renglón al que pertenecía el elemento que se declaraba.

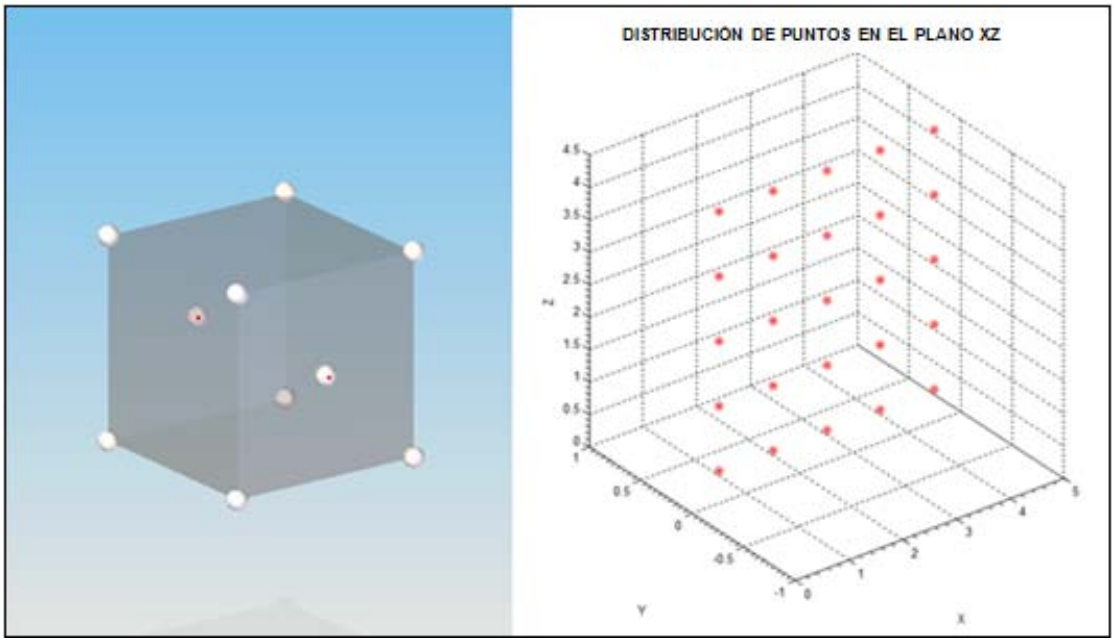


Figura III.5 Distribución de puntos en el plano paralelo a XZ en y=0.

Para la creación de las matrices *alphaXY* y *alphaYZ* se usan los mismos ciclos, la diferencia es la posición de las variables de control (*gg, hh y ff*) en la definición de la matriz, la variación de la posición de dichas variables es con el objeto de colocar la distribución de puntos como muestra en la figura III.5 paralela al plano que le corresponde, con lo cual se generan todos los puntos que caracterizan a la estructura cristalina cúbica centrada en las caras. La figura III.6 muestra la distribución de puntos completa.

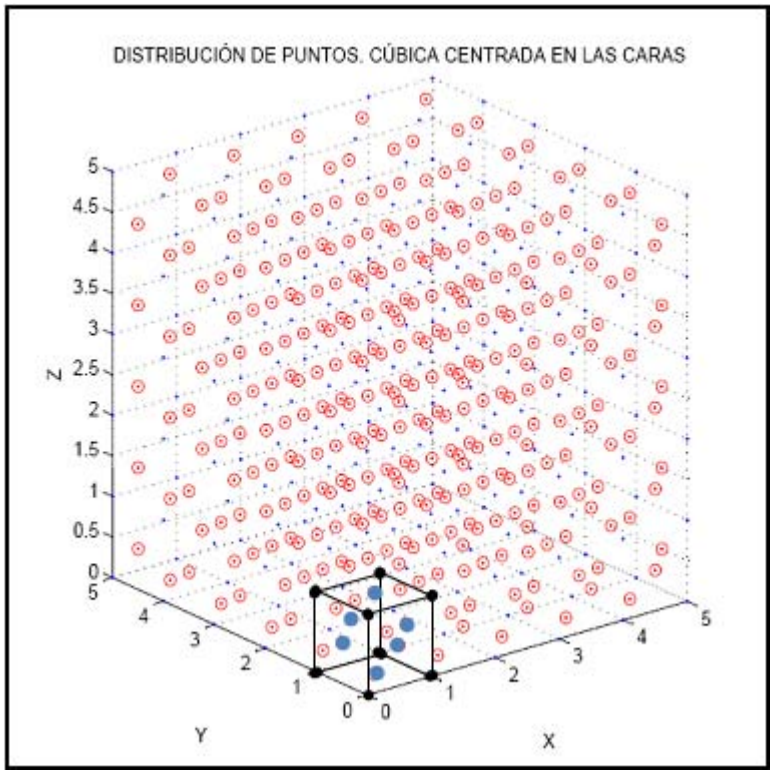


Figura III.6 Distribución de puntos con el patrón de la estructura cristalina cúbica centrada en las caras.

Con este conjunto de puntos se tienen ya tres tipos de distribuciones, a partir de las cuales es posible generar tres distintos arreglos de Voronoi, este conjunto de arreglos diferentes es pequeño, por ello, con el fin de aumentar este número y observar como es el comportamiento de las estructuras a partir de variar las características geométricas de dichos arreglos, se distorsionar las bases, es decir la base cúbico simple, cúbico centrado en las caras y cúbico centrado en el cuerpo, para obtener un número mayor de muestras y observar su comportamiento.

La perturbación de las distribuciones de los puntos base se realiza sumando un número aleatorio a las coordenadas *(x,y,z)* de cada uno de los puntos que forma dicha distribución. Como se mencionó anteriormente las distribuciones están definidas con matrices, por ello, para realizar la perturbación se crea la *matriz D.* Como se muestra a continuación.

```
%MATRIZ D "DISTURBIO" %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for nn=1:1:size(S,1)
     D(nn,:)=[random('norm',0,sigx),random('norm',0,sigy),random('norm',0,sigz)];
end
F=S+D;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Dicha *matriz D,* debe tener el mismo número de renglones que la *matriz S* (distribución de puntos) para realizar la suma, para lo cual, estos quedan en dependencia del ciclo *"for"*, de forma que al colocar a la variable de control del ciclo dentro del paréntesis (D (*nn*, :)) ésta controla el número de renglones que se crean para la *matriz D,* por lo tanto el valor final de la variable de control debe ser el número de renglones de la *matriz S,* este limite se define usando el comando **size** el cual indica número de renglones o columnas a partir de lo siguiente.

```
size (A, 1) Número de renglones de la matriz A
size (A, 2) Número de columnas de la matriz A
```

A partir de esto, el valor final del ciclo es size(S, 1) con esto se tiene a la *matriz D* en cuanto al número de renglones, adecuada para la suma. Otra característica que debe tener está matriz es que sus elementos sean números aleatorios, para satisfacer esta necesidad se utiliza el comando ***random,*** éste genera números aleatorios de una determinada distribución estadística, los parámetros de dicho comando son:

```
random (nombre de la distribución, Al, A2, A3,[m,n,...])
```

Donde A1, A2 y A3 son parámetros de la distribución, los corchetes dentro de la declaración indican si los números aleatorios se van a dar como un arreglo de *m x n x…* *si se omite* esté parámetro se toma por default una matriz de uno por uno. Como se observa en las líneas de programación de está sección, cada elemento de la *matriz D* usa este comando para determinar el número aleatorio, con objeto de que las matrices sean compatibles para la suma, el numero de columnas que se definen son tres. En cuanto al nombre de la distribución, cabe mencionar que no se escribe explícitamente el nombre, cada distribución utiliza una palabra clave y los parámetros A1, A2 y A3 dependen de cada distribución; para fines de esté trabajo se utilizaron números aleatorios de una distribución normal, los parámetros para esta distribución en específico son:

```
random('norm', media, desviación_estandar)
```

Teniendo la *matriz D* adecuada para perturbar los puntos se suma con la matriz de puntos.

```
F=S+D;
```

Está notación solo se puede utilizar si el número de columnas y renglones de las matrices S y D son iguales. Con esto se tiene la *matriz F* que representa a la *matriz S* después de la perturbación, esta matriz F permite a partir de una base, obtener un número mayor de modelos, los cuales pueden ser caracterizados por la desviación estándar y la media de la distribución que determina los números aleatorios.

Además de la perturbación, una transformación geométrica que se toma en cuenta en el programa es la distorsión de los arreglos de Voronoi, definir está transformación dentro de la líneas de programación se declara de la siguiente forma.

```
%MATRIZ DE TRANSFORMACIÓN T
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
T= [ax 0 0; 0 ay 0; 0 0 az];
A=F*T;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Se observa que T se define como una matriz de 3x3 donde ***ax, ay*** y ***az*** son los factores de escalonamiento que se aplican sobre las coordenadas (*x, y, z*) de la *matriz F, de la siguiente forma:*

$$\begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{mn} \end{pmatrix} \begin{pmatrix} a_x & 0 & 0 \\ 0 & a_y & 0 \\ 0 & 0 & a_z \end{pmatrix} = \begin{pmatrix} a_x*a_{11} & a_y*a_{12} & a_z*a_{1n} \\ \vdots & \ddots & \vdots \\ a_x*a_{m1} & a_y*a_{m2} & a_z*a_{mn} \end{pmatrix}$$

$$F \quad * \quad T \quad = \quad A$$

Para este trabajo los factores de escalonamiento se mantienen en una unidad, dedicándose únicamente al análisis de las estructuras antes y después de aplicarles una perturbación.

En este momento se tiene ya la base para generar la distribución de puntos aplicándole una perturbación y una distorsión. Con esto termina la sección dedicada a la generación de semillas de Voronoi, en la sección siguiente se describe la generación de las celdas de Voronoi a partir de un conjunto de puntos.

III.2 Generación de la estructura de Voronoi

La generación de la estructura de Voronoi, tiene como base al comando ***voronoin*** de MATLAB®, el cual crea la estructura en tres dimensiones a partir de un conjunto de puntos. Los datos de entrada para está función son las coordenadas (*x, y, z*) de los puntos, éstos definidos en una *matriz X* de nx3 donde n es el numero de puntos y tres las dimensiones en que se trabaja, a partir de esta matriz, ***voronoin*** devuelve los vértices *V* y las celdas *C*. *V* es una matriz de *numvx*3 donde *numv* es el número de vértices en el espacio. *C* es una matriz de vectores donde cada vector tiene como elementos a los índices de *V*, los cuales indican que vértices forman una celda.

En el ejemplo siguiente se presenta el comando ***voronoin*** aplicado a un conjunto de puntos aleatorios (*matriz X*) con el objeto de visualizar el formato en la declaración de la función, así como la presentación que MATLAB® le da a los datos de salida.

```
>>X  %Matriz del conjunto de puntos
     % X          Y         Z
   -0.0015   -0.0064   -0.0084
   -0.0070    0.0082   -0.0057
    0.0038   -0.0066   -0.0037
    0.0086    0.0034   -0.0070
>> [V,C] = voronoin(X) %Aplicación de la función voronoin a la matriz X

  V  Matriz de coordenadas de los vértices del arreglo Voronoi
     % X          Y         Z
        Inf        Inf       Inf
   -0.0021   -0.0081    0.0026
   -0.0222   -0.1104    0.0169
   -0.0041   -0.0076   -0.0002
   -0.0087   -0.0013   -0.0042
  C  %Matriz de vectores
    [1x6   double] % Vector renglón de seis columnas
    [1x6   double]
    [1x12  double]
```

En el ejemplo se observa que el primer dato de salida es la *matriz V,* cada renglón representa las coordenadas (x, y, z) de los vértices, esta matriz tiene siempre como primer elemento las coordenadas de un punto en el infínito el cual sirve como referencia para determinar si una celda es abierta o cerrada. Como segundo elemento de salida se tiene a la *matriz de vectores C* donde cada renglón de este arreglo es un vector que contiene los índices de la *matriz V,* MATLAB® lo presenta en forma simplificada indicado solamente las dimensiones del vector. En el ejemplo, el primer término indica que es un vector renglón de 6 elementos ([1x6 double]) cada uno de doble precisión (tipo de valor numérico definido en MATLAB®).

Para poder visualizar cada vector de forma explicita se utiliza el operador llaves { } y dos puntos (:) de la siguiente forma.

```
>> C{:}
%C{1} ans = 1   3   4   5   12   13
%C{2} ans = 1   5   6   7   11   13
%C{3} ans = 2   4   5   6   7   9   10   11   12   13   14   15
```

Con esta declaración se presentan explícitamente los vectores que contiene *C*, cada vector es un conjunto de vértices que forman una celda de Voronoi, la forma de dar estos vértices en los vectores es indicando solamente los índices o número de renglón de la *matriz V,* es una forma de simplificar para no poner las coordenadas. A continuación se presenta de forma explicita este hecho.

Sea *C {3} un vector* de C:

```
C {3} =2    4    5    6    7    9   10   11   12   13   14   15
```

El primer elemento es el número 2 por lo tanto se refiere al segundo elemento de *V* que es el que se muestra.

```
V (2, :)= -0.0070    0.0082   -0.0057
```

De igual forma se sigue para todos los elementos y vectores de C.

Como ocurre en dos dimensiones, en un diagrama de Voronoi existen celdas abiertas y cerradas, la forma de diferenciarlas a partir de los datos de salida, V y C, es observando los elemento de los vectores de C, si algún vector de C tiene como primer elemento el uno se trata de una celda abierta ya que se tendría un vértice en un punto del infinito, mientras esto no ocurra se trata de una celda cerrada. Esta diferenciación es útil debido a que la aplicación que se le da al concepto es la de representar a las trabeculas del hueso como las aristas de un diagrama de Voronoi, por ello una celda abierta no tiene sentido, ya que una celda de este tipo tiene aristas de longitud infinita lo cual no sucede en las trabeculas del hueso, por esta razón solo se trabajará sobre las celdas cerradas.

En este punto se tienen las coordenadas de los vértices, el conjunto de puntos que forman una celda así como la diferencia entre celdas cerradas y abiertas, sin embargo se requiere conocer también como esta dada la conexión entre los vértices, es decir determinar las aristas del arreglo, para lo cual se aplica el concepto de envolvente convexa[2] de un conjunto

[2] La **envolvente convexa** de un conjunto de puntos P es el menor conjunto convexo que tiene a dichos puntos (es decir, la intersección de todos los conjuntos convexos en los que P esta contenido).

de puntos, la razón de aplicar dicho concepto es que al obtener la envolvente convexa del conjunto de vértices de Voronoi que forman una celda cerrada se obtiene la celda misma (Figura III.7).

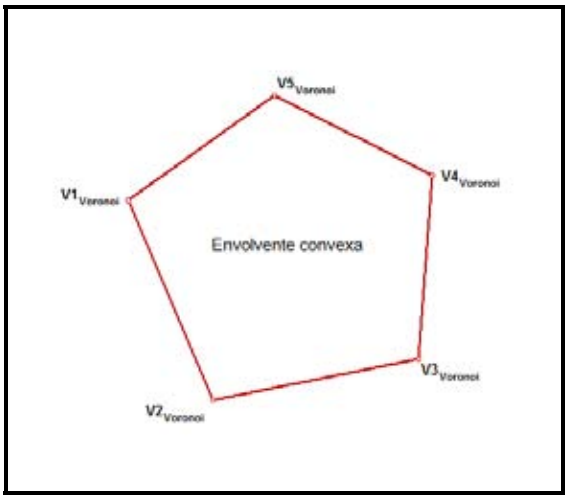


Figura III.7 Envolvente convexa de un conjunto de vértices que forman una celda cerrada de Voronoi

Llevando esta idea a MATLAB®, se trabaja con el comando ***convhulln*** el cual determina la envolvente convexa de un conjunto de puntos, el dato de entrada para este comando es el conjunto de puntos dados como una *matriz A* de mxn donde m es el numero de puntos y n es el espacio en el cual se trabaja, en este caso es una matriz de mx3, el dato de salida es la *matriz k* de pxn donde p es el numero de caras de la envolvente y n la dimensión en la que trabaja, esta matriz tiene como elementos los índices o número de renglones de A que forman parte o una cara completa de la envolvente convexa.

En el ejemplo siguiente se presenta el comando ***convhulln*** aplicado a un conjunto de puntos aleatorios (*matriz A*), con el objeto de visualizar el formato en la declaración de la función, así como la presentación que MATLAB® le da a los datos de salida.

```
>>A % Matriz de coordenadas del conjunto de puntos
    %  x          y          z
   -0.0023   -0.0079   -0.0016
   -0.0034   -0.0082    0.0012
    0.0012    0.0016    0.0150
    0.0070    0.0041    0.0031
   -0.0052   -0.0051    0.0020
>> K=convhulln (A) % Se aplica el convhulln a la matriz A
K % Matriz de vértices de la envolvente convexa
  % Números de renglón de la matriz A
     1     4     7
     5     1     4
     4     2     5
```

Como puede observarse, la *matriz A* es una matriz de coordenadas mientras que la *matriz K* es una matriz de índices referidos a la *matriz A,* la misma relación de la matriz de vectores C y la *matriz V*, por lo cual se tiene en cada renglón de K las coordenadas de 3 puntos

que son los vértices de triángulos que forman parte de una cara o son una cara completa de la envolvente convexa. Gráficamente K representa la triangulación que existe en las caras de la envolvente convexa (Figura III.8).

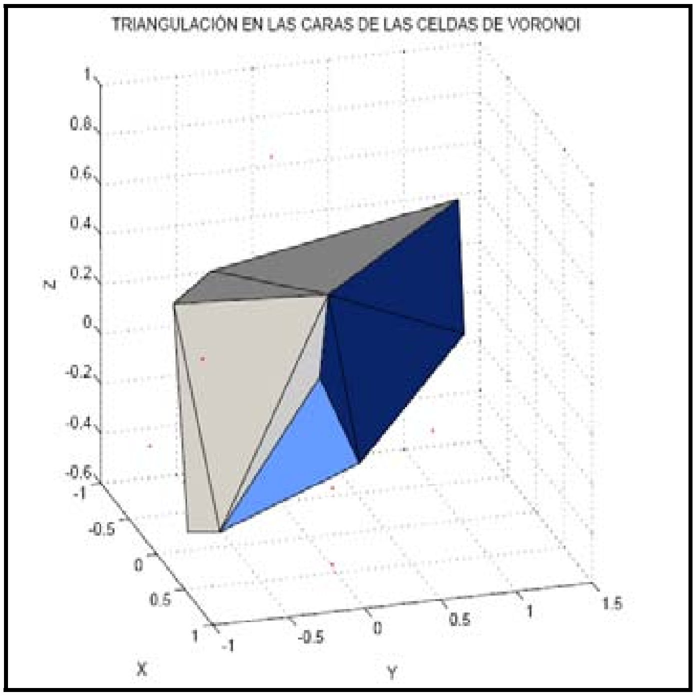


Figura III.8 Triangulación en las caras de la celda. Los triángulos con el mismo color pertenecen a una cara de la celda de Voronoi

Con esta herramienta se pueden ya definir las celdas cerradas de Voronoi, la forma de trabajar con ambas funciones se ejemplifica a continuación, donde se determina la celda de Voronoi de un conjunto de puntos aleatorio aplicando las funciones ***voronoin*** y ***convhulln***.

```
%Celda Unitaria de Voronoi
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
d=[-1 1];
[x,y,z]=meshgrid (d,d,d);

      X=[x(:),y(:),z(:)];
      X(9,:)=[0 0 0];
      X=X.*(0.01*rand(size(X)));
            [V,C]=voronoin(X);
            Z=V(C{9},:); %Celda cerrada
            K=convhulln(Z);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Esta sección maneja en sus primeros cuatro renglones las funciones que se utilizaron para calcular las distribuciones de puntos, el quinto renglón tiene la función de distorsionar los puntos determinados en la *matriz X* multiplicando dicha matriz por un numero aleatorio, dicho número se obtiene a partir de la función ***rand*** la cual genera de manera uniforme número pseudoaleatorios, el operador paréntesis que aparece después de la función se utiliza para indicar las dimensiones de la matriz de números pseudoaleatorios, en este caso se utiliza las dimensiones de la *matriz X*.

```
X=X.*(rand(size(X)))
```

Se observa también que se utiliza el operador punto (.) antes de declarar la multiplicación, esto debido a que X es una matriz de 9x3 al igual que el multiplicador, por lo tanto no es posible realizar la operación, para resolver esto se antepone el punto a la multiplicación con lo cual se indica que se realizará una multiplicación de elemento por elemento, con esto se crea una matriz de numeros pseudoaleatorios. Con X se tiene ya el conjunto de puntos a partir de los cuales se obtendrá el diagrama de Voronoi, por lo cual se le aplica la función ***voronoin***.

```
[V,C]=voronoin(X)
```

Como se describió, la función devolverá los vértices V y la relación de vértices C, dado que no tienen sentido para este trabajo las celdas abiertas, se descartan todos los vectores de C que representen esté tipo de celdas, analizando cada elemento de C se determina que el único elemento que representa una celda cerrada es el vector 9, por lo cual se obtiene la envolvente convexa de esta celda.

```
Z=V(C{9},:); %Celda cerrada
K=convhulln(Z);
```

Con esto se tiene una celda cerrada, definida por triángulos dados sus vértices en la *matriz K*. Gráficamente se tiene la celda representada en la figura III.9 El ejemplo presento únicamente una celda cerrada, sin embargo si la distribución de puntos es más compleja y numerosa el arreglo de Voronoi igualmente se complica y el número de celdas cerradas aumenta, con lo cual el proceso de determinar cual es una celda cerrada o abierta debe de ser a través de un ciclo iterativo, esto es lo que se realizó en el trabajo ya que el conjunto de puntos fue mucho mayor al igual que su complejidad, a pesar de trabajar con distribuciones diferentes, a partir de este punto las líneas de comando para generar el arreglo de Voronoi son iguales independientemente del tipo de distribución que se trabaje.

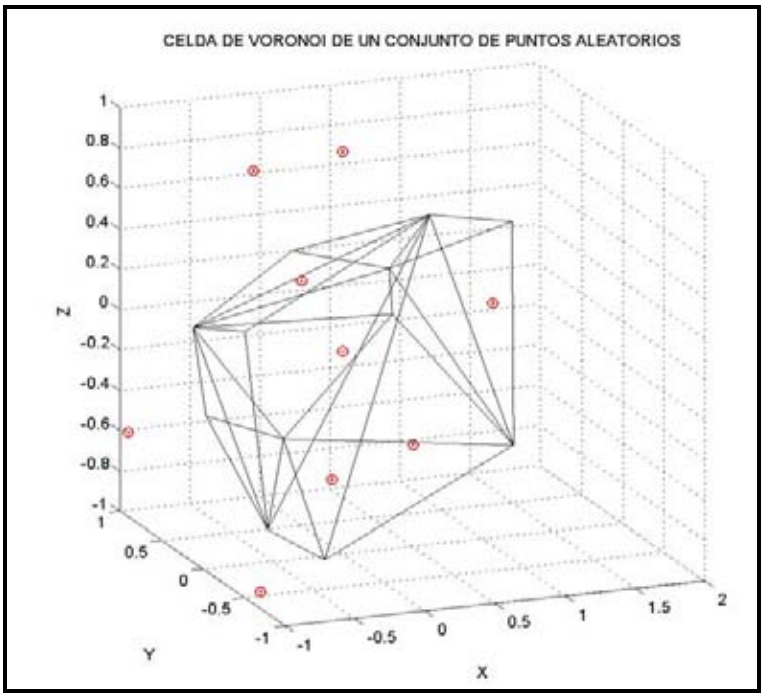


Figura III.9 Celda unitaria de Voronoi

Con esta premisa se presenta a continuación la parte del programa que genera las celdas del arreglo de Voronoi, de un conjunto cualquiera A de puntos.

```
%MATRIZ K DE VORONOI
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[V,C]=voronoin(A);
banda=0;
for a=1:size(C,1)
    X=V(C{a},:)
    if(C{a}(1)~=1)
        Z=X
        K=convhulln(Z);
        matrizZ(banda).matrix1=Z
        matrizK(banda).matrix1=K
        banda=banda+1
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Esta sección del programa presenta en la primera línea la aplicación de la función ***voronoin*** al conjunto de puntos a partir del cual se generará el diagrama, en este caso la *matriz A*, posteriormente se declara la variable *banda* que es simplemente un contador, la tercera línea indica el inicio de un ciclo "*for*" el cual realiza un proceso iterativo con un número de iteraciones igual al número de renglones de C, esto debido a que como ya se mencionó cada renglón de C representa una celda, con lo que se busca analizar cada una de ellas.

```
for a=1:size(C,1)
```

Dentro del proceso del ciclo "*for*" se declara a la *matriz X* de la siguiente forma.

```
X=V(C{a},:)
```

Dicha matriz agrupa las coordenadas de los vértices que forman una celda, para esto se utiliza los vectores de C y la *matriz V*, de tal forma que en cada iteración se llama a un vector de C que contiene los índices de V que forman una celda.

```
>> C{a} %Una vector de C cualquiera. Índices de V
            1     2     3     5
>> V
          Inf       Inf       Inf %índice 1
      -0.3187   -0.9530    0.0858 %índice 2
      -0.2282   -0.7940   -0.1571 %índice 3
      -0.3351   -0.8199    0.1185 %índice 4
       0.1158    0.1617    1.5038 %índice 5
```

Para presentar las coordenadas explícitamente, se refieren los índices directamente a V, para ello se índica que cada elemento de C es un renglón de V.

```
V(C{a},:) %C{a} número de renglón. Para cualquier columna (:)
```

Con esto se presentan, las coordenadas de los vértices de una celda en una matriz, a la cual se le asigna el nombre *matriz X*.

```
>> V(C{a},:)%Coordenadas de los puntos que forman una celda "a"
       Inf       Inf       Inf
   -0.3187   -0.9530    0.0858
   -0.2282   -0.7940   -0.1571
   -0.3351   -0.8199    0.1185
```

Siguiendo con el programa, se establece una condición para trabajar con las celdas cerradas, para esto se utiliza la condición "*if*" de la siguiente forma.

```
if(C{a}(1)~=1)
```

Está condición indica que sí el primer elemento de cualquier vector de C (**C{a}(1)**) es igual a uno no entra al proceso, con está declaración se descartan todas las celdas abiertas, pues como se indicó, si el primer elemento de un vector de C es 1 se refiere a las coordenadas de un punto en el infínito. Al pasar la condición el vector y por lo tanto la *matriz X*, se entra al cuerpo de la sentencia condicional, donde se le asigna a X el nombre de Z, siendo esta *matriz Z* un conjunto de puntos se aplica a dicha matriz la envolvente convexa.

```
K=convhulln(Z)
```

De está instrucción se obtiene a la *matriz K* correspondiente a su conjunto de puntos. En cada iteración se repite el proceso de tal forma que se obtiene por cada *matriz X* una *matriz K*, estás matrices deben ser guardadas de tal forma que exista correspondencia entre ellas. Ya que las dimensiones de X como de K varían dependiendo del tamaño de la celda no pueden utilizarse las hipermatrices, por lo cual se utiliza un tipo de dato conocido en MATLAB® como estructura, que es una agrupación de datos de diferente tipo bajo un mismo nombre, estos datos se llaman miembros o campos. Esquemáticamente puede representarse una estructura de la siguiente manera.

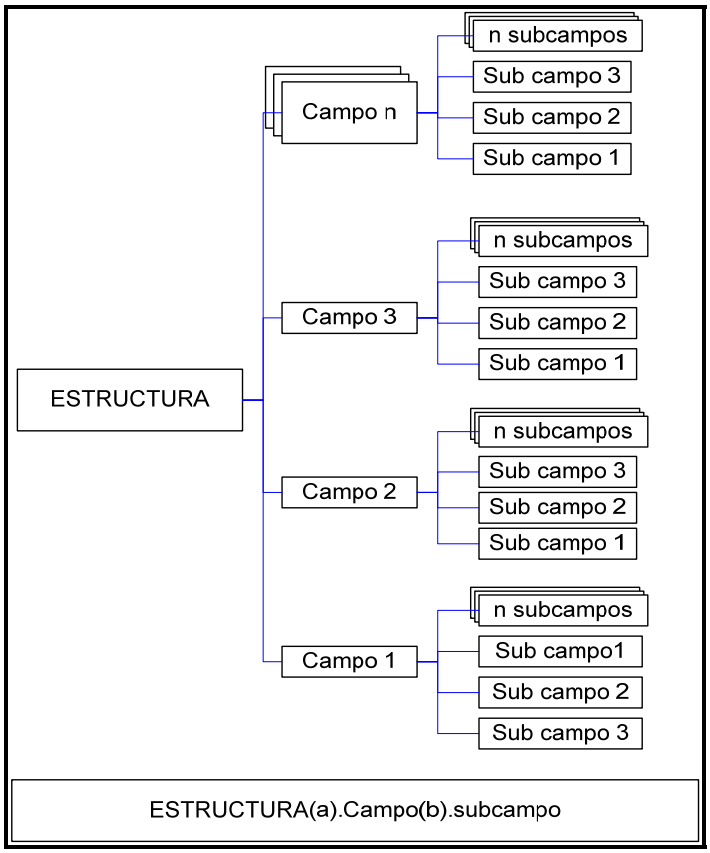


Figura III.10 Representación esquemática de una estructura.

En este tipo de dato no existe una limitante para el número de campos y subcampos, lo mismo ocurre con las divisiones que de ellos se derive, los datos que permiten almacenarse pueden ser de diferente tipo como cadenas de caracteres, matrices, hipermatrices, estructuras, números, etc. las matrices pueden ser o no de las mismas dimensiones. La forma de acceder a los miembros de la estructura se muestra en la parte inferior de la figura III.10. Por ejemplo, la forma de acceder al subcampo 3 que pertenece al campo 2, es escribiendo lo siguiente.

```
Estructura(2).campo(3).subcampo
```

Con esta herramienta se resuelve el problema de las matrices de diferentes dimensiones y su almacenamiento. La forma de aplicar este concepto es a través de la definición de dos estructuras *matrizZ* y *matrizK* cuyo subcampo es *matrix1,* en estas estructuras se guardan las coordenadas del conjunto de puntos y la relación de vértices que forman una celda. La declaración en MATLAB® es la siguiente.

```
        matrizZ(banda).matrix1 %Coordenadas de vértices
matrizK(banda).matrix1 %Relación de vértices que forman una celda
```

Con esta declaración en cada iteración del ciclo en la que se tenga una celda cerrada, se guardaran los datos de dicha celda en la estructura correspondiente, el contador banda será el factor para relacionar a la *matriz Z* con su *matriz K* ya que será el mismo para ambas matrices. Es decir en el momento en que se requieran los datos de la celda diez se declaran.

```
matrizZ(10).matrix1
matrizK(10).matrix1
```

Lo cual devolverá las matrices Z y K correspondiente a la celda 10. Al final del ciclo el contador banda tendrá un valor igual al número de celdas de Voronoi.

En este punto se tienen determinadas las dos características importantes de un arreglo de Voronoi con las cuales queda definido, éstas son las coordenadas de los vértices de las celdas que forman dicho arreglo y la relación que existe entre estos vértices, es decir como es que es que van unidos. A pesar de esto, dada la aplicación a la cual se dirige el concepto se requiere trabajar más los datos de salida, para lo cual se presenta en el apartado siguiente el proceso de manipulación de dichos datos.

III.3 Eliminación de la triangulación y determinación de las aristas.

Como se indico al inicio del capítulo, a partir de este punto no se hará explicito el código en todas las secciones ya que la mayoría del mismo maneja conceptos básicos de programación y se considera que es más importante plantear la lógica manejada, sea hará una presentación del código solo en aquellas secciones donde la explicación del mismo no haga extenso en exceso el capítulo.

Ahora, como se ha mencionado anteriormente, en esté momento se tienen definidas las caras de las celdas del arreglo Voronoi por triángulos, sin embargo el objetivo es definirlas, debido a la aplicación que se les pretende dar, por sus aristas. El proceso de eliminación de triángulos se divide en dos partes, la primera de ellas es agrupar aquellos triángulos que

pertenecen a una misma cara, de manera representativa lo que se busca se presenta en la figura III.11

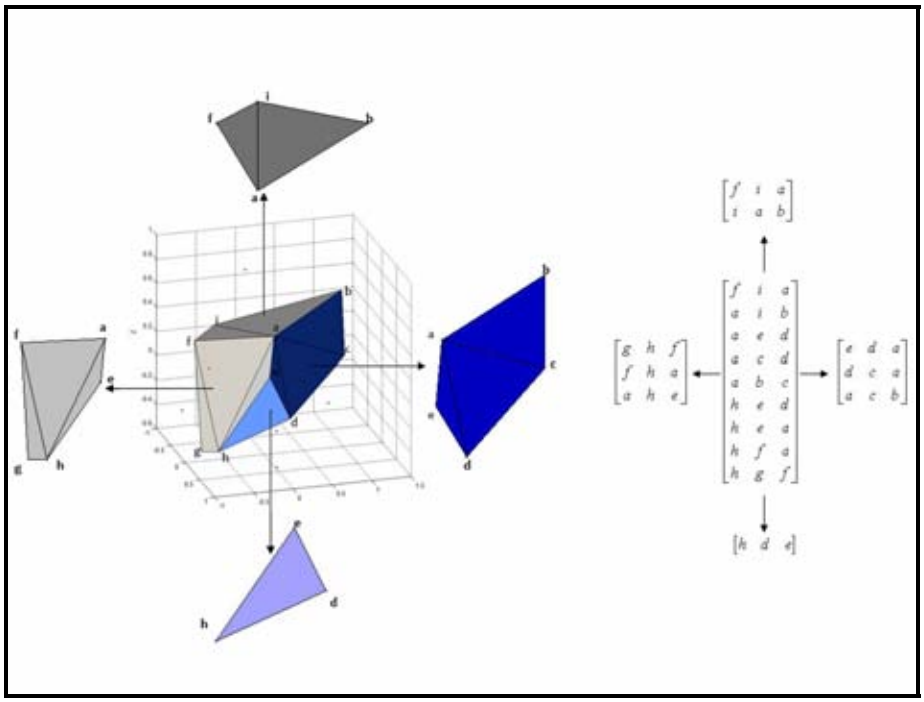


Figura III.11 Identificación y agrupación de los triángulos que pertenecen a una misma cara de una celda de Voronoi.

Es decir, para la primera parte se toma una celda que es una matriz de triángulos y se separa en matrices que contienen triángulos que forman las diferentes caras de la celda, este proceso se repite para todas las celdas que forman el arreglo de Voronoi. La segunda parte del proceso consiste en determinar, a partir de esta agrupación, las aristas de cada cara del arreglo.

Para la descripción de la primera parte del proceso, se presentará a continuación un ejemplo y posteriormente un diagrama de flujo para mostrar cuales son los datos de entrada y salida. El ejemplo se apoya de la siguiente figura.

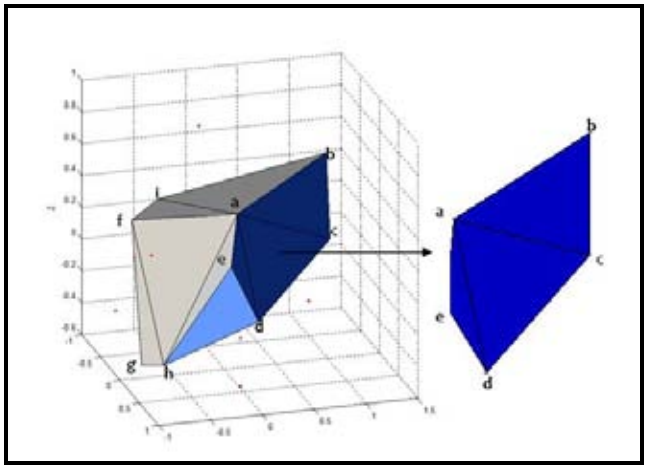


Figura III.12 Agrupación de los triángulos pertenecientes a una cara de la celda.

En este ejemplo se considerará únicamente la descripción para determinar los triángulos que pertenecen a una cara de la celda, los mostrados en la figura III.12, sin embargo el proceso es el mismo para las demás caras e igualmente para todas las celdas. Los datos que se tienen para esta celda son los triángulos que la forman y están dados de la siguiente manera:

$$\begin{bmatrix} a & b & c \\ a & i & b \\ a & e & d \\ a & c & d \\ f & i & a \\ h & e & d \\ h & e & a \\ h & f & a \\ h & g & f \end{bmatrix}$$

El objetivo es determinar cuales pertenecen a la misma cara, en este caso la cara azul de la figura III.12 y agruparlos en una matriz, antes de continuar se debe tener claro que la matriz antes mostrada son únicamente índices que refieren a su respectiva de matriz coordenadas. El proceso para agrupar los triángulos es el siguiente.

1. Se toma el primer renglón de la matriz y con las correspondientes coordenadas de los índices *a, b* y *c* se definen dos vectores directores (***u*** el vector que va del punto *a* al punto *b* y ***v*** el vector que va del punto *a* al punto *c*) también se define el vector normal a ellos, con lo cual es posible determinar la ecuación del plano que contiene a estos tres puntos y por lo tanto a la cara a la cual pertenecen, recordando que un plano queda completamente definido por tres puntos no colineales en este caso *a, b* y *c.*
2. Con la ecuación del plano se analizan todos los renglones, exceptuando aquel con el que se trabaja, es decir se toman las tres coordenadas de los tres índices de cada renglón y se evalúan en la ecuación del plano, solo si las coordenadas de los tres índices satisfacen la ecuación del plano se guarda dicho renglón ya que pertenece al mismo plano que el triangulo de vértices *abc* y por lo tanto a la misma cara. Para este ejemplo se observa que el segundo renglón no satisface esta condición, ya que si bien los vértices *a* y *b* pertenecen al mismo plano para el tercer vértice *i* no ocurre lo mismo por lo cual este renglón no se guarda, para el tercer renglón se observa de la figura anterior que los tres vértices *a, e* y *d* si satisfacen esta condición ya que se encuentran en el mismo plano es decir en la misma cara, por lo cual este renglón si se guarda, para el cuarto renglón ocurre lo mismo ya que los tres vértices *a, c* y *d* se encuentran en el mismo plano como se observa en la figura III.12, por lo cual este renglón se guarda. Para los demás renglones se puede observar por la figura anterior que no satisfacen la ecuación del plano ya que no pertenecen a éste por lo cual la matriz para la cara azul es la siguiente.

$$\begin{bmatrix} e & d & a \\ d & c & a \\ a & c & b \end{bmatrix}$$

Con esta matriz se termina el análisis para esa cara de la celda, el siguiente paso es tomar el segundo renglón y repetir el proceso, con esto al analizar toda la matriz inicial se determinan y agrupan los triángulos de acuerdo a la cara a la cual pertenecen. De este método se puede observar, que se generaran matrices repetidas ya que se crean matrices que

representan caras por cada triangulo de la celda, para el ejemplo, con el primer renglón se obtuvo ya la matriz de triángulos que pertenecen a la misma cara (azul) sin embargo el proceso se repite para cada renglón por lo cual al llegar al tercer renglón se determinará la misma matriz ya que se sigue el mismo análisis e igualmente para el cuarto renglón, por lo cual se tendrán para la cara azul 3 matrices iguales, debido a la complejidad que implica resolver este problema en este punto, se dejara de lado por el momento, ya que este efecto se verá solo como renglones repetidos en una matriz global que contenga a todos los vértices y este problema es más sencillo de resolver al implementar una rutina, que se presentará más adelante, la cual sea capaz de eliminar los renglones repetidos.

Antes de presentar el diagrama de flujo del proceso, es importante tener presente los datos de relevancia que en este punto se tienen, estos son las estructuras *matrizK* y *matrizZ* que esquemáticamente se tiene de la siguiente forma.

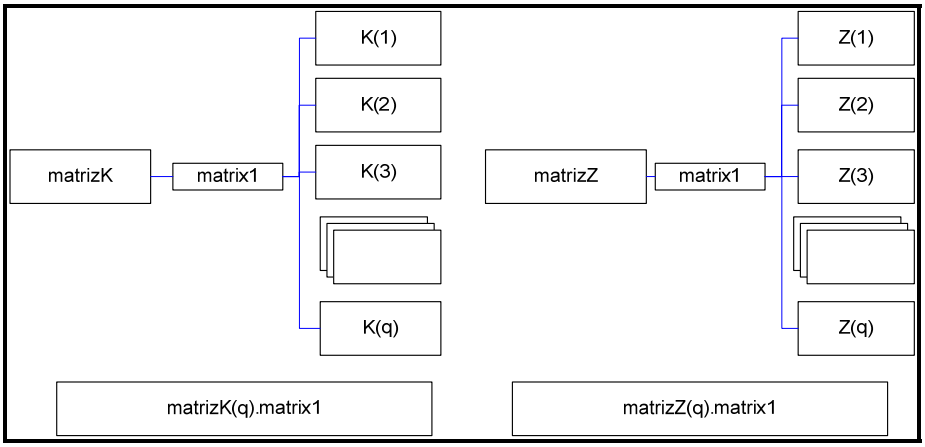


Figura III.13 Representación esquemática de las estructuras *matrizK* y *matrizZ*. En la parte inferior la declaración para acceder a las matrices que guardan cada una de estás estructuras.

Como se observa las estructuras guardan los datos de tal forma que se tienen bien divididos los datos por celda, ya que la matriz K(1) está solamente referida a la matriz Z(1) ambas características de la celda 1, de esta forma se da la correspondencia y no se pierde el orden en los datos. Otro aspecto que se debe tener claro es la manera en la cual se puede trabajar con las coordenadas de los vértices de las celdas y no únicamente con los índices, la manera de determinar las coordenadas mediante líneas de código es la siguiente.

```
matrizZ(q).matrix1(matrizK(q).matrix1(g,i),:)
```

Con la declaración ***matrizZ(q).matrix1*** se indica que se toma la q-ésima matriz *Z*, al abrir el paréntesis se indica que se va especificar que renglón y columna se quiere tomar, el primer espacio es para el renglón, en esté caso va determinado por el índice de la matriz *K*, por lo cual se coloca la notación sombreada, está indica que va a tomar el elemento del g-ésimo renglón de la i-ésima columna de la q-ésima matriz *K*. La forma en que están almacenados los datos se muestran en la figura III.14.

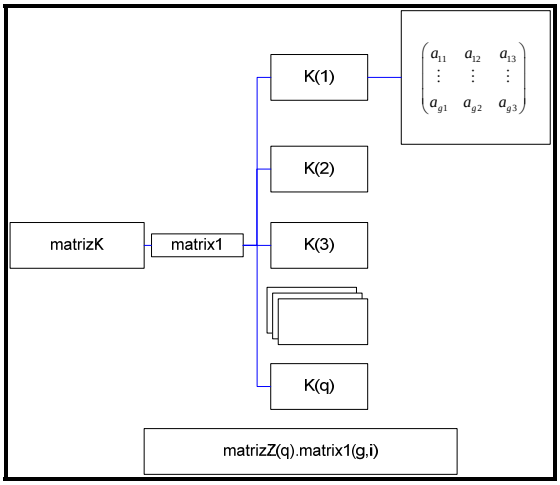


Figure III.14 Esquema de la estructura *matrizK*

Por ejemplo si se requiere el elemento a_{12} de la matriz *K(1)* se declara.

```
matrizK(1).matrix1(1,2)
```

Lo que devolverá un número de renglón de la matriz *Z*. Con esto se tiene especificado el renglón, como las coordenadas deben ocupar las tres columnas se anota el operador dos puntos en el espacio dedicado a las columnas. Para dejar clara esta idea se presenta el siguiente ejemplo.

Teniendo las estructuras y datos como se muestra en la figura III.15 determinar las coordenadas del primer vértice del primer triangulo de la matriz K(2).

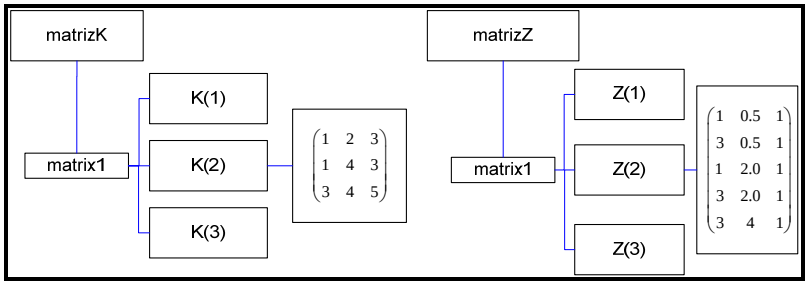


Figura III.15 Ejemplo

El primer triangulo de la matriz de K(2) es el de vértices [1 2 3] para determinar las coordenadas del primer vertice (índice 1), como lo pide el ejemplo, es necesario referirse a su correspondiente matriz Z(2). Para las coordenadas del primer vértice, renglón 1 de la matriz Z(2) se declara lo siguiente.

```
k1_1=matrizZ(2).matrix1(matrizK(2).matrix1(1,1),:);
```

Está línea especifica que se toma a la matriz Z(**2**), el renglón de está matriz debe ser el especificado en la columna 1 reglón 1 de la matriz K(2) para lo cual se declara matrizK(**2**).matrix1(**1**,1)=[1], la columna no importa por lo que se coloca el operador dos puntos. La declaración entrega las coordenadas de la siguiente forma.

```
k1_1=[1 0.5 1]
```

Ahora, teniendo presente los datos de entrada y la forma de acceder a las coordenadas de los vértices se presenta a continuación el diagrama de flujo para esta primera parte.

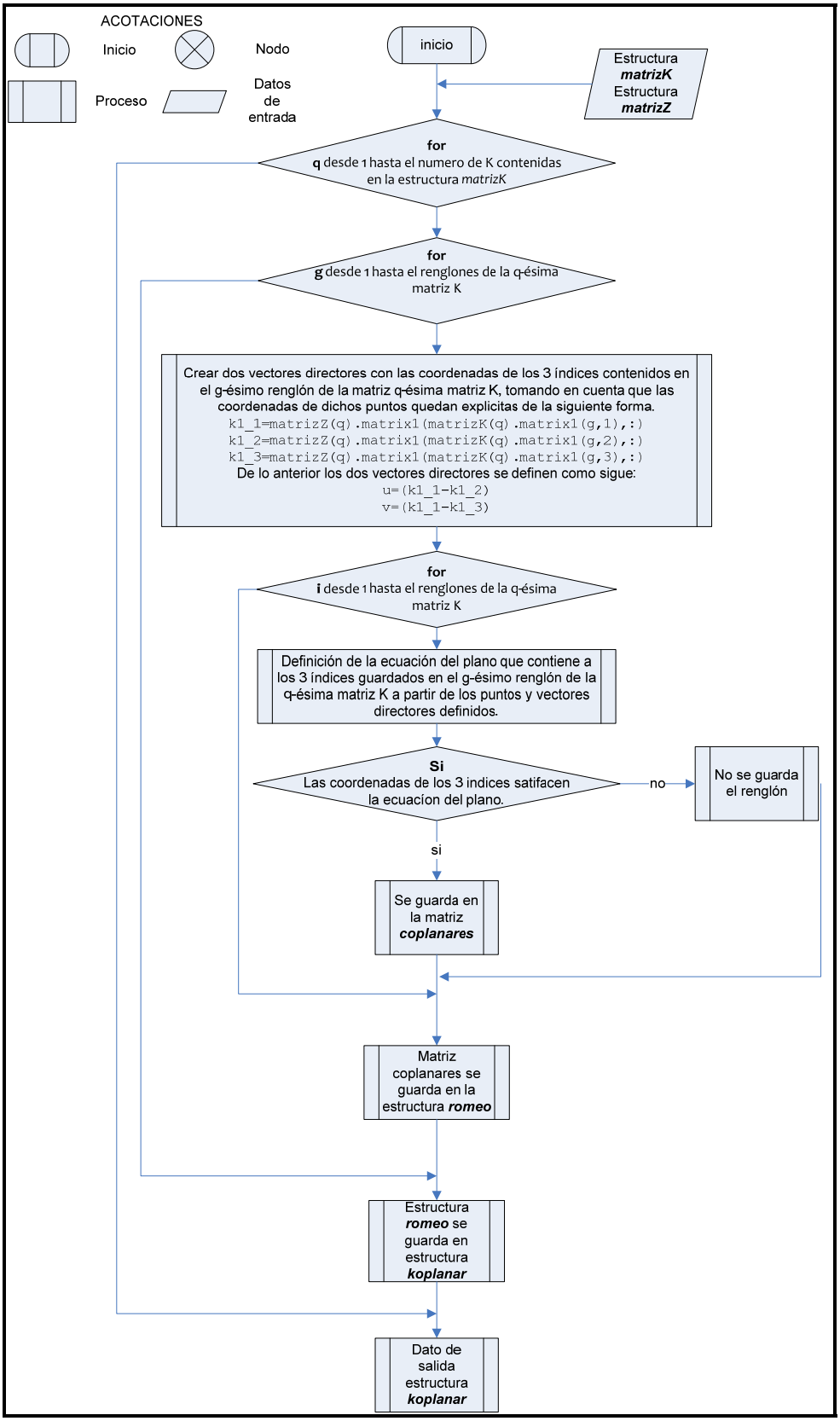


Figura III.16 Diagrama de flujo para la agrupación de los triangulos que forman las celdas de Voronoi de acuerdo a la cara a la que pertenecen

El dato de salida para este diagrama de flujo de manera detallada es el siguiente.

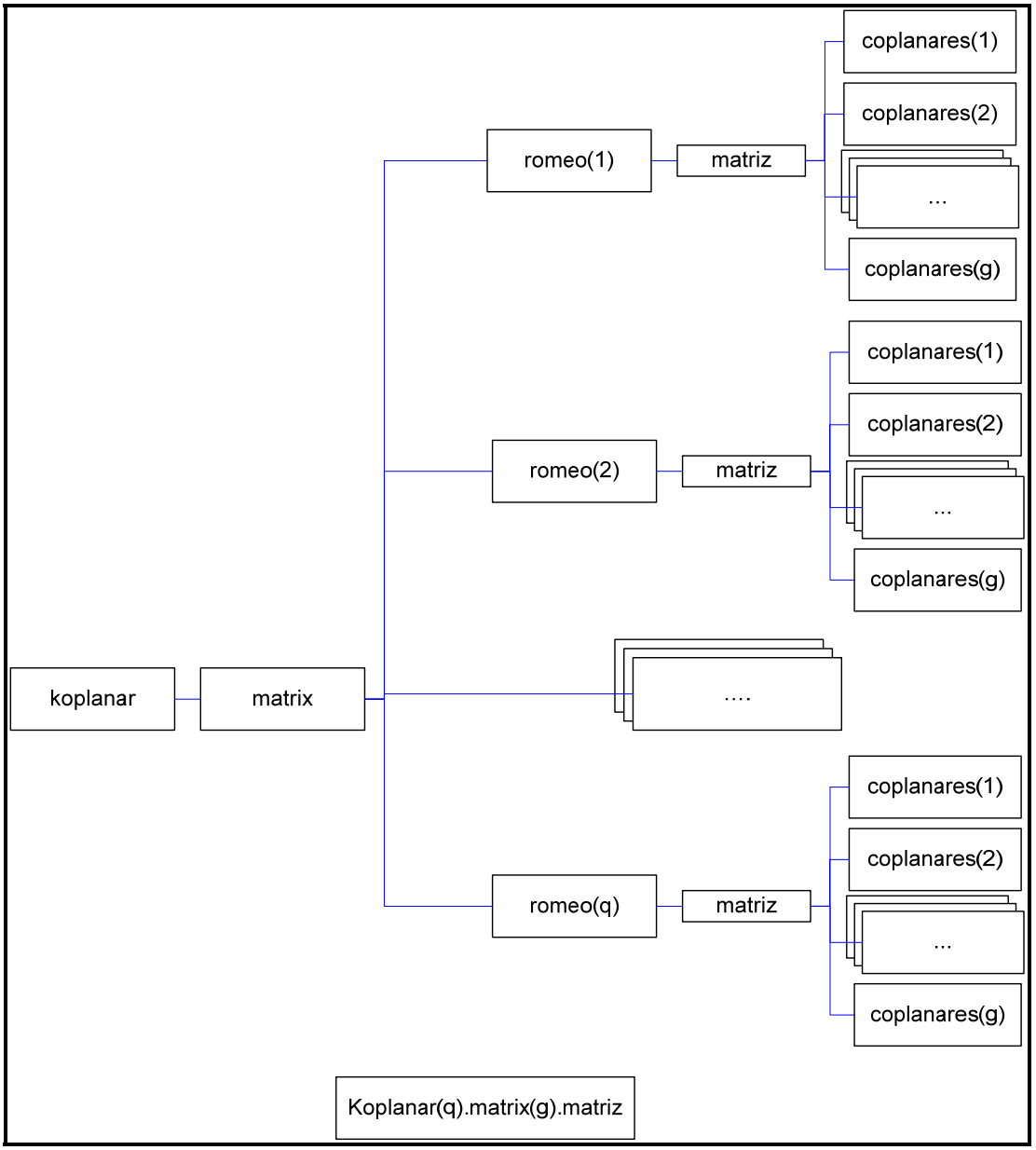


Figura III.17 Estructura *koplanar*. Estructura *romeo* utilizada para dividir los datos. En la parte inferior de la figura se muestra la forma de acceder a cada una de las matrices que se tienen almacenadas en está estructura.

Con la determinación de la estructura *koplanar* se tienen ya separados los triángulos por celdas y por caras, esto cumple con la primera parte de la eliminación de la triangulación; a partir de este punto se trabajara en la segunda parte, es decir, con los grupos de triángulos que pertenecen a un mismo plano para eliminarlos y obtener las aristas a partir de ellos. Para esta segunda parte se presenta a continuación la lógica utilizada que se siguió para la determinación de las aristas, lo cual se hace a través de un ejemplo, el cual se apoya en la figura III.18

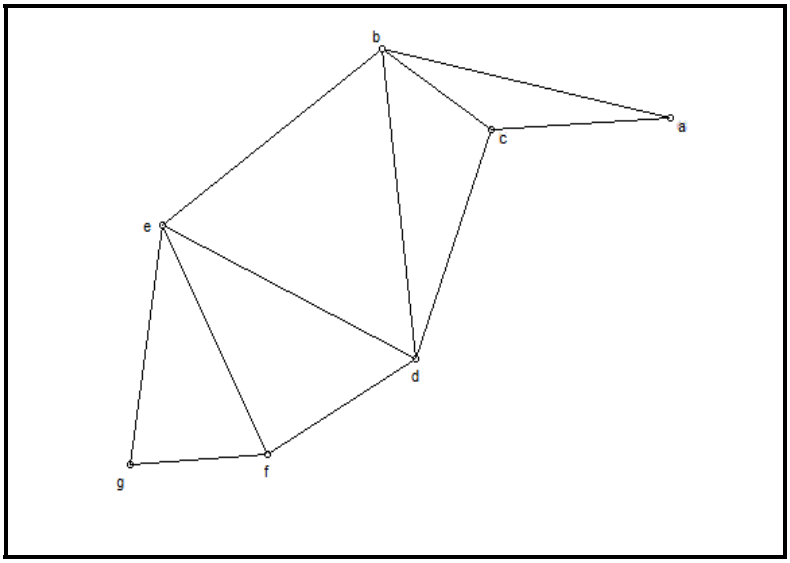


Figura III.18 Polígono irregular formado por triángulos.

Se escoge esta forma por ser arbitraria e irregular. De tener una cara de la manera indicada la matriz de triángulos que tendría asociada sería la siguiente.

$$\begin{bmatrix} b & c & a \\ b & d & c \\ b & d & e \\ d & e & f \\ e & f & g \end{bmatrix}$$

El objetivo es pasar de esta matriz que asocia triángulos a otra que asocie las aristas de la figura. El proceso es el siguiente:

1. Se escoge del primer renglón los dos primeros vértices, en este caso ***b*** y ***c***, se busca en todos renglones, exceptuando aquel con el que se trabaja, si en uno de ellos se encuentran ambos vértices (no importa el orden en que se localicen) indicará que ese lado es compartido por dos triángulos, por lo cual no será una arista. Para el ejemplo se observa que ***b*** y ***c*** aparecen en el segundo renglón por lo cual se descarta como arista.
2. Se escogen ahora los vértices ***b*** y ***a***, se observa nuevamente todos los renglones, exceptuando el primero, buscando si se encuentran en alguno de ellos, para el lado ***b*** y ***a*** se verifica que no aparecen en ninguno de los demás renglones, por lo cual es una arista y se guarda este dato.
3. Debido a que se están tomando en cuenta las combinatorias que se pueden hacer del primer renglón (no importa el orden de los vértices al definir el lado de un polígono, es decir, ***ca*** define el mismo lado que ***ac***) el último lado que puede tomarse en cuenta es el formado por los vértices ***c*** y ***a***, por lo tanto se busca en todos los renglones y se determina que no se encuentran en ningun otro renglón, por lo cual es una arista.

Este proceso se lleva acabo para cada uno de los renglones, es decir tomando las combinatorias posibles de los elementos del renglón y guardando los lados que satisfagan la condición de no ser compartidos por dos triángulos. Al finalizar el proceso se tendrá a la figura determinada por sus aristas y a la matriz asociada a ello como lo muestra la figura siguiente.

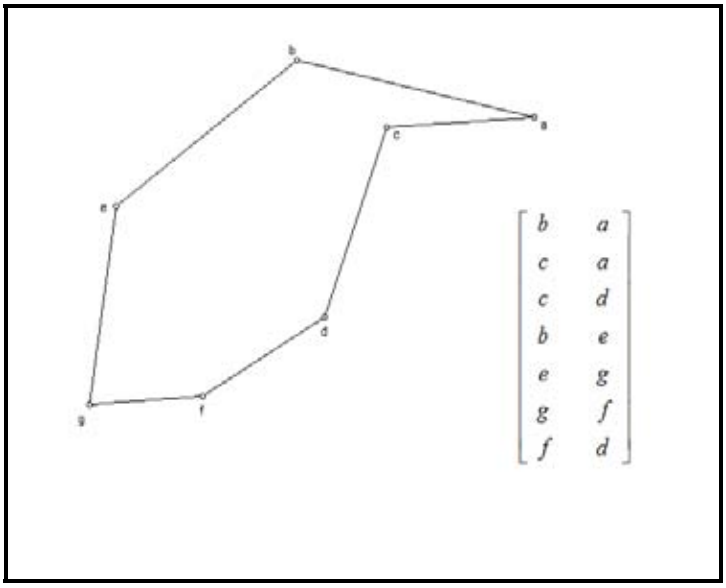


Figura III.19 Polígono irregular determinado por sus aristas adjunta su matriz de aristas.

Con lo anterior, se tiene la descripción del proceso para determinar las aristas a partir de una triangulación, a continuación se presentará de manera general este proceso.

Debido a que esta rutina de trabajo utiliza elementos básicos de programación como son los condicionales ("*if*" y "*while*"), ciclos "*for*" y elementos como las estructuras que se explicaron previamente, no se presentará el código en lugar de ello se presenta el diagrama de bloques del programa con el objeto de ahorrar espacio en la explicación de un código sencillo pero largo. Para seguir dicho diagrama debe tenerse presente la figura III.17 y lo siguiente:

- Estructura *koplanar.* Contiene en su campo *matrix* a las estructuras *romeo*, cada una de ellas representa a una celda y por ello contiene datos en forma de ordenada de cada una de ellas. Para acceder a cada estructura romeo se escribe lo siguiente ***koplanar(f).matrix*** donde ***f*** es el número de estructura romeo que se analiza.
- Estructura *romeo.* Cada una de estas estructuras contiene en su campo *matriz* a las matrices *coplanares* que contienen a los triángulos que pertenecen a una misma cara en una celda de Voronoi. Para acceder a cada matriz coplanar de una celda (***f***) se escribe lo siguiente ***koplanar(f).matrix(n).matriz*** donde ***n*** es el número de matriz coplanar que se analiza.

Las acotaciones para los diagramas de bloques que se presentaran a partir de este punto.

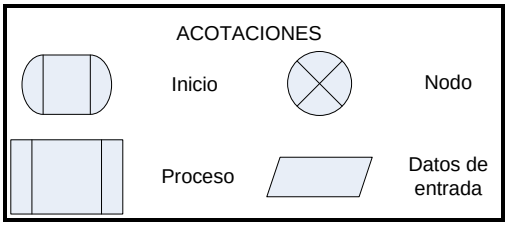


Figura III.20 Acotaciones para diagramas de bloques.

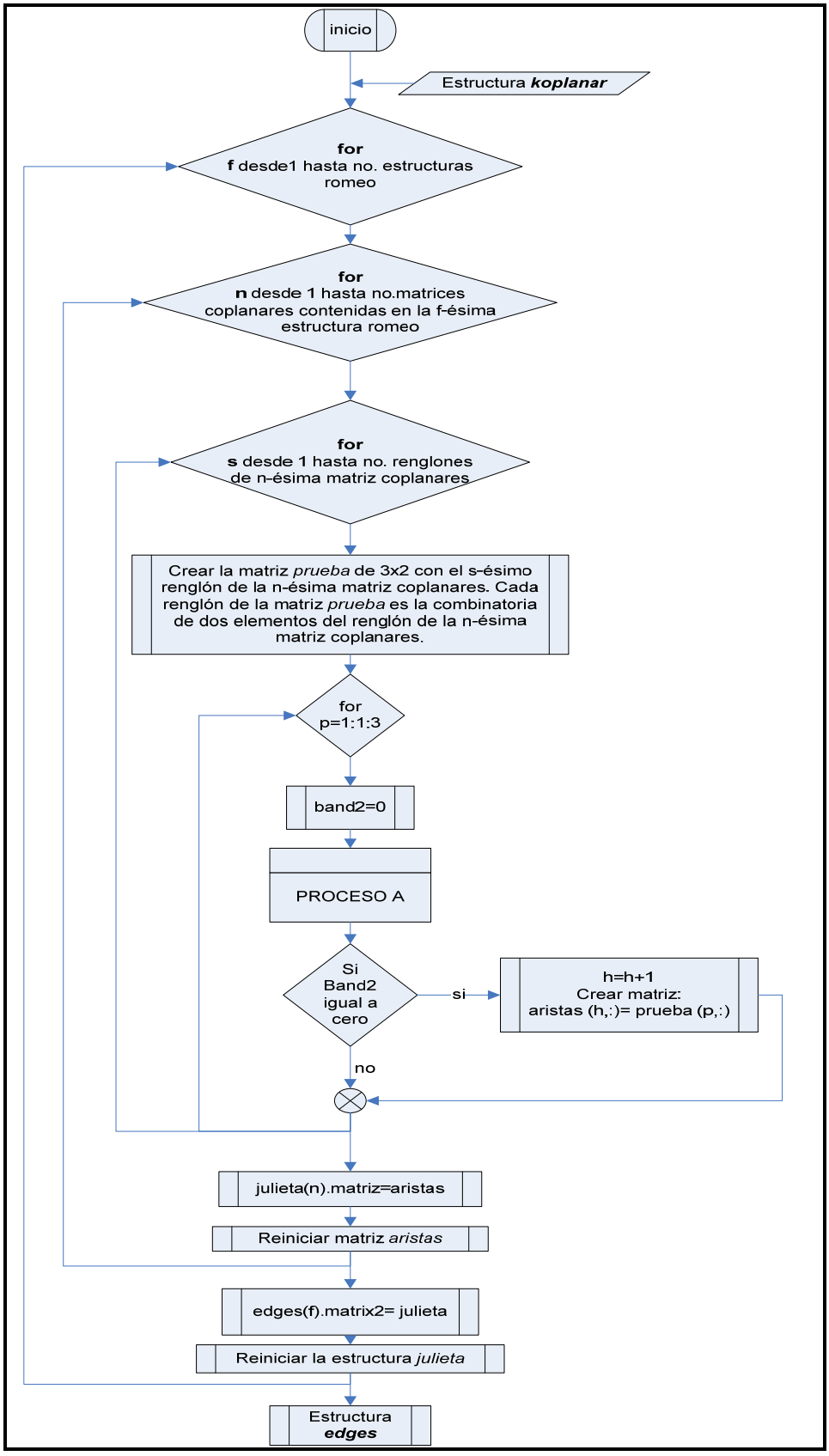


Figura III.21 Diagrama de bloques del proceso de determinación de aristas.

Debido a la extensión del diagrama se opto por tomar una parte del diagrama de bloques y desarrollarlo en un esquema separado, el cual se muestra a continuación. Esta parte corresponde al PROCESO A que se presento como un bloque de proceso en el diagrama anterior.

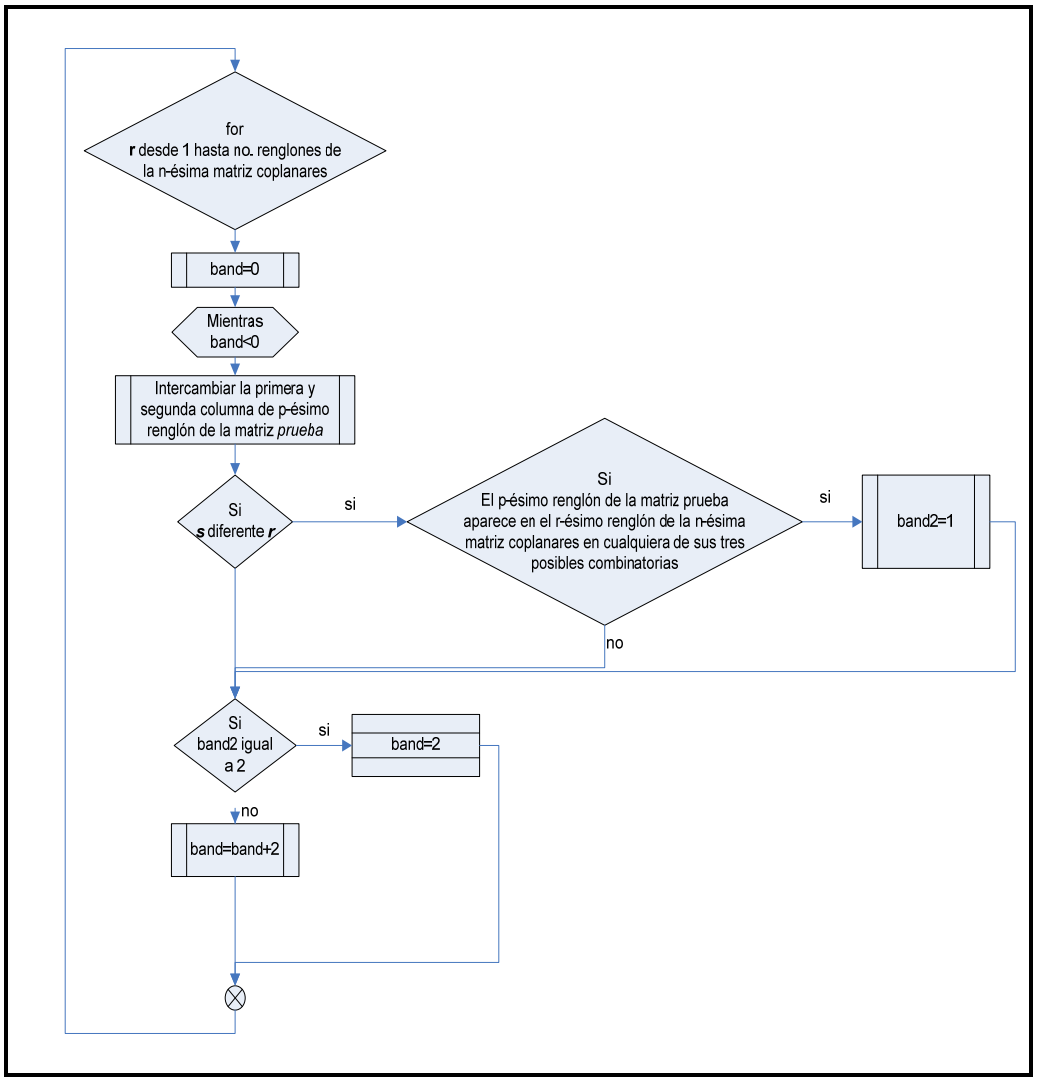


Figura III.22 Diagrama de bloques PROCESO A

Al final del proceso se puede observar que se generan dos estructuras *julieta* y *edges* (*figura III.21*), esto se hace con el fin de no perder el orden en cuanto a las celdas con las que se trabaja así como las caras de dichas celdas, es decir se separan los datos por celda y por caras, de manera esquemática como se presenta en la figura III.23 En dicha figura se puede observar que cada matriz *julieta* corresponde a una celda, dentro de cada una de estas estructuras se guardan las aristas de la celda, sin embargo se separan de acuerdo a la cara que pertenecen mediante las matrices *aristas*. De esta manera se tienen ya los datos prácticamente de la manera que se requieren para exportarlos a la paquetería de elementos finitos, no obstante se debe recordar que en este punto se tienen índices y no coordenadas que es lo que finalmente importa, por lo cual el siguiente paso es obtener las coordenadas los vértices de las celdas para lo cual tiene la siguiente rutina de trabajo.

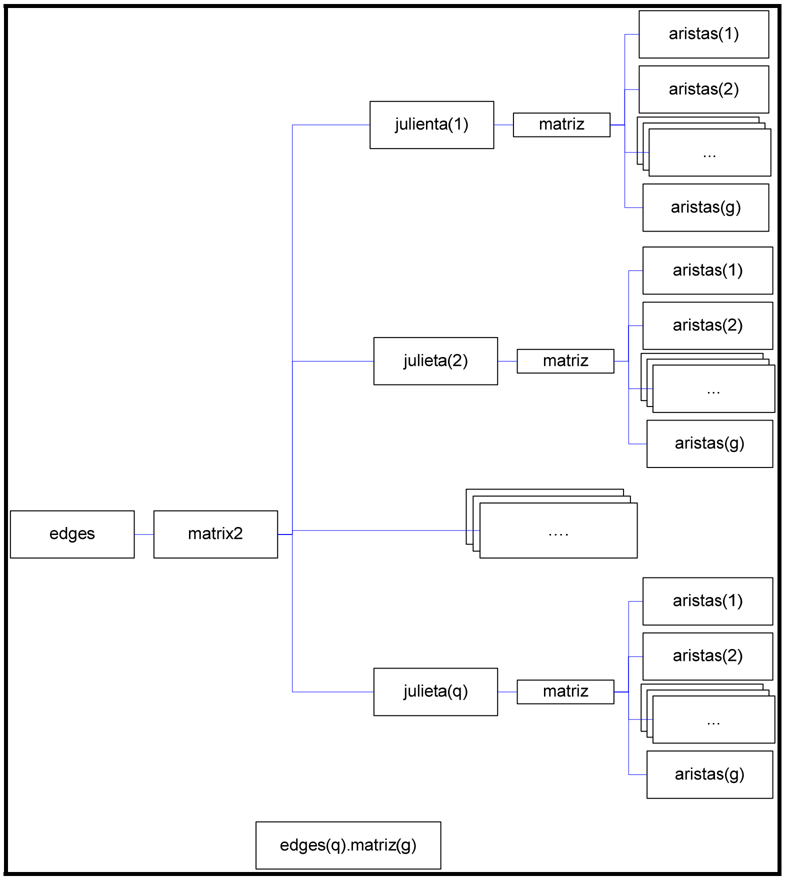


Figura III.23 Datos de salida del proceso de determinación de aristas.

```
%DETERMINACIÓN DE LAS COORDENADAS DE LAS ARISTAS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Coordenadas = [ , ];
marca7=size (matrizZ);
for k=1:1:marca7(2) %PRIMER CICLO
    marca1=size (edges (k).matrix2);
        for y=1:1:marca1(2) %SEGUNDO CICLO
            marca2=size (edges (k).matrix2 (y).matriz,1);
                for x=1:1:marca2 %TERCER CICLO
      coordenadas = [coordenadas; vert1, vert2];
      vert1 = [matrizZ (k).matrix1 (edges (k).matrix2 (y).matriz(x,1),:)]
      vert2 = [matrizZ (k).matrix1 (edges (k).matrix2 (y).matriz(x,2),:)]
                end
                coordenadas;
        end
 koordinaten =coordenadas;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Se observa que esta rutina tiene tres ciclos. El *PRIMER CICLO* se crea con el fin de recorrer a todas las celdas, por lo cual se utiliza como limite superior del ciclo el número de matrices Z guardadas en la estructura *matrizZ* de la siguiente forma ***size (matrizZ, 2)***, se recuerda que cada matriz Z contiene las coordenadas de los vértices de una celda de Voronoi. El SEGUNDO CICLO hace referencia al número de caras de cada celda, por lo cual se indica que

el limite superior de este ciclo es de la forma siguiente **size (edges (k).matrix2)** donde **k** es la variable de control del primer ciclo. El TERCER CICLO recorre cada renglón de la matriz de aristas de la celda y cara correspondiente, este ciclo tiene la función de referir los indicies a las coordenadas que le corresponden, por lo cual, dentro de él se definen a las variables *vert1* y *vert2* como sigue:

```
vert1 = [matrizZ (k).matrix1 (edges (k).matrix2 (y).matriz(x,1),:)]
```

En está línea se observa que *vert1* refiere a la estructura *matrizZ*, es decir una estructura de matrices de coordenadas, la parte sombreada es el número de renglón que se debe tomar para extraer a las coordenadas de la k-ésima matriz Z, este número de renglón se relaciona con la estructura *edges* que contiene a los índices de las aristas de las celdas, por lo que se toma el elemento de la primera columna de un renglón, cara y celda especifica, es decir un índice, en este caso el renglón correspondiente a la matriz Z, para *vert2* cambia el hecho de tomar la primera columna ya que se toma la segunda, con ello se extraen los dos vértices de una arista y se acomodan en un mismo renglón dentro de la matriz *koordinaten,* con las iteraciones de los tres ciclos se abarcan todos los datos por lo cual la dimensión de está matriz al final del proceso es de nx6 donde *n* es el número de vértices del arreglo. Al obtener a la matriz *koordinaten*, se tiene una sola matriz sin arreglos por celda o por cara, solo se tienen las coordenadas de los vértices de Voronoi ordenados de tal forma que se define el arreglo por aristas.

$$koordinaten = \begin{bmatrix} x_1^1 & y_1^1 & z_1^1 & x_2^1 & y_2^1 & z_2^1 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ x_1^n & y_1^n & z_1^n & x_2^n & y_2^n & z_2^n \end{bmatrix}$$

Figura III.24 Representación de la matriz *coordenadas.*

El siguiente paso es depurar los datos obtenidos en la matriz *coordenadas*, se busca eliminar dos tipos de datos, uno de ellos son los renglones repetidos y el segundo son los renglones que contienen a los dos mismos elementos pero en diferente orden.

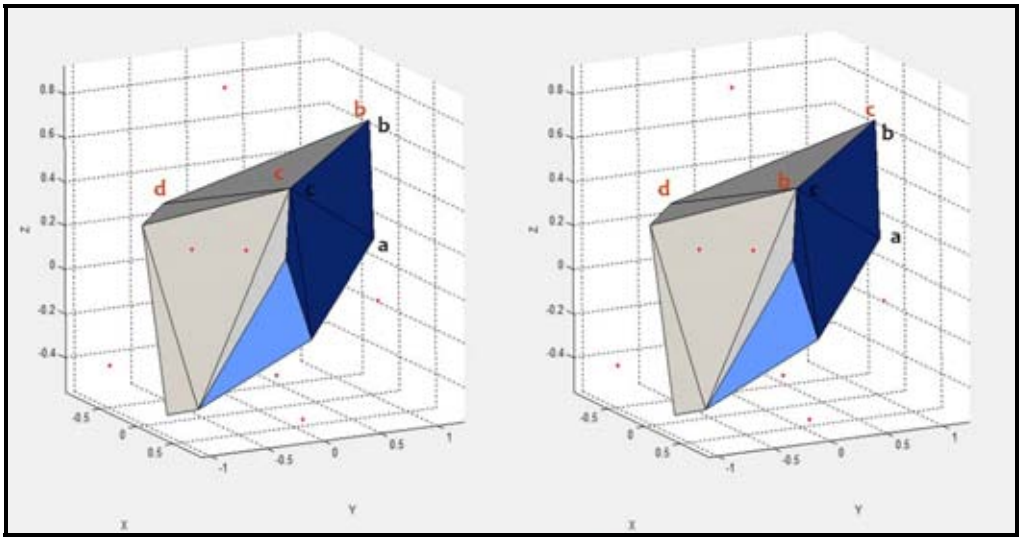


Figura III.25 Representación gráfica de la presencia los renglones repetidos y renglones "espejo" en la matriz *coordenadas*

Estos datos se presentan debido a que se trabajaron por separado cada una de las caras de las celdas, por lo cual todas las aristas de un cara son compartidas por otra, pudiendo estar en la matriz *coordenadas* exactamente iguales o invertidas (figura III.25), ya que solo se necesita definir una sola vez estas aristas deben eliminarse para agilizar el proceso y reducir el tamaño de las matrices con las que se trabajan, para esto se generaron las rutinas para eliminar los renglones repetidos y otra para eliminar los renglones "espejo".

A continuación se presenta el diagrama de flujo de la rutina dedicada a la eliminación de los renglones repetidos.

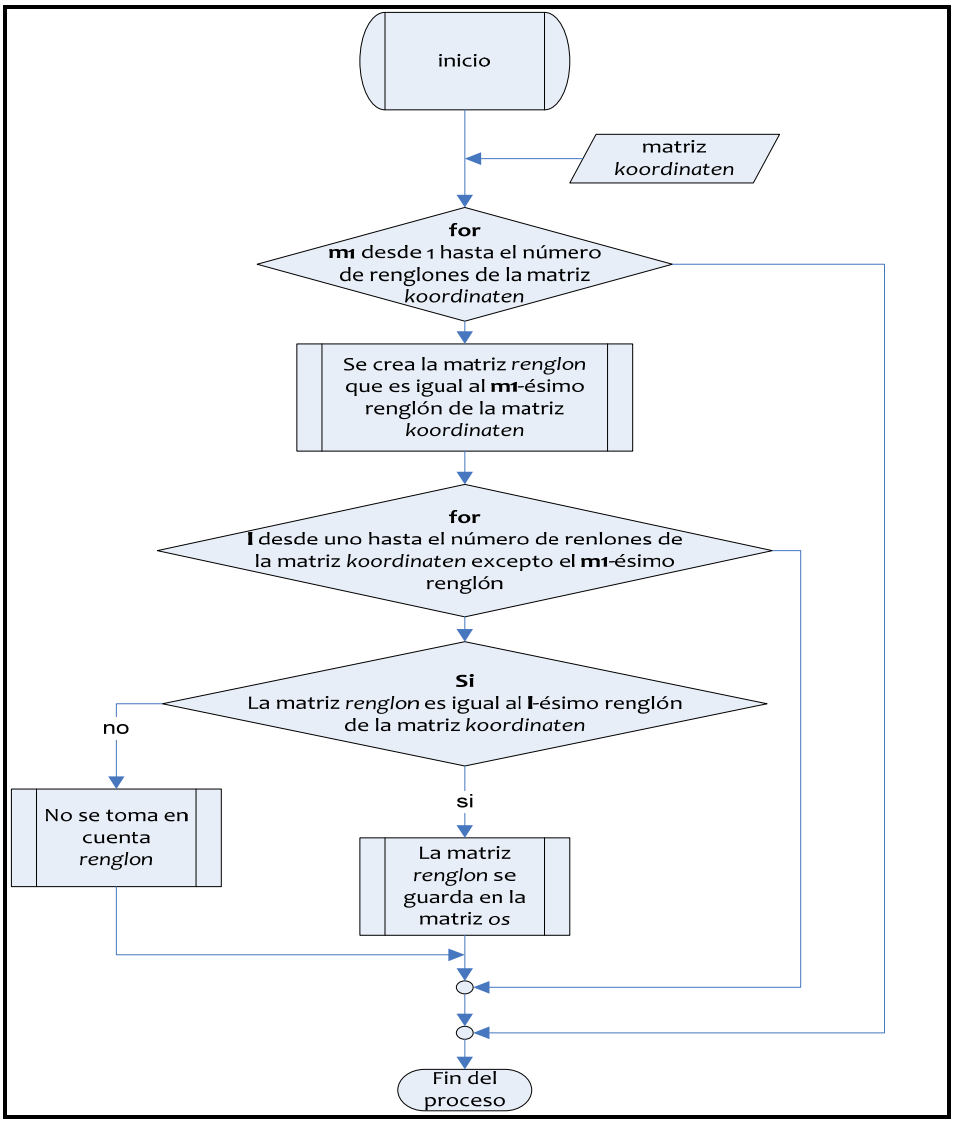


Figura III.26 Diagrama de bloques del proceso de eliminación de renglones repetidos

Se puede observar que como dato de salida se tiene a la matriz *os* que representa exactamente lo mismo que la matriz *koordinaten* simplemente dicha matriz paso por la primera parte de depuración y no existen en ella renglones repetidos. El siguiente paso es la eliminación de los renglones espejo, para describir dicha rutina se presenta el siguiente diagrama de flujo (figura III.27).

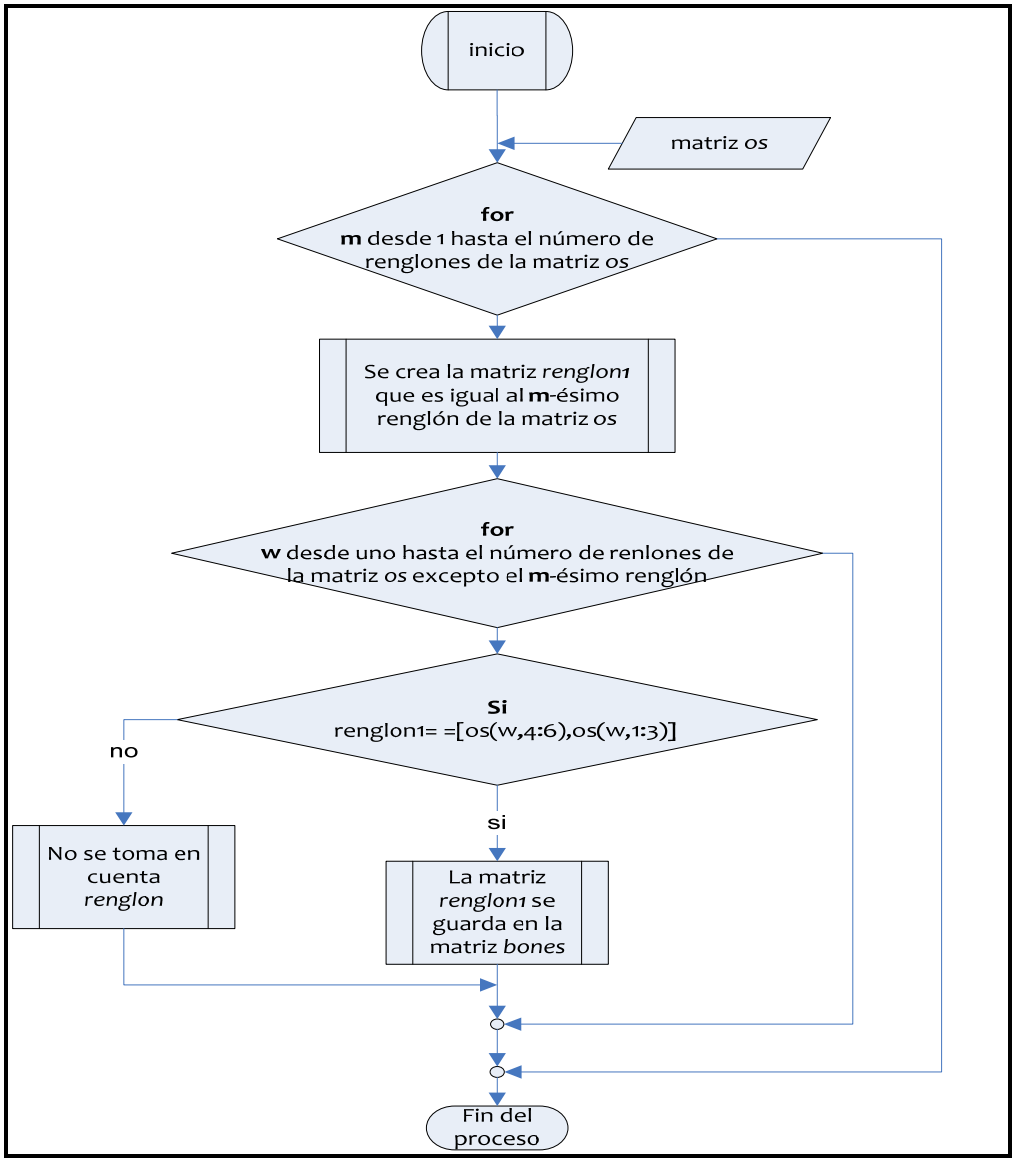


Figura III.27 Diagrama de flujo de la eliminación de los renglones espejo.

Para este diagrama es importante dejar clara la condición, por lo cual con este afán se deja explicita las variables que en ella se ocupan.

$$renglon1 = [v_{x_1}\ v_{y_1}\ v_{z_1} \quad v_{x_2}\ v_{y_2}\ v_{z_2}]$$

$$[os(w,4:6), os(w,1:3)] = [v_{x_2}\ v_{y_2}\ v_{z_2} \quad v_{x_1}\ v_{y_1}\ v_{z_1}]$$

Con la determinación de la matriz *bones* establecida en el proceso anterior, se tiene ya la matriz de coordenadas de aristas depurada. El siguiente paso es obtener del arreglo de Voronoi, la muestra de trabajo, es decir un cubo de 4.864 unidades de lado, para ello se deben imponer condiciones de frontera en las direcciones **x, y** y **z** para ejemplificar el proceso por medio del cual se hicieron los cortes se describirá a continuación el procedimiento para cortar al arreglo en el plano Z=0.123, es decir, a las aristas que atraviesen el plano se cortaran asignándoles un nuevo vértice que está sobre este limite, las aristas que no se encuentren en este caso se guardaran en un nueva matriz *fronteraii*. Para describir este proceso se utiliza el siguiente diagrama de flujo.

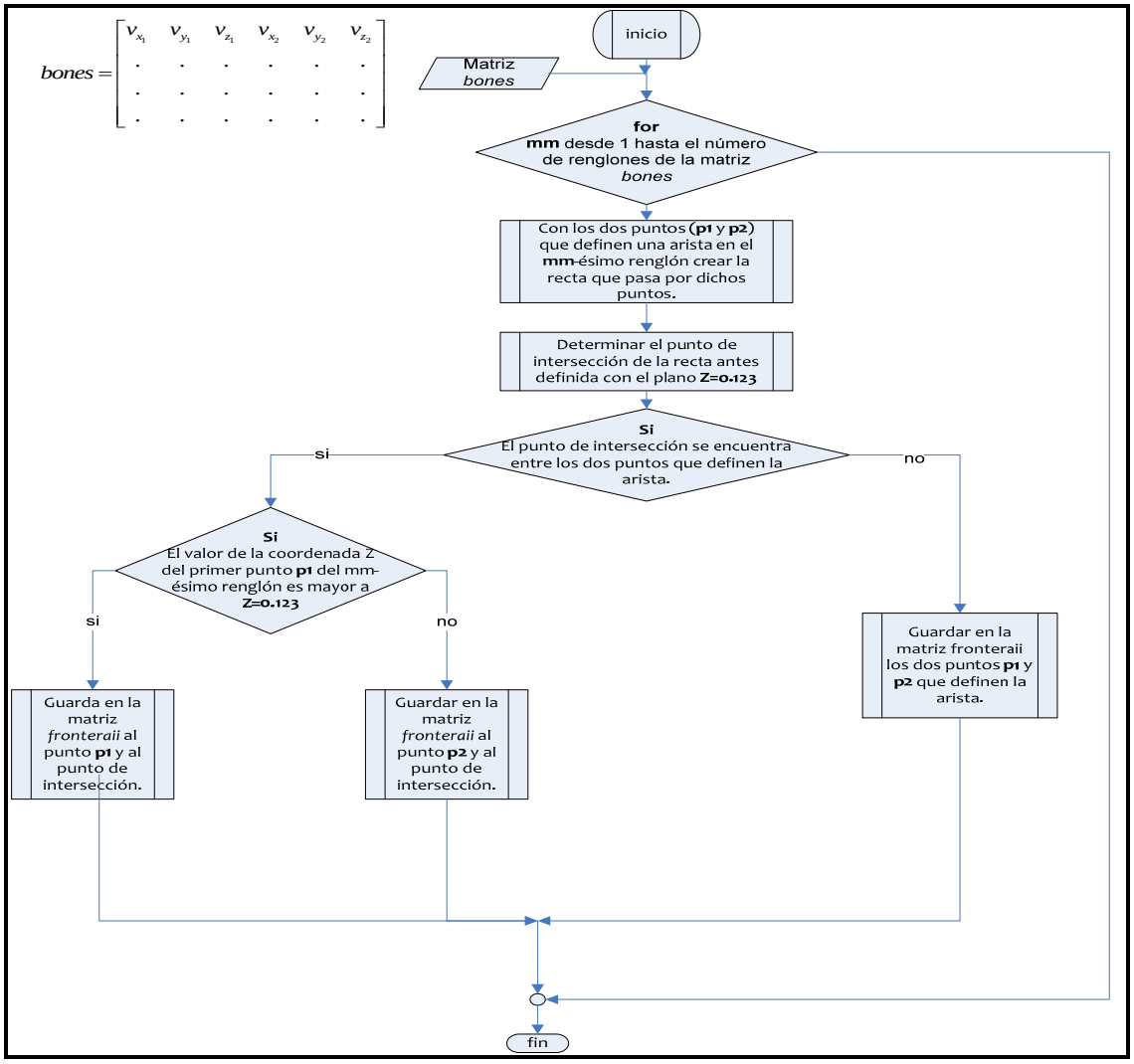


Figura III.28 Diagrama de flujo del proceso para el corte en el plano Z=0.123

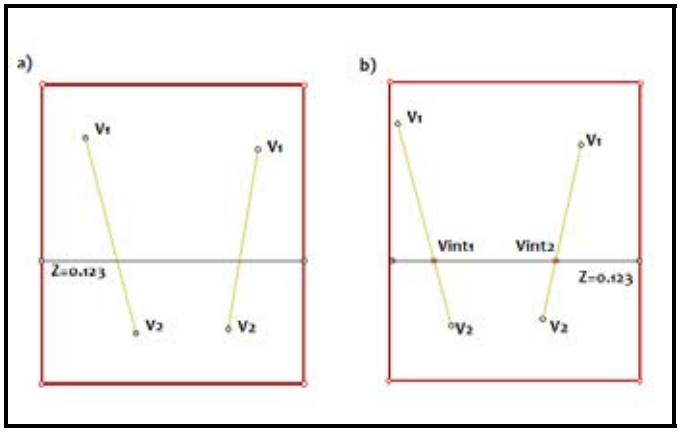


Figura III.29 Representación gráfica del proceso de corte. a) Identificación de las aristas que cruzan el plano de cote. b) Determinación del punto de intersección de la arista.

Mediante las figuras III.28 y III.29 se muestra el proceso de corte, en el cual se identifican las aristas que cortan al plano, se determina el punto de intersección y se guarda en la matriz *fronteraii* el segmento de recta que se encuentre arriba del plano, en este caso Z=0.123, para toda arista que no se encuentre en ese caso se guarda en la misma matriz. Para realizar el corte en el plano Z=4.987 se sigue de manera análoga el proceso anterior, cambiando únicamente que el segmento de recta que se guarda es el que se encuentra debajo del plano Z=4.987. Con esto se ha limitado parcialmente el arreglo entre los planos Z=0.123 y Z=4.987, se indica que parcialmente ya que si bien se cortaron aquellas aristas que atravesaban el limite, toda viga que no se encontrara en ese caso se guardaba, por lo cual en el caso del corte en Z=0.123, aquella arista que no cortara el limite pero estuviera por debajo del plano se guardaba, dado que satisface la primera condición que se muestra en el diagrama de flujo de la figura III.28, para eliminar estas aristas y acotar perfectamente entre los planos antes mencionados se tiene la siguiente rutina.

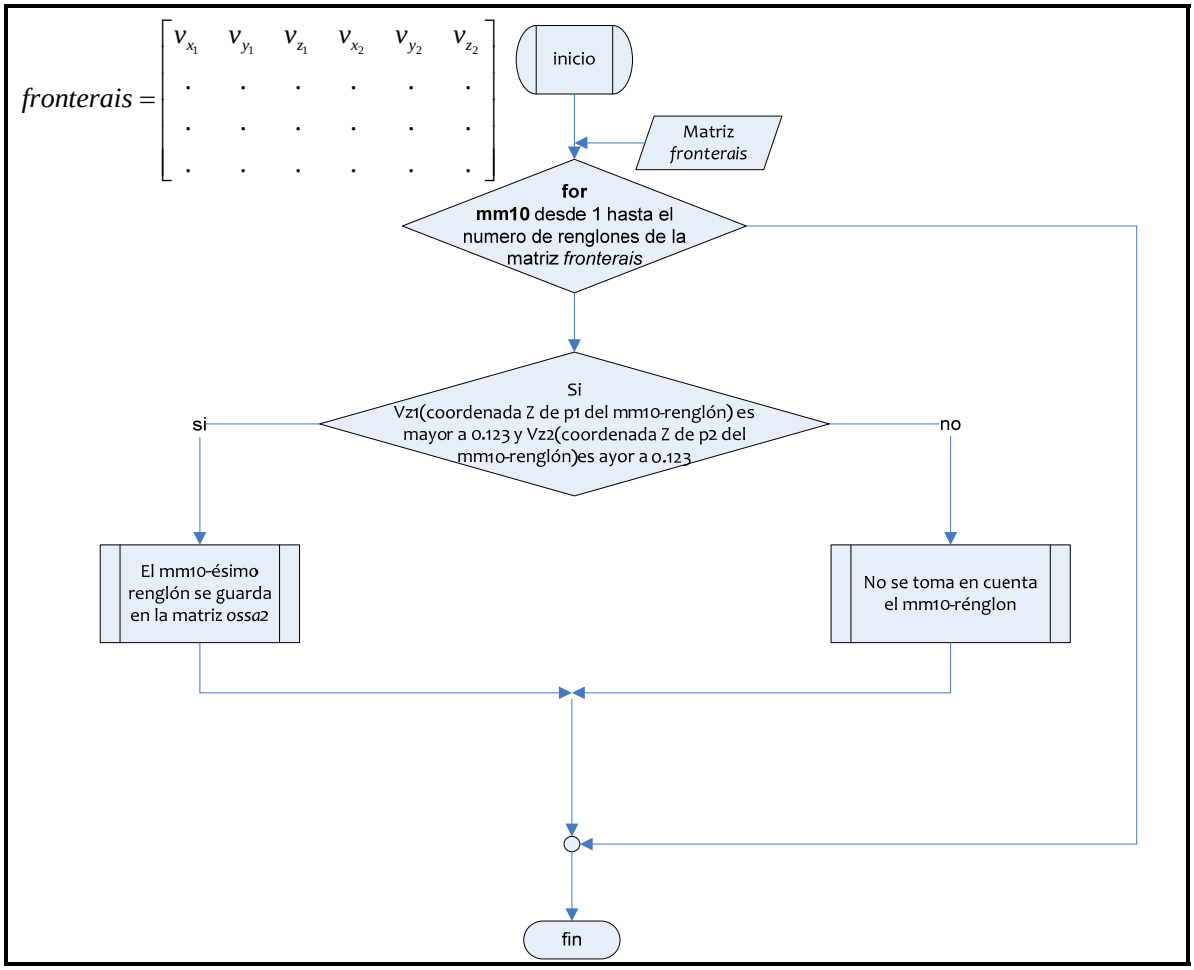


Figura III.30 Diagrama de flujo de la rutina para eliminar todos las aristas cuyas coordenadas z sea menor a 0.123. Dato de salida matriz *ossa2* nx6.

Se puede observar de este diagrama de flujo que la matriz de entrada es *fronterais,* esta matriz se determina después de efectuar el corte en el plano Z=4.987, este proceso no se mostró debido a que como ya se mencionó es similar al del corte en Z=0.123. La rutina que se presenta en la figura III.30 elimina sólo vértices cuya coordenada ***z*** es menor a 0.123 en el

mismo renglón, es decir, las aristas que están debajo de este plano y guardando en *ossa2* las aristas que se encuentran arriba de este. Para eliminar los datos que están sobre el plano Z=4.987 se sigue un proceso similar al anterior modificando solo la condición, es decir, se guardaran los vértices del mismo renglón cuya coordenada **z** sea menor a 4.987, implementando ambos procesos se llega a delimitar perfectamente el arreglo del plano Z=0.123 al plano Z=4.987. Para realizar los cortes en las direcciones de X y Y se sigue el mismo procedimiento, cambiando únicamente la condición de frontera, para los cortes en la dirección ***x*** se trabajo con las condiciones X=0.123 y X=4.987, en la dirección ***y*** se trabajó con las condiciones Y=0.123 y Y=4.987. Cabe mencionar de este proceso que la razón de utilizar valores con cifras tan singulares (0.123 y 4.987) se hizo con el fin reducir la posibilidad de que algún vértice de una arista se encontrara en el límite, ya que dicha situación provocaba que el programa arrojará errores, al no determinar la solución de dichos errores se optó por colocar los limites con los valores antes mencionados con lo cual se evitaron los errores y la modificación del algoritmo.

A lo largo de la generación y evaluación del programa se observó que después de los cortes, se presentaba la situación donde algunos de los vértices de una arista eran exactamente los mismos, con lo cual en lugar de tener una arista se tenía un punto, después de analizar el proceso y llevar a cabo varias corridas del programa se llegó a la conclusión de que el problema se presentaba al tener una arreglo de vigas muy complicado y debido al manejo de cifras significativas de MATLAB® ya que al trabajar con valores exactos y arreglos sencillos no se presentaba dicho problema. Como este problema no alteraba el arreglo de Voronoi y los casos en que se presentaban eran mínimos, se optó por eliminarlos puntualmente y seguir trabajando con el arreglo, la rutina que se utilizó para dicha tarea se presenta en el siguiente diagrama de flujo.

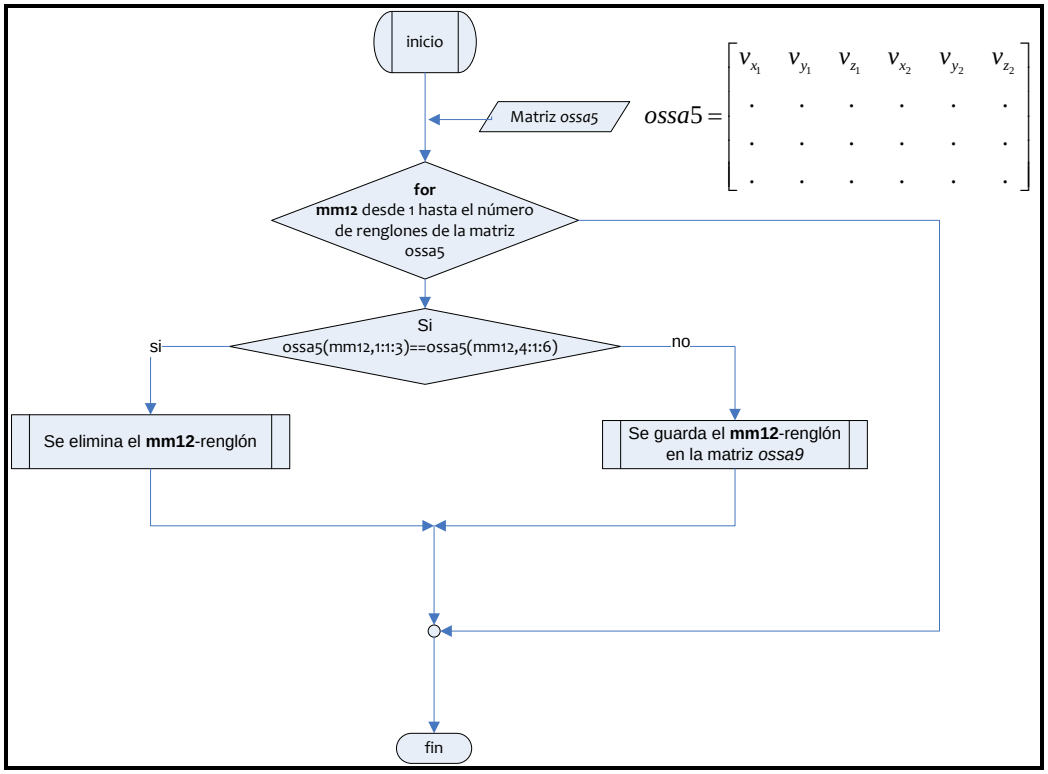


Figura III.31 Diagrama de flujo para eliminar los renglones que definen un punto. La matriz de salida es *ossa9*.

Como dato de salida de la rutina antes presentada se tiene a la matriz *ossa9,* dicha matriz presenta en un renglón las coordenadas (x, y, z) de los dos vértices que definen una arista después de los pasos de depuración antes descritos. El proceso que sigue es eliminar de las muestras de Voronoi elementos que físicamente no pueden existir después de un corte, estos son aquellos que no tiene forma de mantenerse unida a la estructura por lo cual en un corte real caerían de la pieza. Los elementos que se buscan eliminar son casos especiales y se agrupan como se muestra en la siguiente figura.

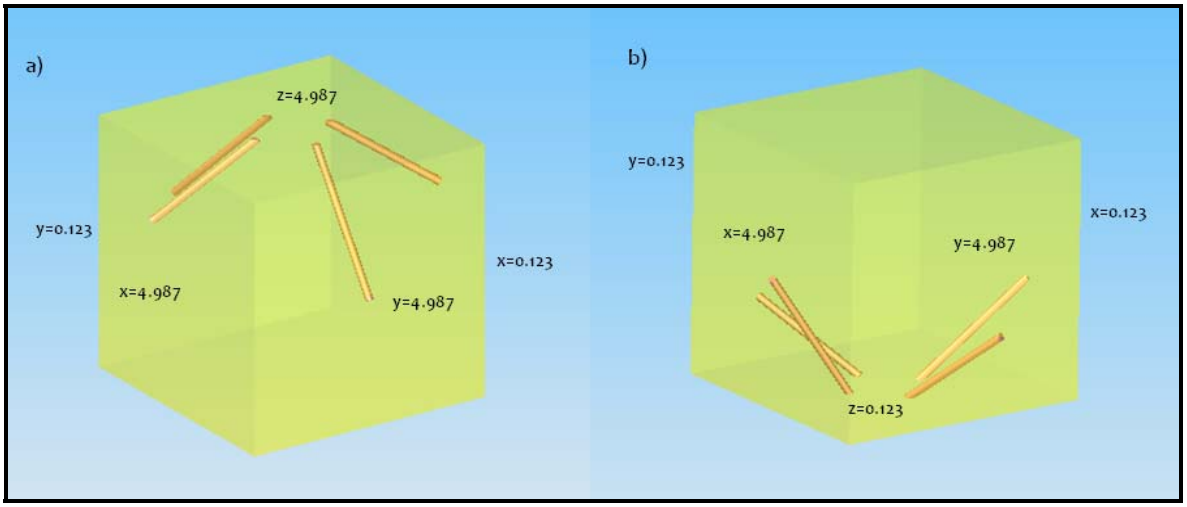


Figura III.32 Representación gráfica de los elementos que físicamente no pueden mantenerse unidos a la estructura.

En la figura anterior se puede observar que cada una de las vigas que se consideran ya no pueden formar parte del arreglo, porque no tienen unión con el resto de la estructura después de los cortes. En está figura se muestran cada uno de los planos con los que tienen intersección las vigas, a partir de dichas intersecciones se crearon pequeñas rutinas para eliminar a cada una de estas vigas. Cabe mencionar que estas no son todas las posibilidades que se consideraron, la figura III.33 que se muestra a continuaciones presentan el resto de las posibilidades.

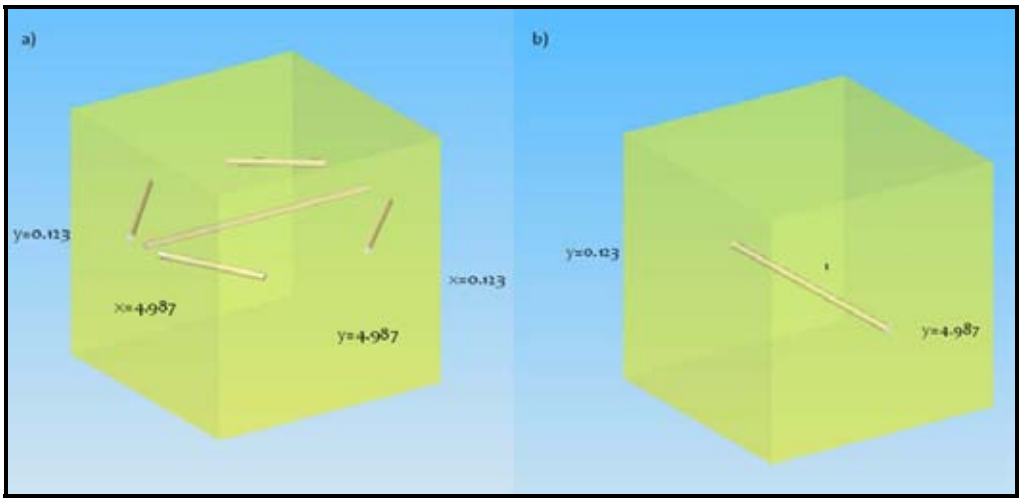


Figura III.33 Elementos que físicamente no pueden mantenerse unida a la estructura.

En última figura se consideran otras posibilidades en donde los elementos ya no puede forma parte de la estructura. Para cada posibilidad se creó una rutina donde se busca eliminar toda viga que se entre en dicha posibilidad, para ejemplificar el proceso, se presenta a continuación la rutina para eliminar la viga marcada con un el número uno en la figura III.33b.

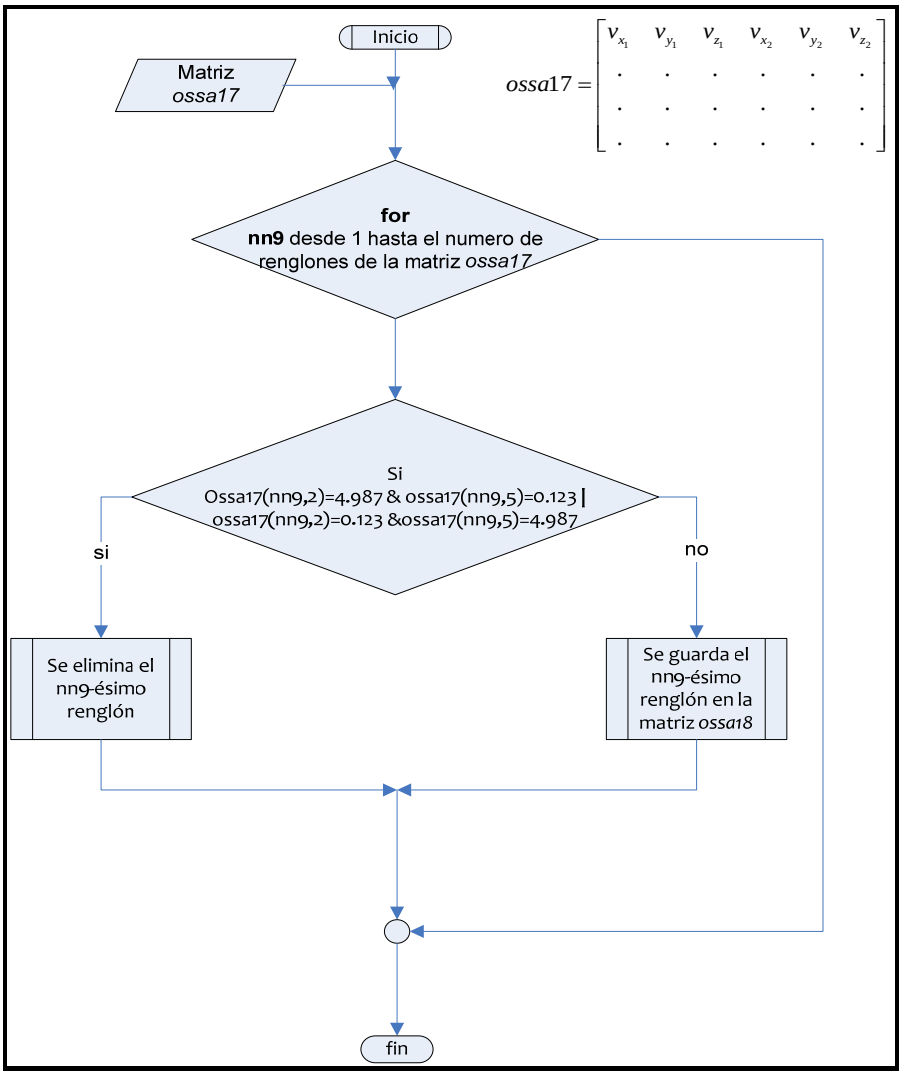


Figura III.34 Diagrama de flujo de la rutina para eliminar todas las vigas cumplan de las condiciones de la figura marcada con uno de la figura III.29b

Se puede observar que esta rutina es muy sencilla, contiene un ciclo "*for*" para recorrer todos los renglones de la matriz de entrada, en este caso la matriz *ossa17*, así como el condicional "*if*" para verificar que la coordenada **y** de los vértices de la viga que se analiza, no coincidan con los limites de corte en **y**, en caso de coincidir no se toma en cuenta el renglón y se pasa al siguiente, de lo contrario se guarda el renglón en una nueva matriz *ossa18*. Esta rutina es para eliminar a las vigas que cumplan con las condiciones de la viga uno, para las demás posibilidades se sigue la misma lógica del código. Con este conjunto de rutinas se termina de depurar los datos, con lo que se tiene la posibilidad de tomar el siguiente paso que es el darle el formato adecuado a los datos de salida que en este punto se tienen, este formato se refiere a la creación de dos matrices, una de ellas es la matriz *nodos*, la cual es una matriz de nx4, en la que

la primera columna de esta matriz se tiene el número de nodo y las tres columnas restantes son las coordenadas (x, y, z) de un vértice; la segunda es la matriz *elemento* la cual es de mx3 donde *m* es el numero de aristas, contiene tres columnas, la primera es el número de elemento o arista y las dos columnas restantes son los dos números de nodos que forman dicha arista, estos referidos a la matriz *nodos.* A continuación se muestra el diagrama de flujo dedicado a la creación de la matriz *nodos.*

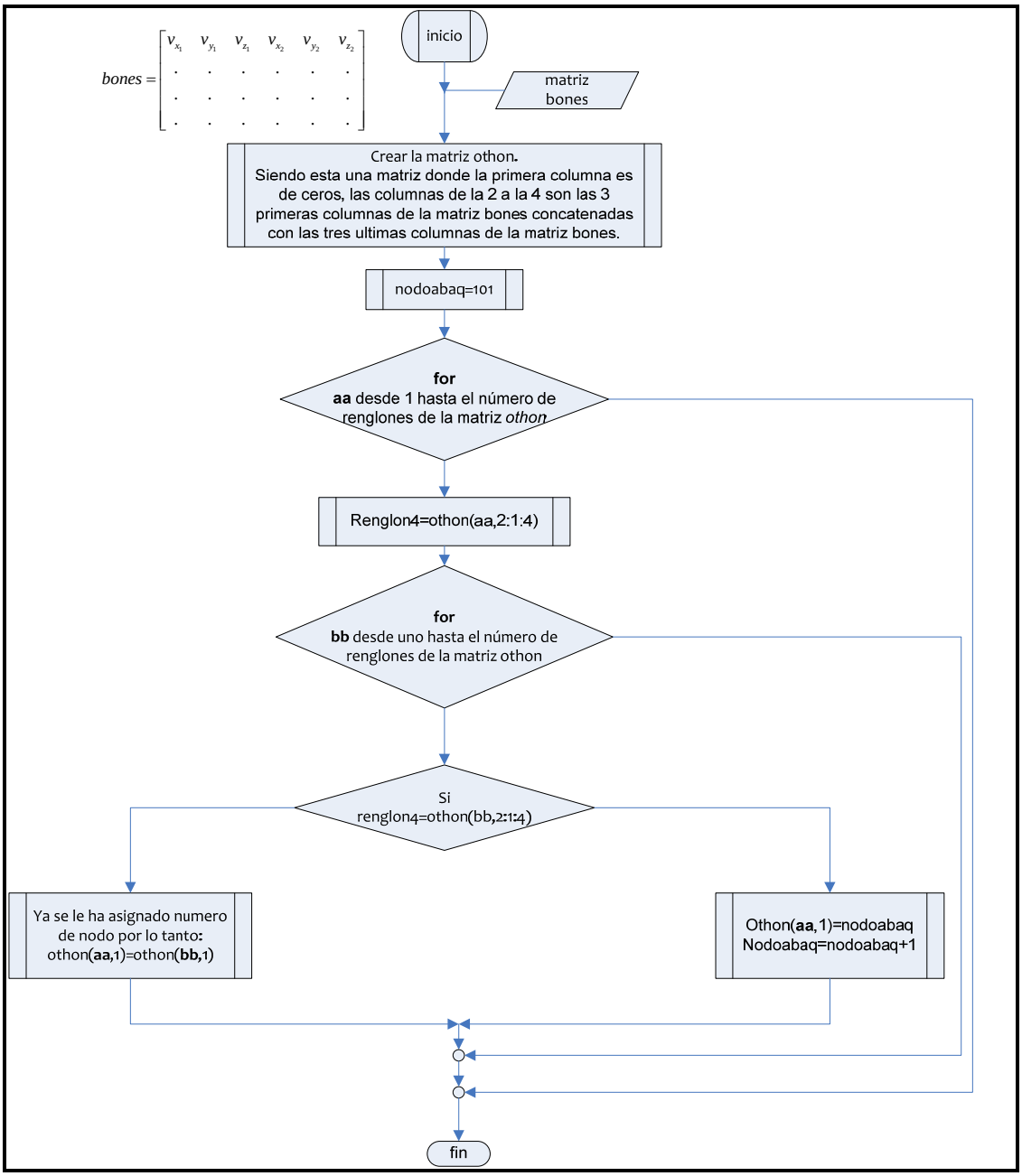


Figura III.35 Rutina para asignar número de nodo o vértice del arreglo Voronoi.

De este ciclo se puede observar que como dato de salida se tiene a la matriz *othon*, la cual tiene como primera columna el número de nodo asignado a cada uno de los vértices y las tres restantes son las coordenadas (x, y, z) de un vértice del arreglo. Esta matriz tiene los

elementos para ser la matriz *nodos*, sin embargo, al ser formada a partir de la matriz *bones* se tienen vértices repetidos, ya que cada vértice puede formar parte de más de una arista, por lo tanto para determinar a la matriz *nodos* se deben eliminar a los nodos renglones repetidos de la matriz, la rutina para eliminar renglones repetidos ya se usó y quedó representada en la figura III.27, de ella se modifican solo los datos de entrada y de salida siendo estos la matriz *othon* y *nodos* respectivamente, con esto se tiene ya una de las matrices necesarias para exportar los datos a la paquetería de elementos finitos ABAQUS®. La siguiente matriz importante es la matriz *elemento*, para determinar esta matriz se toma como apoyo a la matriz *othon* (dato de salida de la figura III.35), para lo cual se forma una nueva matriz siguiendo la rutina que se muestra a continuación.

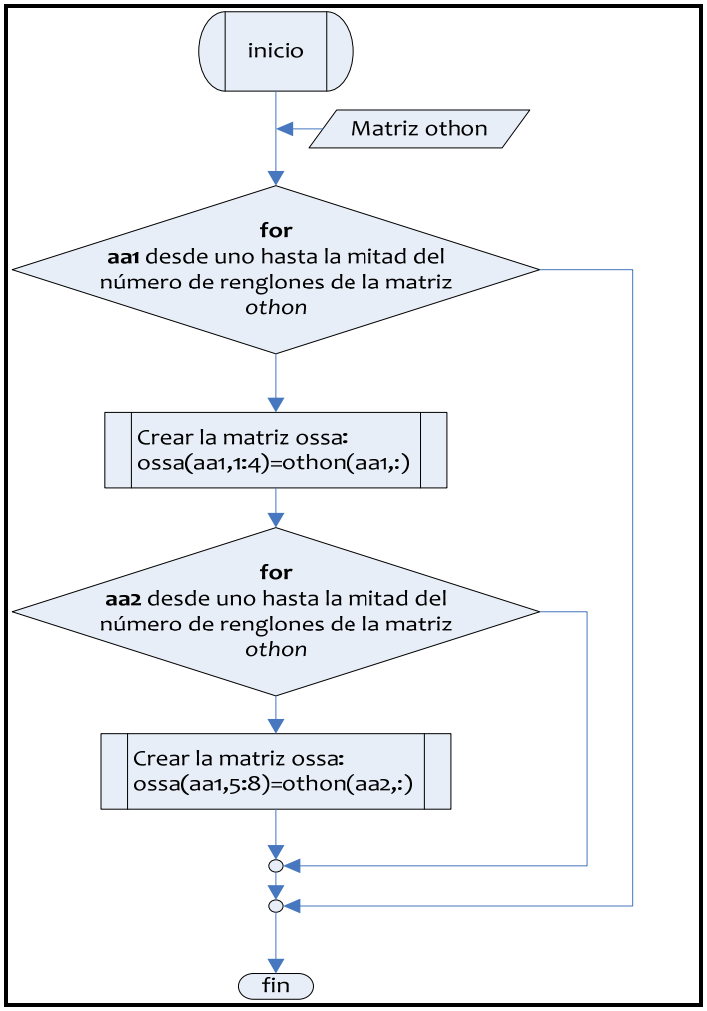


Figura III.36 Rutina para la creación de la matriz *ossa.*

De este ciclo, vale la pena resaltar que la matriz de salida *ossa* tiene la siguiente forma.

$$ossa = \begin{bmatrix} no.nodo & v_{x_1} & v_{y_1} & v_{z_1} & no.nod & v_{x_2} & v_{y_2} & v_{z_2} \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \end{bmatrix}$$

El siguiente paso para la determinación de la matriz *elemento* es formar a la matriz *coneccion* la cual se estructura de acuerdo a lo que se establece en la figura III.36

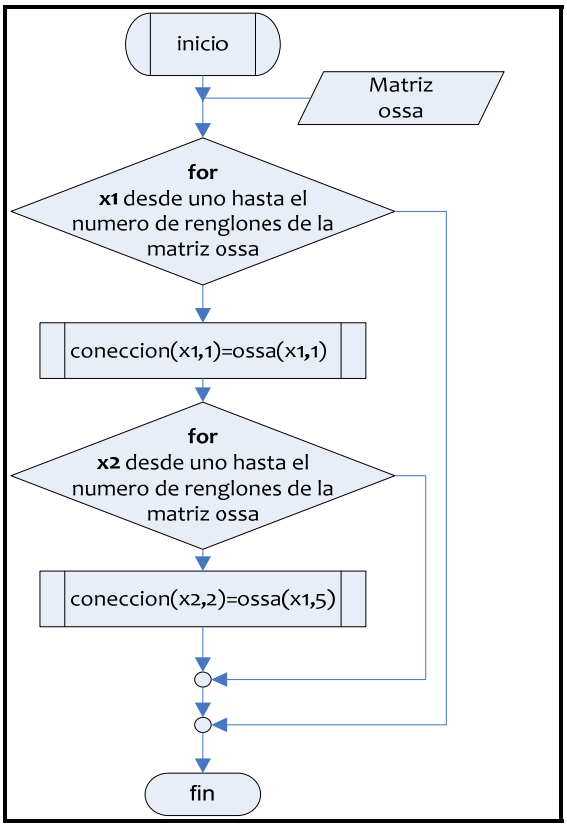


Figura III.37 Rutina para la determinación de la matriz *conección.*

Con esta rutina se tiene ya la relación de vértices tomando en cuenta su número de nodo de la matriz *nodos,* el siguiente paso es asignarle un número a cada elemento que equivale a asignarle un número a cada renglón. El proceso es muy sencillo y se representa en el siguiente diagrama (Figura III.38).

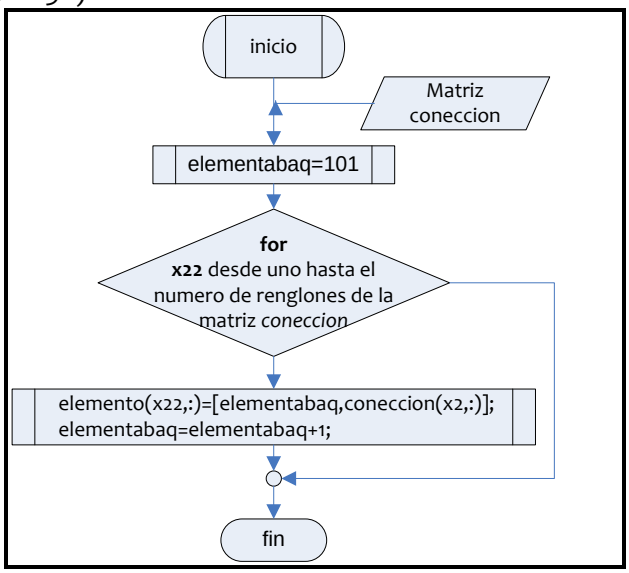


Figura III.38 Rutina para la determinación de la matriz *elemento.*

Los datos de salida que en este punto se tienen solo los siguientes.

$$nodos = \begin{bmatrix} no.nodo & v_{x_1} & v_{y_1} & v_{z_1} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix} \qquad elemento = \begin{bmatrix} no.elemento & no.nodo1 & no.nodo2 \\ . & . & . \\ . & . & . \\ . & . & . \end{bmatrix}$$

Estos datos servirán como piezas fundamentales en la formación del archivo *inp* que más adelante se explicará, otros datos que se requieren son los nodos que se encuentran en las fronteras de la muestra, debido que sobre ellos se aplicará condiciones de contorno o se determinarán datos específicos sobre su comportamiento. A continuación se presentará la rutina para determinar los nodos que encuentran en sobre el *plano superior* es decir z=4.987.

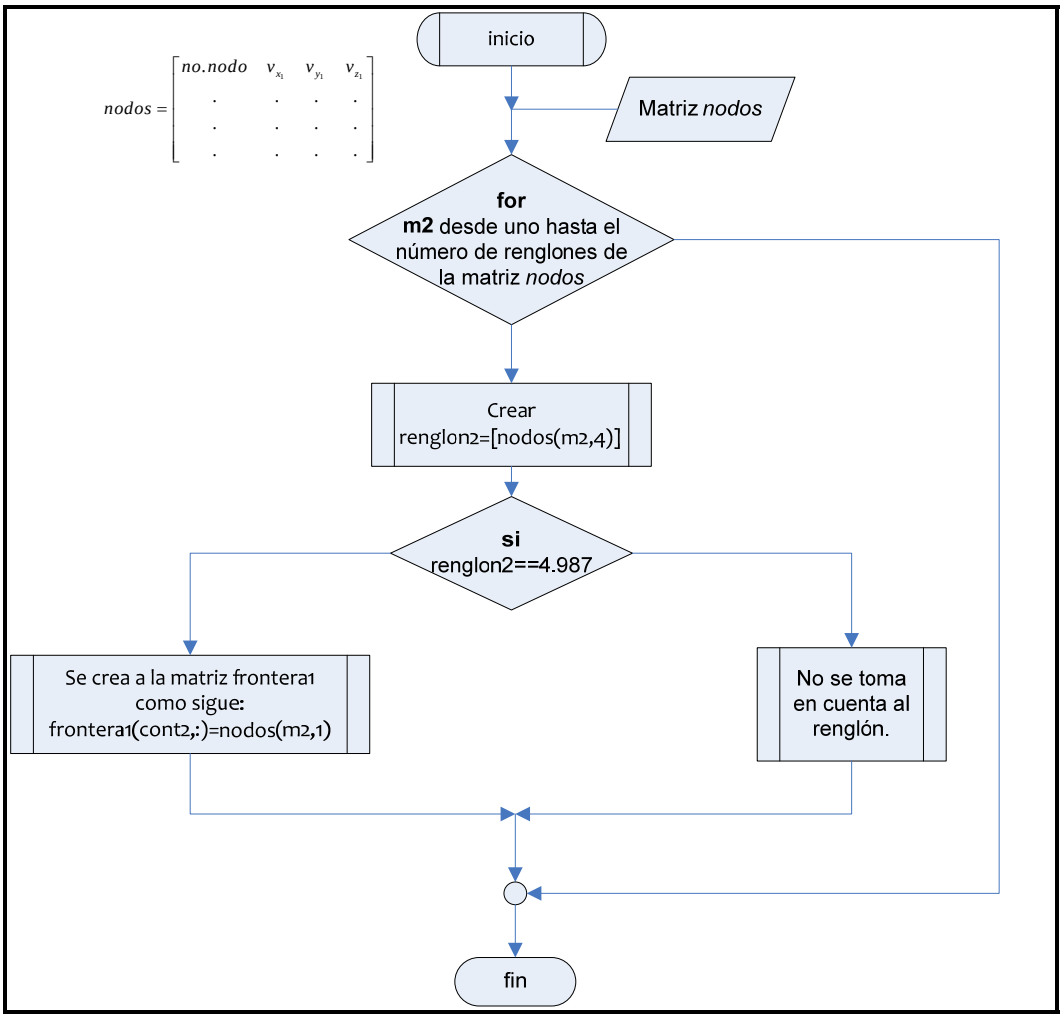


Figura III.39 Rutina para identificar en la matriz *frotera1* los nodos que se encuentran sobre el plano z=4.987

Con esta rutina se tiene ya, en una matriz, identificados los nodos que tienen la coordenada z=4.987, para identificar los nodos que están sobre los planos frontera se sigue la misma lógica, cambiando solo la condición por la adecuada en cada caso. A cada conjunto de

nodos que se encuentren sobre los planos frontera, se le asignó una matriz y nombre como se muestra a continuación.

Tabla III.1 Asignación de matrices a nodos frontera

Matriz	Plano sobre el que se encuentran los nodos
boundary	z=0.123
limiteinf_X	x=4.987
limitesup_X	x=0.123
limiteinf_Y	y=4.987
limitesup_Y	y=0.123

Recapitulando, en este punto se tienen ya definidas ocho matrices que se utilizaran como datos en la formación del archivo de entrada para la paquetería de ABAQUS® estas son: la matriz *nodos* que define y enumera todos los vértices que forman cada arreglo de Voronoi, la matriz *elemento* que relacione estos vértices por su número en la matriz nodos para definir las aristas del arreglo, la matriz *frontera1* que identifica a los nodos que se encuentran en el plano z=4.987 sobre los cuales se aplicara condiciones de desplazamiento, la matriz *Boundary* que identifica a los nodos que figuran como la base del arreglo y sobre los cuales se aplicaran condiciones restrictivas de desplazamiento, las matrices limiteinf_X, limitesup_X, limiteinf_Y y limitesup_Y que identifican a los nodos que pertenecen a las fronteras de la muestra (arreglo estructural cúbico de 4.864 de arista) y que se crean con el fin de conocer la deformación que sufre la estructura a través de los desplazamientos de estos nodos. Un dato importante de estos datos de salida es que se considera que las coordenadas están dadas en milímetros, por lo cual se tienen arreglos estructurales cúbicos de 4.864 [mm] de arista, esto es importante para el posterior trabajo de estos datos.

A continuación se presenta la descripción y el manejo de los comandos de MATLAB® para la inserción de las matrices de salida antes mencionadas, con el fin de la estructuración de un archivo de entrada *inp* para ABAQUS®.

III.4 Generación del archivo *inp* para ABAQUS®

En este punto se ha generado ya la geometría de la estructura de Voronoi mediante la paquetería de MATLAB® a manera de matrices, cuyos elementos contienen los nodos y relaciones de adyacencia entre ellos, lo cual permitirá definir una malla en la paquetería de análisis por elementos finitos. Para poder crear la comunicación entre MATLAB® y ABAQUS® se requiere generar un archivo de entrada *inp,* el cual contiene todos los datos que definen a la estructura de Voronoi en la paquetería de análisis por elementos finitos, este archivo tiene una estructura específica y bien definida, los parámetros y formato que utilizan se presentarán en el siguiente capítulo sin embargo se menciona en este momento con el fin de explicar el porque de la estructura tan concreta que se presentará a continuación. Para generar el archivo antes mencionado, siendo este en esencia un archivo de texto con la extensión *inp,* se utiliza en MATLAB® la función `dlmwrite`.

La instrucción `dlmwrite(archivo1,M)` imprime un archivo de texto con la matriz M. Para separar los elementos de la matriz mediante comas (,) es necesario utilizar el parámetro `'delimiter'` de la siguiente manera:

```
dlmwrite(archivo1, M, 'delimiter', ',')
```

Si ahora se quiere imprimir la matriz N en el mismo archivo sin sobrescribir el contenido del mismo, es necesario agregar el parámetro `'-append'`. Para poder definir un cierto número de líneas entre las matrices M y N se puede emplear el parámetro `'roffset'` de modo que si se quiere imprimir la matriz N dos líneas debajo de la matriz M se escribe la instrucción `dlmwrite` como sigue:

```
dlmwrite(archivo1, N, '-append', 'delimiter', ',', 'roffset', 2)
```

Para poder automatizar el proceso, será necesario exportar los datos de MATLAB® en un tipo de archivo con una extensión que sea compatible con ABAQUS®, dicha extensión es *.inp* en lugar de *.txt* del archivo de salida de MATLAB®. Para lograr esto, es necesario escribir el nombre completo del archivo incluyendo la extensión deseada, es decir:

```
dlmwrite(archivo1.inp, …)
```

La estructura de un archivo de entrada para ABAQUS® se estudiará más adelante, sin embargo, dichos archivos incluyen líneas de texto con instrucciones y parámetros que ABAQUS® emplea para crear la simulación. Para poder imprimir estas líneas de texto en el archivo *inp* es necesario crear arreglos de caracteres en MATLAB® mediante la instrucción `char()`. La instrucción `char('titulo')`genera un arreglo de caracteres de una sola línea con el texto "*titulo*". De modo que al escribir la instrucción:

```
intro = char('*Heading','Hueso Trabecular','Estructura de Voronoi');
```

se obtiene el siguiente arreglo:

```
intro =

*Heading
Hueso Trabecular
Caracterización de una estructura de Voronoi
```

De éste modo, es posible generar un archivo con líneas de texto y matrices en un formato y estructura compatible con ABAQUS®.

Parte de la programación empleada se muestra a continuación:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% AÑADIR TEXTO AL INP %%%%%%%%%%%%%%%%%%%%%%%%%%%%
intro = char('*Heading','Hueso Trabecular','Caracterización de una
estructura de Voronoi','*Preprint, echo=NO, model=NO, history=NO,
contact=NO')
elem = char('*Element, type=B33, elset=VORO1');
nset_b = char('*Nset, nset=EMPOTRADO');
nset_f = char('*Nset, nset=DESPLAZAMIENTO');
```

```
nset_xinf = char('*Nset, nset=X_inf');
nset_xsup = char('*Nset, nset=X_sup');
nset_yinf = char('*Nset, nset=Y_inf');
nset_ysup = char('*Nset, nset=Y_sup');
pivote = char('*Nset, nset=pivote');

%%%%%%%%%%%%%%%%%%%%%%%%   Exportar en formato INP  %%%%%%%%%%%%%%%%%%%%%%%%
dlmwrite(name_fin, intro,'')
dlmwrite(name_fin, nodos,'-append','roffset', 0, 'delimiter', ',')
dlmwrite(name_fin, elem,'-append','roffset', 1, 'delimiter', '')
dlmwrite(name_fin, elemento,'-append','roffset', 0, 'delimiter', ',')
dlmwrite(name_fin, section,'-append','roffset', 1, 'delimiter', '')
dlmwrite(name_fin, nset_b,'-append','roffset', 0, 'delimiter', '')
dlmwrite(name_fin, boundary,'-append','roffset', 0, 'delimiter', ',')
dlmwrite(name_fin, nset_f,'-append','roffset', 1, 'delimiter', '')
dlmwrite(name_fin, frontera,'-append','roffset', 0, 'delimiter', ',')
dlmwrite(name_fin, nset_xinf,'-append','roffset', 1, 'delimiter', '')
dlmwrite(name_fin, limiteinf_X,'-append','roffset', 0, 'delimiter', ',')
dlmwrite(name_fin, nset_xsup,'-append','roffset', 1, 'delimiter', '')
dlmwrite(name_fin, limitesup_X,'-append','roffset', 0, 'delimiter', ',')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

IV. ESTRUCTURA DEL ARCHIVO *INP* PARA ABAQUS®

El objetivo de éste capítulo es describir la estructura y generación de los archivos de entrada para ABAQUS® *inp* para simular ensayos de compresión y deslizamiento.

El archivo de entrada o *input file* es el medio de comunicación entre el preprocesador, por lo general ABAQUS/CAE®, y el software de análisis, ABAQUS/Standard o Explicit®. Este archivo contiene una descripción completa del modelo sujeto a simulación numérica. El archivo de entrada es un archivo de texto que tiene una interfaz intuitiva basada en el formato de KEYWORDS, por lo que es fácil de modificar con un editor de texto si es necesario, de hecho, los pequeños análisis pueden especificarse fácilmente tecleando directamente el archivo de entrada mediante un procesador de textos y cambiando la extensión *.txt* por *.inp*. Los KEYWORDS contienen la información necesaria para definir la malla, las propiedades del material, las condiciones de frontera y el control sobre las salidas del programa. Si se utiliza un preprocesador como ABAQUS / CAE® las modificaciones deben hacerse a través de éste, el cual, al finalizar el modelo, generará automáticamente el archivo *.inp* para su análisis.

A lo largo del presente capítulo, se explicará la estructura e instrucciones principales que se emplearon para realizar los ensayos en las estructuras de Voronoi.

IV.1 Sistema de unidades

Es importante que antes de comenzar a definir el modelo, ya sea a través de ABAQUS CAE® o tecleando los KEYWORDS mediante un procesador de textos, se establezca el sistema de unidades que se va a utilizar, ya que ABAQUS® no maneja unidades ni verifica consistencia en éstas. Durante el modelado, se debe tener cuidado de ser consistentes con las unidades que se manejen al definir la geometría, mallas, propiedades de los materiales, cargas o cualquier condición que se defina para crear un modelo. En la Tabla V.1 se muestran algunos de los sistemas de unidades más comunes. Cabe mencionar que los resultados deben ser analizados tomando en cuenta el sistema de unidades que se haya empleado, para evitar incongruencias en el análisis y conclusiones de este.

Tabla IV.1 Sistemas de unidades comunes

CANTIDAD	SI	SI (mm)	US (ft)	US (inch)
Longitud	m	mm	Ft	in
Fuerza	N	N	lb_f	lb_f
Masa	kg	ton (10^3 kg)	slug	lb_f*s^2 /in
Tiempo	s	s	s	s
Esfuerzo	Pa (N/m^2)	MPa (N/mm^2)	lbf/ft^2	psi (lbf/in^2)
Energía	J	mJ (10^{-3} J)	ft lbf	$In*lb_f$
Densidad	kg/m^3	ton/mm^3	$slug/ft^3$	lb_f*s^2/in^4

En el análisis realizado en este trabajo se utilizó el sistema internacional de unidades en milímetros, ya que las coordenadas de las muestras generadas en la paquetería de MATLAB® tienen dimensiones.

IV.2 Sistema de Coordenadas

El Sistema de Coordenadas que maneja por default ABAQUS® es un sistema cartesiano derecho como el que se muestra en la Figura V.1 Para los ensayos de compresión y deslizamiento se trabajó en tres dimensiones con este sistema derecho.

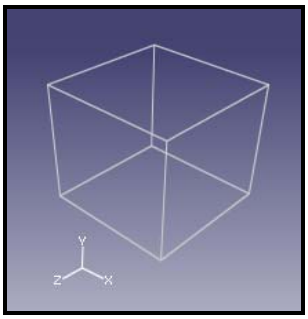


Figura IV.1 Sistema de Coordenadas de ABAQUS®

IV.3 Mallado

El método de elemento finito consiste en dividir el cuerpo, estructura o dominio (medio continuo) sobre el que están definidas las ecuaciones que caracterizan el comportamiento físico del problema, en una serie de subdominios no intersectantes entre sí denominados *elementos finitos*. El conjunto de elementos finitos forma una partición del dominio también denominada discretización. Dentro de cada elemento se distinguen una serie de puntos representativos llamados *nodos*. Dos nodos son adyacentes si pertenecen al mismo elemento finito. El conjunto de nodos considerando sus relaciones de adyacencia se llama *malla*.

Para generar la malla, es necesario elegir un elemento que se adecue a los requerimientos del modelo, tomando en cuenta las condiciones a las que va a estar sujeto y el tiempo de análisis que implica cada uno de ellos. Para el análisis realizado en este trabajo se eligió como elemento a las VIGAS, cuya principal característica es que son una aproximación a una dimensión de un continuo en tres dimensiones, esta reducción es el resultado de considerar a la sección transversal mucho más pequeña que la dimensión a lo largo del eje, esto le da a los elementos VIGA una simplicidad geométrica que es un factor importante en este

trabajo, dado que el arreglo estructural que se trabajo es complejo, por lo que al tomar estos elementos se reduce el tiempo de cómputo. Dentro de los elemento VIGA se tiene una amplia gama de diferentes tipos, los cuales pueden ser elegidos tomando en cuenta las siguientes características.

- Vigas de Euler-Bernoulli (B23, B23H, B33 y B33H). Este tipo no permite deformación cortante transversal; las secciones planas permanecen planas y normales al eje de la viga. Estas vigas son usadas solo para modelar (vigas esbeltas): las dimensiones de la sección transversal serán pequeñas comparadas con la distancia a lo largo de su eje. Las vigas de Euler-Bernoulli usan funciones de interpolación cúbica, con lo cual se obtienen buena precisión en los resultados, en especial en casos que involucran cargas distribuidas a lo largo de la viga.

- Vigas de Timoshenko (B21, B22, B31, B31OS, B32, B32OS, PIPE21, PIPE22, PIPE31, PIPE32). Estos elementos permiten deformación cortante transversal. ABAQUS® asume que el comportamiento cortante transversal de estas vigas es elástico lineal con un modulo fijo, y por lo tanto independiente de la respuesta de la sección de la viga a la deformada axial y a flexión.

Las vigas de Euler-Bernoulli son usadas para análisis con pequeñas deformaciones, mientras que las de Timoshenko se usan para grandes deformaciones axiales.

De esta gama de elementos VIGA se eligió para este trabajo a las vigas tipo B33, dado que éstas se utilizan generalmente para trabajos en tres dimensiones, para deformaciones pequeñas y tienen una interpolación cúbica que permite utilizar un solo elemento para cada uno de los miembros, lo cual es útil en este caso dado que los datos de salida obtenidos del programa desarrollado en MATLAB® son trabéculas definidas únicamente por dos nodos, por lo que cada trabécula será un elemento y no se tiene en este punto la posibilidad de determinar más nodos en las trabeculas.

A partir de las consideraciones anteriores los elementos que se trabajarán son los elemento BEAM B33 con lo cual se aproximan a las trabeculas del hueso como cilindros de sección transversal circular. La forma de declarar estos elementos en los archivo de entrada se describirá más adelante.

IV.4 Modelo

La estructura de un archivo *inp* en ABAQUS®, debe contar con toda la información que describe el problema y los resultados que se quieren obtener de éste. Dicha información debe contener al menos la geometría, el mallado, las propiedades de los materiales, tipo de análisis, cargas, condiciones de frontera y salidas del sistema.

Las instrucciones o KEYWORDS siempre van antecedidas de un asterisco (*) y precedidos por los parámetros de dicha instrucción. La instrucción y sus parámetros deben ser separados mediante el uso de comas. Por ejemplo:

```
*ELEMENT, TYPE=T2D2
```

*ELEMENT es la instrucción que define un tipo de elemento para conectar los nodos, mientras que TYPE=T2D2 indica el tipo de elemento a utilizar. Los KEYWORDS comúnmente van seguidos por líneas de información, las cuales, a diferencia de éstos, no requieren ningún tipo de símbolo al inicio, aunque sus elementos deben separarse mediante el uso de comas. En caso de requerir agregar información dentro del *inp* a modo de comentarios, debe colocarse dos asteriscos (**) al inicio de la línea.

IV.4.1 Heading

La primera instrucción en un archivo *inp* debe de ser el ***HEADING*** (o encabezado). En esta instrucción se puede agregar una breve descripción del modelo que permita identificar el *inp* posteriormente. En algunos casos, es recomendable agregar una descripción del sistema de unidades, propiedades o características que ayuden a entender el modelo. Para el caso particular del análisis de la estructura de Voronoi se estableció dicha instrucción de la siguiente forma:

```
*Heading
Hueso Trabecular
Caracterización de una estructura de Voronoi
```

IV.4.2 Impresión de datos

La instrucción *PREPRINT permite controlar la información que se desea imprimir en un archivo de salida con extensión .dat. Dicha instrucción presenta la siguiente estructura:

```
*PREPRINT, ECHO=YES, MODEL=YES, HISTORY=YES
```

En el caso particular de los ensayos sobre las estructuras de Voronoi, se dejaron todas las opciones activadas.

IV.4.3 Geometría

Una vez definido el título para el archivo *inp* y activadas las opciones de impresión de datos, se deben introducir los datos para describir la geometría del modelo. En esta sección se deben ingresar los nodos y elementos que conforman la malla. En caso de contar con una geometría compuesta de varias piezas, se deben introducir de manera separada asignándole un nombre a cada una de ellas mediante la instrucción *Part obedeciendo a la siguiente estructura:

```
*Part, name=pieza1
**Geometría
*End Part
```

Para luego ser ensambladas en la sección Assembly.

Gracias a que la geometría de la estructura de Voronoi fue generada a través de MATLAB® como una sola pieza, no fue necesario generar piezas que debieran ser ensambladas posteriormente, de modo que las coordenadas de los nodos se ingresaron directamente en la sección de Ensamble (*Assembly*).

Para este análisis en particular se tiene especial interés en conocer los niveles de esfuerzos en las vigas durante los ensayos, las fuerzas de reacción en los nodos donde se aplica el desplazamiento y los desplazamientos en las fronteras del cubo que se tiene de muestra, para lo cual se investigaron diversos tipos de requerimientos de salida en la sección "`output request`", para las fuerzas de reacción y desplazamientos de nodos no hubo mayor problema, sin embargo obtener los niveles de esfuerzo resultaba un problema, por lo que realizando varias modificaciones al archivo *inp* se observó que la única manera de obtener los niveles de esfuerzo fue generando una pieza sin geometría dentro de la sección `*Part` para luego agregar la instrucción *****instance dentro de la sección `Assembly`.

Los parámetros de la instrucción `*instance` son: el nombre del elemento (`name=instance1`) y la pieza a que hace referencia (`part=pieza1`).

Para poder asignar un nombre al ensamble se utiliza la instrucción `*Assembly`, cuyo parámetro es `name=Assembly1`. La línea de comando para definir la geometría del problema resulta de la siguiente forma:

```
** PARTS
*Part, name=alambre
*End Part
** ASSEMBLY
*Assembly, name=Assembly
*Instance, name=alambre-1, part=alambre
```

Para definir la malla, se debe utilizar la instrucción `*NODE` para ingresar un listado con el número y coordenadas de cada uno de los nodos que componen dicha malla. Esta lista debe iniciar en 101 seguido de las coordenadas (x, y, z) del primer nodo. El archivo de salida de MATLAB®, un archivo *inp,* contiene en uno de sus apartados la siguiente estructura donde se coloca instrucción nodo y a continuación se enumera los vértices de los cuales esta formada la estructura.

```
*Node
101,0.30272, 0.85190, 0.93839
102,0.16037, 0.87313, 1.06490
103,0.85156, 0.49795, 0.29212
104,...
```

Para definir el tipo de elemento a utilizar y la relación de adyacencia entre nodos, se utiliza la instrucción `*ELEMENT` cuyos datos deben ser ingresados a manera de lista bajo el formato que se muestra a continuación.

<número de elemento>, <nodo 1>, <nodo 2>

Donde *nodo 1* y *nodo 2* son los extremos del elemento.

El elemento BEAM B33 se asignó mediante el parámetro `type=B33` y se agruparon en un set (conjunto) mediante el parámetro `elset=VOR01`. El bloque completo de la instrucción `*ELEMENT` para asignar el tipo de elemento en los ensayos sobre la estructura de Voronoi se muestra a continuación:

```
*Element, type=B33, elset=VOR01
101,101, 102
102,101, 760
103,102, 760
104,...
```

Finalmente, se requiere asignar una sección y definir las propiedades del material del cual están constituidos los elementos asignados en la malla. Para el elemento B33 se debe utilizar la instrucción `*Beam General Section`, asignando en su parámetro `elset` el nombre del conjunto de elementos generado anteriormente. Las trabeculas para el modelado se aproximaron como cilindros de sección transversal circular cuyas características se muestran a continuación.

Tabla IV.2 Caracteristicas asignadas a los elemetos VIGA B33

CARACTERISTICAS PARA LOS ELEMENTOS B33	
Radio	0.08 [mm]
Coeficiente de Poisson	0.3
Modulo de Young	20000 [Mpa]
Modulo de rigidez a corte	7692.307[Mpa]

De las características anteriores es importante mencionar dos aspectos, el primero de ellos es que la propiedad geométrica que se le asigna, es decir el radio de las vigas, fue tomado de mediciones experimentales (Ramirez, 2007) donde este valor se encontró entre los grosores trabeculares más pequeños, esto se hace con el afán de analizar un caso extremo; el segundo aspecto son las propiedades mecánicas que se le asignas a las trabeculas, dichas propiedades son de hueso cortical, ya que si bien se esta considerando que el hueso esponjoso puede simplificarse como un arreglo de vigas (Gibson, 1985) también que se considera que dichas vigas son un material sólido, elástico e isotropico de propiedades similares a las de hueso cortical, esto con el fin de simplificar el análisis (Kim, 2002), a partir de lo anterior se le asignaron propiedades de hueso cortical reportadas en la literatura (Cuppone, 2004). La instrucción y los parámetros para asignar estas propiedades quedan como se muestra en el bloque de instrucciones siguiente.

```
*Beam General Section, elset=VOR01, poisson = 0.3, section=CIRC
0.08
0.,0.,-1.
20000.,7692.307
```

En el tercer renglón de esta sección de instrucciones se define la orientación de la sección transversal de las vigas, esta orientación es definida en términos de un eje local que sigue la regla de la mano derecha (t, n1, n2) donde t es la tangente sobre el eje del elemento, positiva desde el primer al segundo nodo; n1 y n2 son vectores que definen las direcciones locales 1 y 2 de la sección, como se muestra en la figura V.2

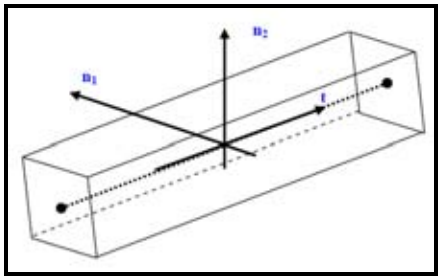


Figura IV.2 Definición del eje local.

Una vez definida la geometría, elementos, sección y propiedades de la malla, se debe finalizar la instrucción `*instance` mediante el comando `*End instance`. Antes de finalizar la sección de ensamble, se definieron conjuntos de nodos pertenecientes a las caras de la estructura de Voronoi, sobre las cuales se van a establecer las condiciones de frontera (BASE y DESPLAZAMIENTO), también es importante determinar estos conjuntos de nodos debido a que se extraerán de ellos los desplazamientos y reacciones que se presenten ante las condiciones impuestas. Para generar estos "*sets*" (conjuntos) de nodos, se debe utilizar la instrucción `*Nset` cuyo parámetro es: `nset=name`, y las líneas de información para dicha función es una lista con los números de los nodos que se desean agregar al set. Los números de nodo deben ir separados por comas y el número de nodos por línea no debe ser mayor a quince. Los sets de nodos que se crearon y la sintaxis utilizada es la siguiente:

```
*Nset, nset=BASE
761, 766, 769 …
*Nset, nset=DESPLAZAMIENTO
1074, 1075, 1076 …
*Nset, nset=X_inf
837, 841, 842 …
*Nset, nset=X_sup
764, 765, 767 …
*Nset, nset=Y_inf
807, 811, 812 …
*Nset, nset=Y_sup
762, 763, 775 …
*Nset, nset=pivote
770
*End Assembly
```

Los *sets* creados corresponden a los nodos que se encuentran en los diferentes planos de corte, el *set pivote* es un caso especial dado que solo contiene un nodo, el cual es extraído del set empotrado y que sirve como apoyo en los ensayos de compresión y deslizamiento. La tabla siguiente muestra a que plano pertenecen los diferentes sets.

Tabla IV.3 Sets y plano al que pertencen los nodos de dichos conjunto.

Set	Plano de corte
EMPOTRADO	z=0.123
DESPLAZAMIENTO	z=4.987
x_inf	x=4.987
x_sup	x=0.123
y_inf	y=4.123
y_sup	y=0.123

IV.4.4 Propiedades del material

En ésta sección se definen las propiedades de los materiales de las piezas involucradas en el análisis. Para definir las propiedades de un material, se emplea la instrucción `*Material` cuyo parámetro es: `name=material1`. En las líneas de información de dicha función, se debe especificar el tipo de propiedades que se va a definir ya que en algunos casos es necesario establecer propiedades térmicas, elásticas, plásticas, eléctricas, entre otras, de acuerdo a los modelos que correspondan. ABAQUS® cuenta con una amplia gama de modelos que incluye algunos no lineales. Para éste caso se utilizó el modelo de elasticidad lineal.

IV.4.5 Perfil del análisis

El objetivo de esta sección dentro del *inp* es definir los diferentes eventos en que se desarrollará la simulación, así como definir el tipo de análisis que se desea realizar y las condiciones de carga y frontera a las que estará sujeta la estructura. Para este caso, se busca simular el desplazamiento unidireccional de una de las caras de la celda de Voronoi, este proceso solo requiere de un evento y debe definirse mediante el comando *****STEP, cuyo único parámetro será el nombre del mismo mediante el comando "name" antes empleado para otros comandos. Luego de haber definido el evento para la simulación, es necesario especificar el tipo de análisis que se desea realizar, los más comunes son: *estático y dinámico*. En este caso, lo que se desea obtener es la respuesta estática en estado permanente de la estructura bajo las condiciones de desplazamiento, el comando para definir un análisis estático es `*STATIC` y las líneas de información para este comando definen el tiempo de duración del evento y los incrementos de tiempo para el análisis.

Los comandos empleados para definir el evento y tipo de análisis a realizar para la caracterización de la estructura de Voronoi, con sus parámetros y líneas de información se muestran a continuación:

```
*Step, name=paso-1
*Static
0.1, 1.0, 1e-05, 0.1
```

Los siguientes datos que se deben introducir en la sección de `*STEP` son: condiciones de frontera, cargas y salidas del sistema, y pueden ser ingresados en cualquier orden.

En este caso se comenzará por definir las condiciones de frontera las cuales se utilizan para restringir movimiento en los nodos o a definir un desplazamiento conocido sobre ellos. En algunos casos, se utilizan para definir simetría en el modelo. Dichas condiciones, deben ser introducidas mediante el comando *BOUNDARY y especificando el nodo (o set de nodos) sobre el cual se va a aplicar, el número correspondiente al grado de libertad que se desea restringir y el desplazamiento. Es decir:

```
*Boundary
<nodo>, <grado de libertad>, <desplazamiento>
```

La convención de enumeración utilizada para los grados de libertad en ABAQUS® es la siguiente:

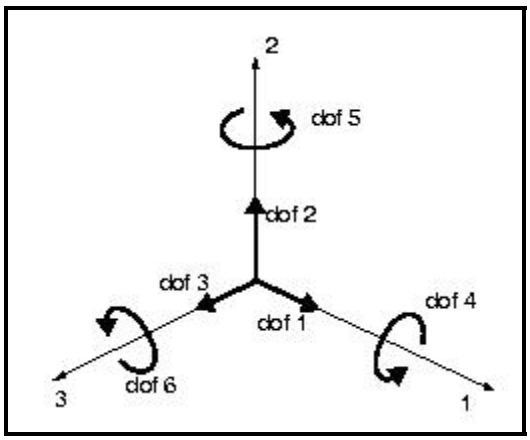


Figura Iv.4 Convención de enumeración para los grados de libertad.

En lugar de especificar cada grado limitado de libertad, algunos de los más comunes se pueden ingresar directamente usando los siguientes comandos:

Tabla IV.4 Comandos para la restricción de grados de libertad en ABAQUS®

KEYWORD	FUNCIÓN
ENCASTRE	*Restringe todos los desplazamientos y rotaciones en un nodo*
PINNED	*Restricción en todos los grados de libertad traslacionales*
XSYMM	*Simetría sobre un plano perpendicular a x*
YSYMM	*Simetría sobre un plano perpendicular a y*
ZSYMM	*Simetría sobre un plano perpendicular a z*
XASYMM	*Anti-simetría sobre un plano perpendicular a x1*
YASYMM	*Anti-simetría sobre un plano perpendicular a y1*
ZASYMM	*Anti-simetría sobre un plano perpendicular a z1*

Las condiciones de frontera que se utilizaron para la estructura de Voronoi dependiendo del ensayo que se simuló fueron:

- **COMPRESIÓN.** Desplazamiento de 0.5 milímetros (corresponde aproximadamente al 10% de la altura de la muestra) en el sentido negativo de la dirección 3 para el *set DESPLAZAMIENTO*, restricción de movimiento en dirección 3 para los nodos en la base de la estructura (*set EMPOTRADO*) y empotrado para el nodo central de la base con el fin de evitar que la estructura gire. Las líneas de comando son las siguientes:

```
*Boundary
DESPLAZAMIENTO, 3, -0.5
*Boundary
EMPOTRADO, 3, 0
*Boundary
pivote, encastre
```

DESLIZAMIENTO. Desplazamiento de 0.5 milímetros en el sentido negativo de la dirección 2 para el *set DESPLAZAMIENTO*, restricción de movimiento en las direcciones 2 y 3 para los nodos en la base de la estructura (*set EMPOTRADO*) y empotramiento para el nodo central de la base con el fin de evitar que la estructura gire. Las líneas de comando son las siguientes:

```
*Boundary
DESPLAZAMIENTO, 2, -0.5
*Boundary
EMPOTRADO, 2, 3, 0
*Boundary
pivote, encastre
```

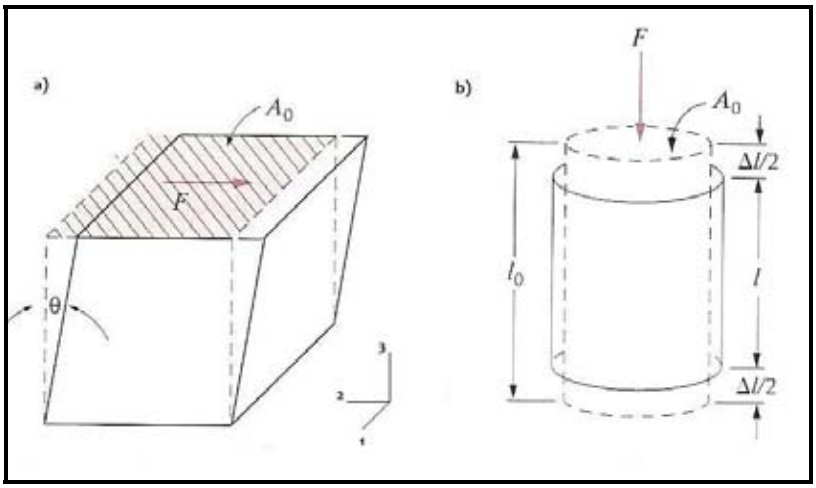


Figura IV.5 a) Ensayo de deslizamiento . b) Ensayo de compresión (Callister, 2004)

Los ensayos se definieron en dos archivos *inp* diferentes. Para los fines de este trabajo, no fue necesario definir ningún tipo de cargas, presiones o temperaturas, de modo que solo resta definir las salidas que desean obtener. El comando mediante el cual se puede definir el tipo de salida que se requiere es *OUTPUT y sus parámetros son: *history* y *field.* El parámetro field se emplea para generar gráficas de contorno o gráficas de deformación de superficies, mientras que el parámetro history se emplea para variables que se desean observar mediante una gráfica de tipo X-Y.

Para éste modelo, se utilizaron los parámetros de salida predefinidos, empleando el parámetro variable=preselect para ambos tipos de respuesta. Las líneas de comando y sintaxis se muestran a continuación:

```
** OUTPUT REQUESTS
**
*Output, field, variable=PRESELECT
**
*Output, history, variable=PRESELECT
```

Una vez que se han definido todos los datos necesarios para el evento, se utiliza el comando *END STEP para marcar el final del evento.

Con esto se tiene ya, definido las estructuras de dos archivos *inp* para simular dos ensayos mecánicos a las muestras virtuales cúbicas de 4.864 [mm] de arista generadas en MATLAB®, dichos ensayos son: compresión y deslizamiento, mostrados en la figura IV.5, con los cuales se busca determinar las propiedades mecánicas de las estructuras generadas, para ello se crearon, con base en la estructura del archivo *inp* mostrada anteriormente, treinta ensayos de compresión y deslizamiento distribuidos de la siguiente forma.

Tabla IV.5 Número de simulaciones realizadas para cada base.

BASE	Distubación (Desviación Estandar)			
	0	0.1	0.2	0.3
Cúbico simple	1	3	3	3
Cúbico centrado en las caras	1	3	3	3
Cúbico centrado en el cuerpo	1	3	3	3

El siguiente capitulo presenta los resultados de los ensayos realizados a través de esta estructura del *input file.*

V. PROPIEDADES MECÁNICAS PARA LAS PROBETAS VIRTUALES

El presente capítulo tiene por objetivo presentar las propiedades mecánicas determinadas para las treinta probetas virtuales creadas en MATLAB® y simuladas en ABAQUS®. Se busca también a través de este capítulo mostrar la forma en que se determinaron dichas propiedades.

V.1 Módulo de elasticidad

El grado con que una estructura se deforma depende de la magnitud de la solicitación impuesto. Para muchos materiales sometidos a cargas de tracción pequeñas, el esfuerzo y la deformación son proporcionales según la relación:

$$\sigma = E\varepsilon$$

Esta relación se conoce con el nombre de ley de Hooke, y la constante de proporcionalidad, *E* es el **módulo de elasticidad,** o módulo de Young.

La determinación del módulo de elasticidad para las probetas es de gran importancia debido a que éste es una medida de la rigidez del material, en este caso, de la estructura (módulo aparente).

Esta propiedad se determinó, para las probetas virtuales, mediante las simulaciones de ensayos de compresión uniaxial en la paquetería de ABAQUS®. Para esto, se obtuvieron las reacciones cada 0.1 [s] sobre los nodos de la cara superior de la estructura, que al sumarse corresponden a la fuerza de reacción resultante. Se sabe que el esfuerzo se define como:

$$\sigma = F \big/ A$$

En donde A, corresponde al área no deformada en la cual se aplica la fuerza, que para el caso de las probetas corresponde al área de la cara superior.

Para determinar las deformaciones, se obtuvo el cociente entre los desplazamientos cada 0.1 [s] que corresponden a 0.05 [mm] (ya que la simulación transcurre en 1 [s] y el desplazamiento total es de 0.5 [mm]) y la altura de la probeta.

Finalmente, se realizó una regresión lineal con estos datos a través de la paquetería Excel® para obtener la pendiente de la recta σ-ε que corresponde al módulo de elasticidad estructural. En la tabla VI.1 se muestra el módulo de elasticidad para las desviaciones estándar programadas en cada una de las estructuras.

Tabla V.1 Modulo de elasticidad promedio

MODULO DE ELASTICIDAD [MPa]			
	CUBICO SIMPE	**CUBICO CENTRADO EN EL CUERPO**	**CUBICO CENTRADO EN LAS CARAS**
0	413.3667763	231.703125	303.1751645
0.1	318.9712171	226.2911184	379.7171053
0.1	343.6044408	226.0699013	382.984375
0.1	336.5633224	225.3717105	374.5287829
0.2	157.5756579	214.2401316	331.9078947
0.2	153.6340461	213.5411184	333.4555921
0.2	157.5756579	216.7648026	333.4555921
0.3	97.72944079	217.3758224	301.0238487
0.3	98.02302632	217.8486842	299.74
0.3	97.80838816	217.3758224	300.32

Debido a que las propiedades asignadas a las vigas que componen el arreglo estructural de Voronoi de cada una de las muestras siguen un comportamiento lineal y además siendo una deformación pequeña la que se le aplica a las muestras se observa que estas siguen un comportamiento lineal. A continuación se presentan las curvas esfuerzo-deformación de tres probetas, para cada una de las distribuciones que se trabajaron (cúbico simple, cúbico centrado en las caras y cúbico centrado en el cuerpo) con una desviación estándar de 0.1 en las perturbaciones, ya que para todas las muestras se presenta el mismo comportamiento lineal en las gráficas no se muestran debido al espacio que ocuparían y siendo el modulo de elasticidad el parámetro de mayor importancia en este comportamiento el cual ya fue presentado en la tabla anterior.

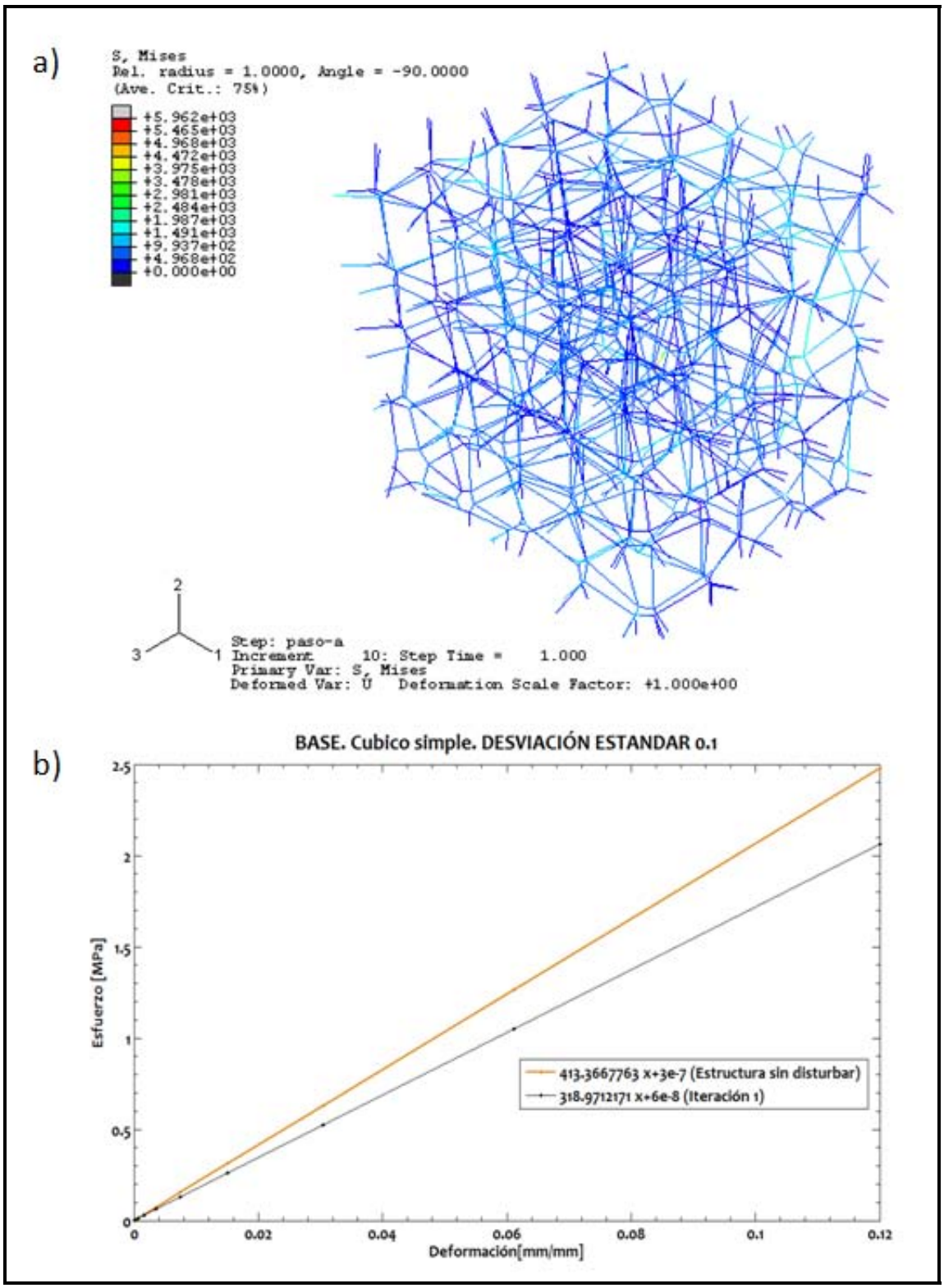


Figura V. 1 a) Nivel de esfuerzos después de una deformación 0.1027 [mm/mm] a una estructura de Voronoi con los nodos distribuidos en un patrón de estructura Cúbica Simple con una desviación estándar de 0.1 en la perturbación, b) Curva esfuerzo esfuerzo-deformación de la misma estructura en forma comparativa con la estructura sin perturbar.

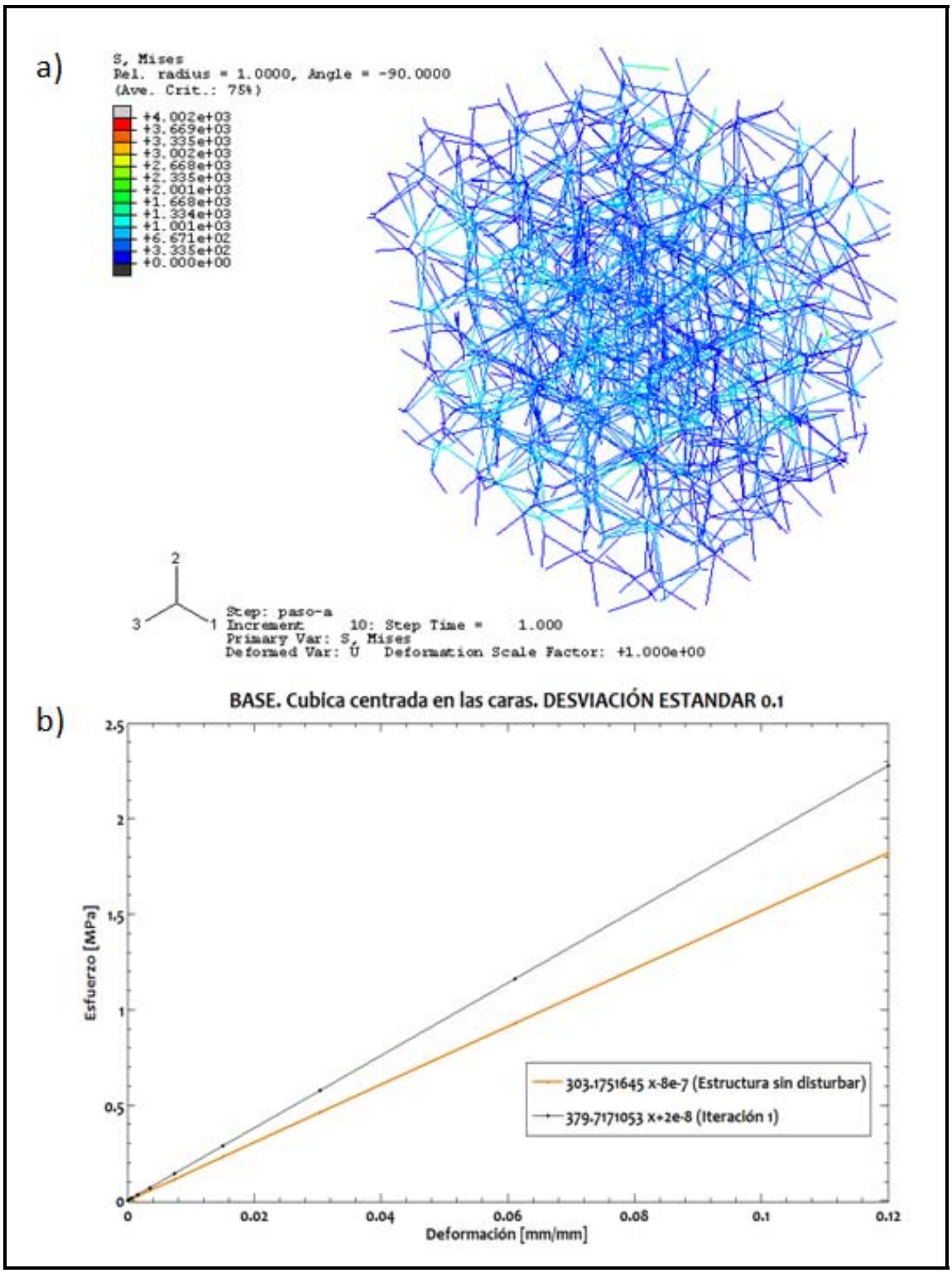


Figura V. 2 a) Nivel de esfuerzos después de una deformación 0.1027 [mm/mm] a una estructura de Voronoi con los nodos distribuidos en un patrón de estructura Cúbica Centrada en las Caras con una desviación estándar de 0.1 en la perturbación, b) Curva esfuerzo esfuerzo-deformación de la misma estructura en forma comparativa con la estructura sin perturbar.

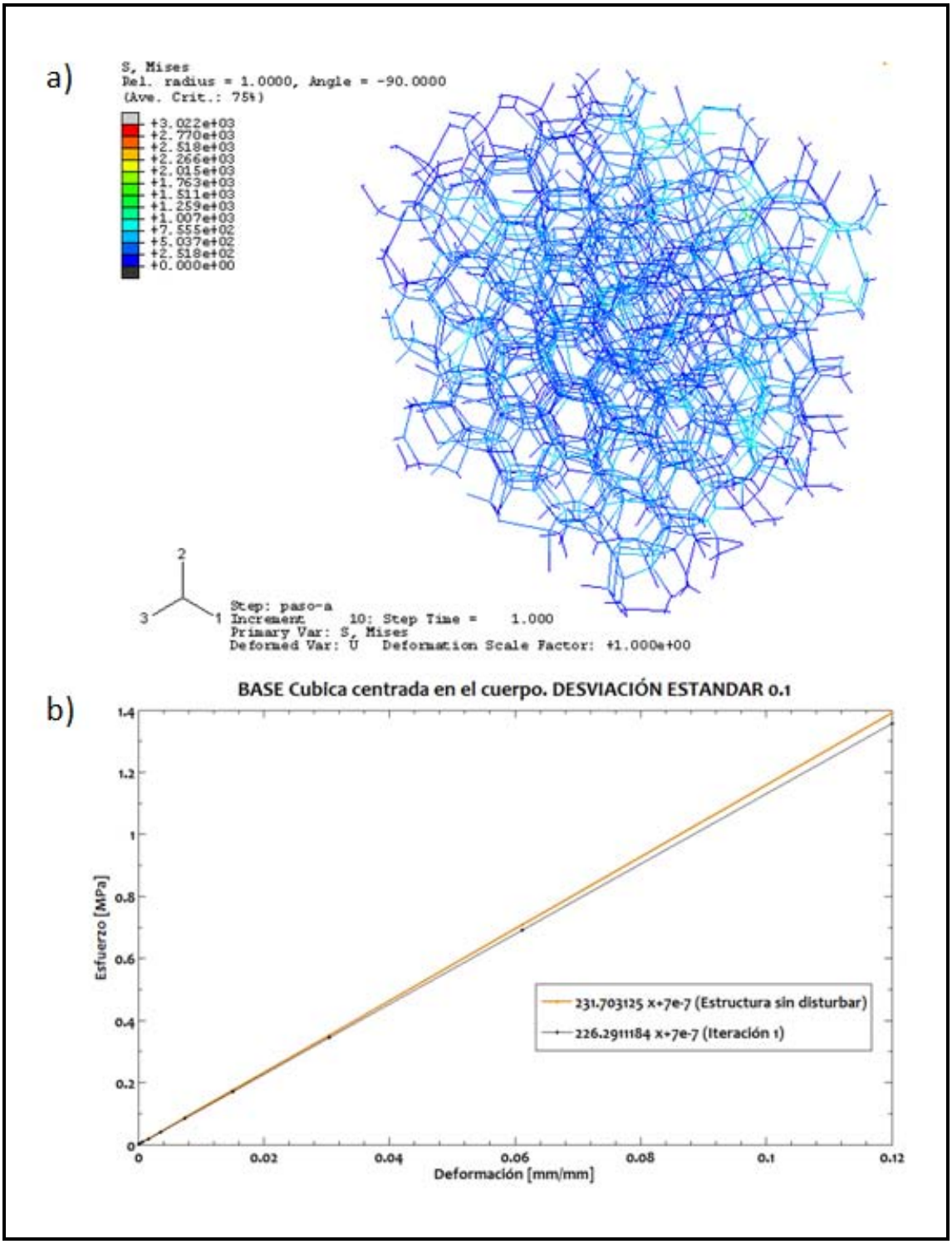


FIgura V. 3 a) Nivel de esfuerzos después de una deformación 0.1027 [mm/mm] a una estructura de Voronoi con los nodos distribuidos en un patrón de estructura Cúbica Centrada en el Cuerpo con una desviación estándar de 0.1 en la perturbación, b) Curva esfuerzo esfuerzo-deformación de la misma estructura en forma comparativa con la estructura sin perturbar.

V.2 Módulo de Rigidez a corte

Cuando una fuerza *F* que actúa sobre el cuerpo es paralela a una de las caras mientras que la otra cara permanece fija, se presenta otro tipo de deformación denominada de cizallamiento o cortante en el que no hay cambio de volumen pero si de forma. Si originalmente la sección transversal del cuerpo tiene forma rectangular, bajo un esfuerzo cortante se convierte en un paralelogramo.

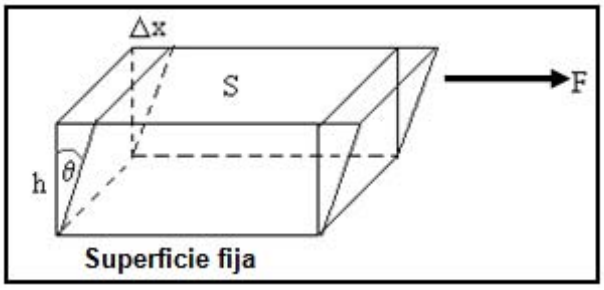


Figura V.4 Esfuerzo cortante

Se define el esfuerzo como *F*/*S*, la razón entre la fuerza tangencial y el área S no deformada de la cara sobre la que se aplica. La deformación por cizalla o cortante, se define como la razón Δ*x*/*h*, donde Δ*x* es la distancia horizontal que se desplaza la cara sobre la que se aplica la fuerza y *h* la altura del cuerpo, tal como se observa en la figura V.4 El módulo de rigidez *G* es una propiedad mecánica de cada material, siendo dicha propiedad la rigidez de un cuerpo ante esfuerzos cortantes.

Considerando de pequeños ángulos de desplazamiento se define lo siguiente.

$$deformación\ cortante = \frac{\Delta x}{h}$$

$$G = \frac{esfuerzo}{deformación} = \frac{F/A}{\Delta x/h} \qquad (1)$$

El módulo de rigidez para las probetas se calcularon mediante ensayos de deslizamiento donde se aplico una desplazamiento cortante a los nodos de las cara superior de la muestra simulando la fuerza cortante. Se extrajeron de ABAQUS® las reacciones en la dirección del deslizamiento (0.5 [mm] en la dirección 1) y mediante el cociente entre éstas y el área de la cara superior de la probeta se obtuvo el esfuerzo cortante.

La deformación se obtuvo mediante el cociente entre los desplazamientos de 0.05 [mm] cada 0.1 [s] y la altura de la probeta.

Finalmente, el módulo de rigidez para cada probeta se obtuvo mediante la aplicación de la ecuación 1. En la tabla V.2 se muestran los resultados del módulo de rigidez para cada desviación estándar modelada.

Tabla V.2 Módulos de Rigidez a corte

MODULO DE RIGIDEZ A CORTE [MPa]			
	CUBICO SIMPE	CUBICO CENTRADO EN EL CUERPO	CUBICO CENTRADO EN LAS CARAS
0	102.7960526	27.97746711	18.91362116
0.1	12.99497199	30.6924301	45.28425899
0.1	12.67435704	30.65395066	43.91319131
0.1	12.81933846	30.52196664	49.09129112
0.2	15.95822218	28.16035773	43.12023914
0.2	15.96259268	28.03243062	43.20668468
0.2	15.96368649	26.58635554	43.20668468
0.3	12.12968703	27.73878319	41.4786049
0.3	12.20499147	27.78596837	41.4746049
0.3	13.29678627	27.73878319	41.4706049

V.3 Coeficiente de Poisson

Cuando un material se somete a un esfuerzo de compresión, se produce una deformación ε_{33} en la dirección de la carga aplicada. Como resultado, se producirá un aumento en las direcciones laterales perpendiculares a la aplicación de la carga.

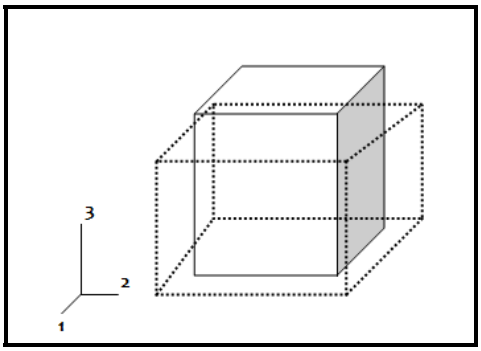


Figura V.5 Deformación en un cubo al aplicarse una fuerza de compresión en la dirección 3

Con esto, se pueden determinar las deformaciones de compresión ε_{11} y ε_{22}, que en un material cúbico son simétricas. Se define entonces el coeficiente de Poisson como se muestra:

$$\nu = \frac{\varepsilon_{11}}{\varepsilon_{22}}$$

Debido a que en las probetas virtuales las deformaciones ε_{11} y ε_{22} no son simétricas, se calculó el promedio de estas para determinar el coeficiente de Poisson expresado a continuación:

$$\nu = \frac{1}{2}\left(\frac{\varepsilon_{11} + \varepsilon_{22}}{\varepsilon_{33}}\right)$$

Tabla VI.3Coeficientes de Poisson

COEFICIENTE DE POISSON			
	CUBICO SIMPE	CUBICO CENTRADO EN EL CUERPO	CUBICO CENTRADO EN LAS CARAS
0	0.381556584	0.30513934	0.225620985
0.1	0.261685388	0.316587818	0.240694132
0.1	0.195621362	0.317474795	0.238845854
0.1	0.203114194	0.317760342	0.227481477
0.2	0.366087415	0.271934062	0.261530499
0.2	0.301520596	0.277824791	0.265127804
0.2	0.368141249	0.32140135	0.265127804
0.3	0.282012198	0.270412998	0.267761021
0.3	0.268721344	0.271864466	0.27394966
0.3	0.375077794	0.270445486	0.267761021

V.2 Coeficientes Anisotrópicos

Como ya se mencionó, las propiedades elásticas del hueso trabecular varían según la dirección en que se aplica la carga, es decir, se trata de un material anisotrópico. Ésto debido al crecimiento irregular natural de las trabéculas del hueso esponjoso. Ahora bien, las probetas que se analizaron a lo largo del presente trabajo parten de estructuras cúbicas que mediante la aplicación ciertos valores de desviación estándar, se distorsionan para aproximar la estructura irregular natural del hueso. Por esto, se definieron dos coeficientes que nos permitirán conocer el grado de anisotropía que presentan las estructuras de Voronoi.

En un material isotrópico, al aplicar una deformación en la dirección principal 3, el material sufrirá deformaciones en las direcciones principales 1 y 2, ambas de de igual magnitud. Para un material anisotrópico esto no se cumple, por lo tanto, se define un coeficiente anisotrópico δ como la diferencia entre las deformaciones ε_{11} y ε_{22} como se muestra a continuación:

$$\delta = \frac{\varepsilon_{11} - \varepsilon_{22}}{\varepsilon_{33}}$$

Por otro lado, en un ensayo de compresión uniaxial sobre un material anisotrópico, se generan deformaciones cortantes que generan rotaciones en la estructura, para medir éste efecto, se define el factor anisotrópico ξ de la siguiente manera:

$$\xi = \frac{\varepsilon_{12}}{\varepsilon_{33}}$$

Los valores de ε_{11} y ε_{22} se obtuvieron anteriormente, el valor de ε_{12} se define de la siguiente manera:

$$\varepsilon_{12} = \frac{1}{2}\left(\frac{\partial U_1}{\partial x_2} + \frac{\partial U_2}{\partial x_1}\right)$$

donde *U* es el desplazamiento.

Tabla V.4 Coeficiente anisotrópico δ

δ			
	CUBICO SIMPE	**CUBICO CENTRADO EN EL CUERPO**	**CUBICO CENTRADO EN LAS CARAS**
0	0.763113153	-0.002331334	-0.00538336
0.1	-0.024627686	0.014382764	0.00858897
0.1	0.028213435	0.014545918	0.011202031
0.1	0.013659566	0.009947497	-0.004634646
0.2	-0.021226372	0.040386136	-0.003636714
0.2	-0.02871114	0.041298457	-0.002091612
0.2	-0.021698622	0.021512916	-0.002091612
0.3	0.000324812	0.065165785	-0.036540709
0.3	-0.00936484	0.067246174	-0.038868485
0.3	0.037187859	0.06510081	-0.039006616

Tabla V.5 Coeficiente anisotrópico ζ

ζ			
	CUBICO SIMPE	**CUBICO CENTRADO EN EL CUERPO**	**CUBICO CENTRADO EN LAS CARAS**
0	0.381556584	0.30513934	0.225620985
0.1	0.261685388	0.316587818	0.240694132
0.1	0.195621362	0.317474795	0.238845854
0.1	0.203114194	0.317760342	0.227481477
0.2	0.366087415	0.271934062	0.261530499
0.2	0.301520596	0.277824791	0.265127804
0.2	0.368141249	0.32140135	0.265127804
0.3	0.282012198	0.270412998	0.265718866
0.3	0.268721344	0.271864466	0.270289874
0.3	0.375077794	0.270445486	0.272263629

VI. ANÁLISIS DE RESULTADOS

Los ensayos de compresión y deslizamiento se realizaron para 3 bases de dispersión de puntos: cúbico simple, cúbico centrado en las caras y cúbico centrado en el cuerpo, las cuales fueron distorsionadas a través del desplazamiento de la distribución de puntos, determinando para cada base 9 muestras, con lo cual se trabajaron con 27 muestras distorsionadas y 3 sin distorsionar. A través de los ensayos mecánicos virtuales se determinaron las propiedades mecánicas de cada una ellas. El objetivo de este capítulo es analizar los resultados obtenidos a través de dichas pruebas, asimismo se busca observar el comportamiento que tienen las propiedades mecánicas con respecto al grado de perturbación y la base con la cual fue generado el arreglo de Voronoi.

VI.1 Módulo de elasticidad (E)

Para la determinación del módulo de elasticidad, como ya se mencionó, se trabajo con una probeta virtual cúbica de 4.864 [mm] de arista, la cual fue sometida a una deformación de 0.1027 [mm/mm], la deformación se obtuvo con la imposición de un desplazamiento aplicado a los nodos de la parte superior del cubo correspondiente a 0.5[mm]. Las vigas que caracterizan al arreglo fueron consideradas como un material sólido, elástico e isotrópico de propiedades similares a las del hueso cortical con lo que se buscó reducir la complejidad en el modelado del hueso trabecular. Al imponer estas condiciones se obtuvieron los valores promedio del módulo de elasticidad para cada base y perturbación, los cuales se muestran en la tabla VI.1.

Tabla VI.1 Módulo de elasticidad promedio para las tres bases con su correspondiente perturbación.

MÓDULO DE ELASTICIDAD [MPa]			
	CUBICO SIMPE	CUBICO CENTRADO EN EL CUERPO	CUBICO CENTRADO EN LAS CARAS
0	413.3667763	231.703125	303.1751645
0.1	333.0463268	225.9109101	379.0767544
0.2	156.2617873	214.8486842	332.939693
0.3	97.85361842	217.533443	300.3612829

De estos resultados se pueden inferir lo siguiente, los valores del módulo de elasticidad obtenidos para la muestras, exceptuando el valor promedio de la estructura de base cúbica simple con perturbación de 0.3 y 0.2 en la desviación estándar, se encuentran entre los rangos de valores reportados en la literatura, ya que para fémur de cabra se tiene el rango de 234[MPa] hasta 770[MPa], para vértebras de cerdo se tienen valores entre

610[MPa] hasta 1550[MPa], para fémur de humano desde 190[MPa] hasta 1610[MPa] (An y Draughn, 2000). A pesar de que los módulos de elasticidad determinados se encuentran dentro del rango de valores, es importante mencionar que están en el límite inferior de dicho rango, esto se debe principalmente a dos aspectos, el primero de ellos es que si bien el hueso esponjoso desde el punto de vista de su microestructura, puede considerarse como la constitución de una celosía de placas y barras (Gibson, 1985), en este trabajo dada la complejidad y tiempo requerido para la generación de la estructura de Voronoi en 3D, se simplifico aún más el modelado al considerar solo un arreglo de vigas, lo cual sin duda provoca que la rigidez de la estructura se vea afectada, por lo tanto el módulo de elasticidad disminuye al eliminar a un elemento que aportaría soporte ante la imposición de una deformación. Otro aspecto que provoca que los módulos de elasticidad determinados para las estructuras se encuentren entre los valores más bajos reportados para el hueso trabecular, es que se consideraron elementos viga con el espesor mínimo determinado experimentalmente (Ramírez, 2004), lo cual provoca que módulo de elasticidad se vea afectado dado que se reduce el área de la sección transversal y por lo tanto su capacidad de soportar esfuerzos sin deformarse permanentemente.

En cuanto al comportamiento de las muestras en la curva esfuerzo-deformación, se presentó un comportamiento lineal como era de esperarse, ya que el comportamiento asignado a las vigas o trabéculas fue el de un material elástico e isotrópico lo cual lleva a que la estructura completa presente un comportamiento lineal.

Otro factor importante a observar es el comportamiento que siguen las propiedades mecánicas con respecto al grado de perturbación aplicado a las estructuras, medido a través de la desviación estándar de la matriz de perturbación, la siguiente gráfica muestra la relación que siguen la desviación σ y el módulo de elasticidad E.

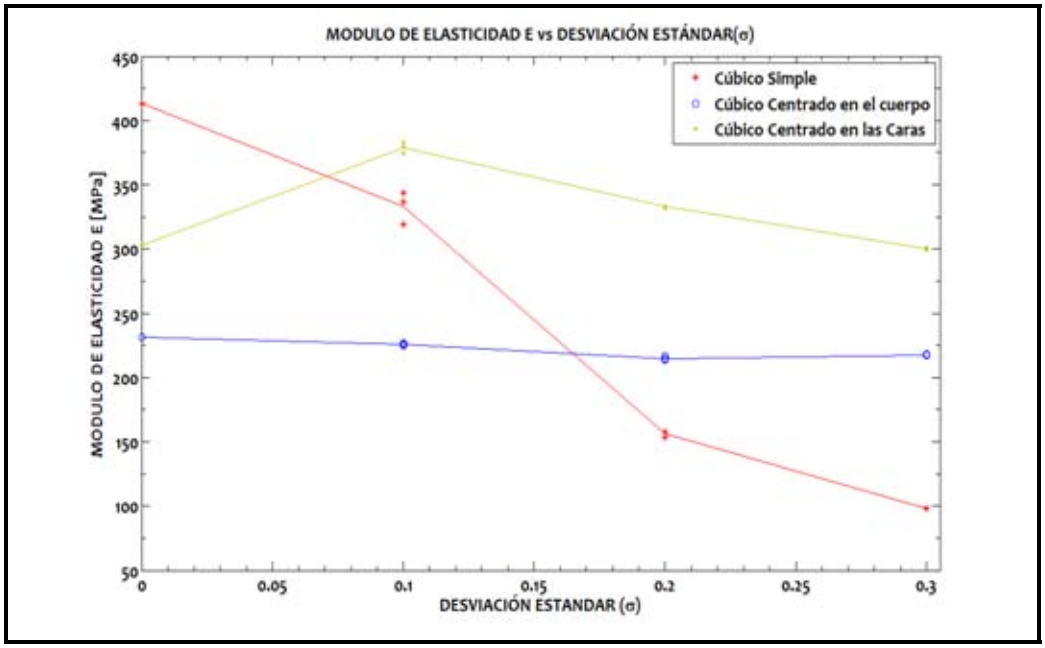


Figura VI.1 Módulo de Elasticidad (E) vs Desviación estándar (σ)

Para analizar esta gráfica, en primera instancia se hará trabajando de manera individual las estructuras base. Las primera de ellas, la base de nodos *cúbica simple,* muestra que al aumentar el grado de perturbación de la estructura disminuye el módulo de elasticidad de está, desde 413.3678 [MPa] hasta 97.8536 [MPa]. Esta disminución en comparación con las demás estructuras es mucho más evidente, la cual puede ser atribuida por un lado, a que cuando se tiene a la estructura sin perturbar el arreglo es completamente regular y tiene vigas paralelas a la dirección sobre la cual se aplica el desplazamiento de nodos (dirección 3), al tener estas vigas alineadas, la resistencia a deformarse en esa dirección se incrementa y en este caso se obtiene la mayor, lo que lleva a obtener el mayor valor del módulo de elasticidad para todas las distorsiones y bases, al aumentar el grado de perturbación las vigas pierden la regularidad y comienzan a tomar posiciones diferentes a la dirección de aplicación del desplazamiento lo que provoca que las fuerzas de reacción tengan que descomponerse dada su posición para actuar en la dirección adecuada, esto provoca que su resistencia a deformarse en la dirección 3 disminuya, con lo que se ve también disminuido el módulo de elasticidad. En la Figura VI.2 se observa como para la base cúbica simple, la perturbación provoca que la estructura pierda regularidad y las vigas tomen posiciones que merman su capacidad de reacción en la dirección 3 sobre la cual se aplica el desplazamiento.

Si bien, en este caso las estructuras al aumentar la perturbación disminuye su módulo de elasticidad, aumenta la resistencia a deformarse en otras direcciones lo cual no ocurre para la estructura cúbica simple sin perturbación, que al aplicarle un desplazamiento en una dirección diferente a la 3, su resistencia se verá disminuida de manera importante. Esto tiene una connotación fisiológica importante, ya que si el hueso trabecular siguiera un arreglo estructural regular como el presentado en la estructura con base cúbica simple, el hueso humano no podría soportar los requerimientos mecánicos a los que esta sometido ya que tendría una elevada resistencia a la deformación en una dirección pero en ante la solicitación de esfuerzo en otras direcciones se vería superado, por lo cual un arreglo regular no puede ser el que se aproxime con mayor exactitud al hueso trabecular.

Otro factor importante que produce la caída del módulo de elasticidad al aumentar la perturbación en la estructura, es que esto disminuye el número de vigas presentes en ella, lo cual no sucede para las demás estructuras en las cuales la tendencia es que al aumentar la perturbación aumenta el número de vigas, este factor es relevante para explicar el porqué de la disminución abrupta del módulo de elasticidad en la base cúbica simple con respecto a las demás bases, ya que al disminuir el número de elementos de la estructura se disminuye también su resistencia a la deformación. La disminución en el número de vigas se observa al determinar la longitud de cada una y sumarla, para obtener la longitud total, la cual disminuye al disminuir el número de vigas, la siguiente tabla muestra cómo se comporta dicha longitud para cada base y perturbación. Otra forma de observarlo es mediante el número de elementos generados en la matriz *elemento*.

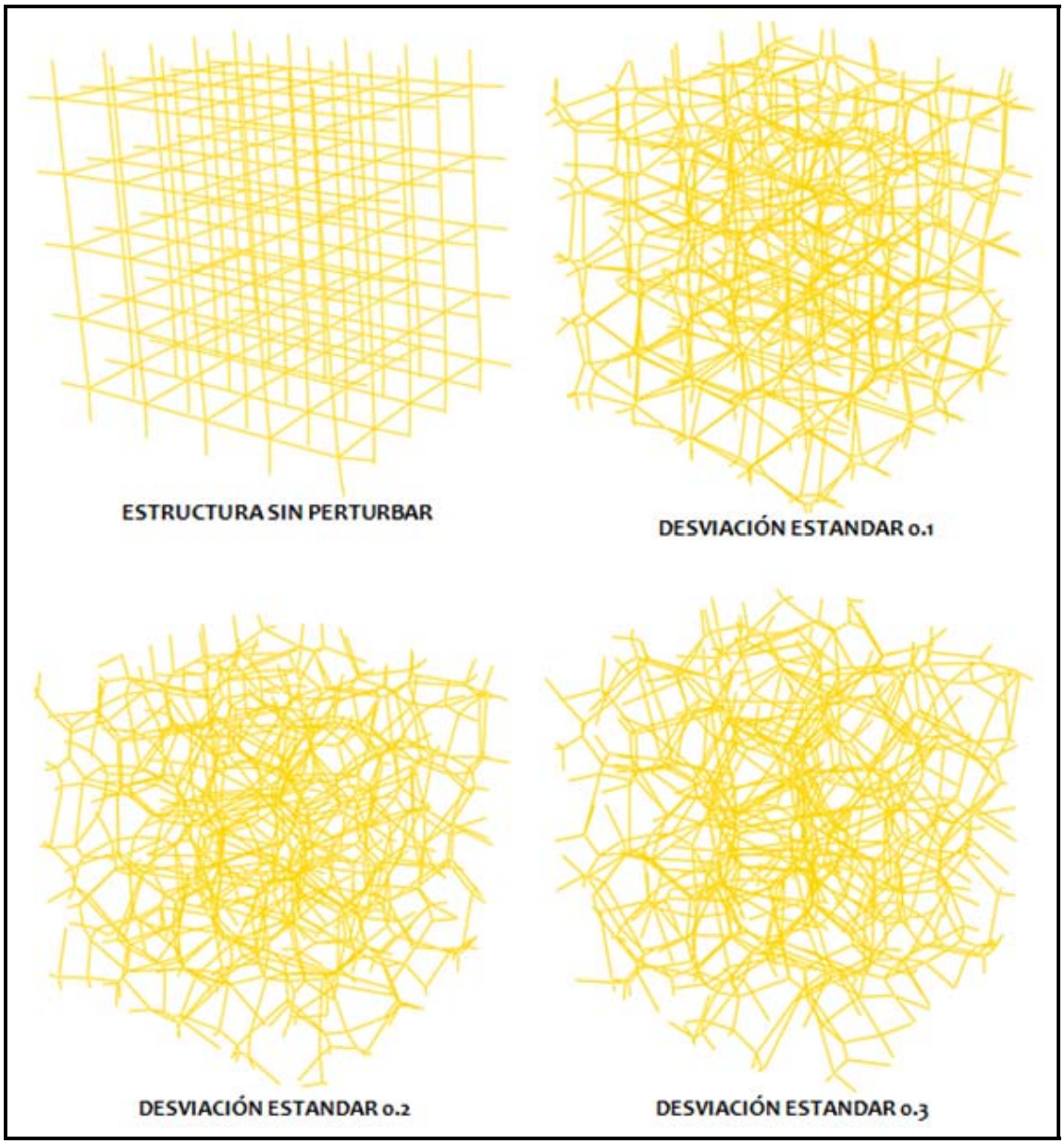


Figura VI.2 Comportamiento de la estructura (base cúbica simple) con respecto al grado de perturbación.

En esta Tabla IV.2 se observa el comportamiento mencionado anteriormente. La siguiente base a analizar es la *cúbica centrada en el cuerpo,* el comportamiento que sigue el módulo de elasticidad con respecto al grado de perturbación, es una disminución de este al aumentar la perturbación, sin embargo dicha disminución no es marcada ya que los valores del módulo de elasticidad se encuentran entre 231.7031 [MPa] y 214.8487 [MPa].

Tabla IV.2 Longitud total de las vigas presente en cada arreglo.

	LONGITUD [mm]		
	CUBICO SIMPE	**CUBICO CENTRADO EN EL CUERPO**	**CUBICO CENTRADO EN LAS CARAS**
0	708.97	887.31	1092.3
0.1	689.209667	970.13	1294.966667
0.2	673.51	1019.833333	1284.8
0.3	651.926667	1023.3	1590.833333

En la figura VI.1 se observa que en comparación con la base *cúbica simple* y *cúbica centrada en las caras* la caída del módulo de elasticidad es pequeña, esto puede ser atribuido por un lado a que el número de nodos en esta base es mayor en comparación con la base cúbica simple, lo cual provoca que al existir una perturbación en los nodos esta no afecta en gran medida el arreglo de Voronoi, ya que si bien existe un mismo desplazamiento de los nodos al existir una mayor cantidad de vecinos en el mismo espacio disminuye la posibilidad de alterar la celdas dada la cercanía entre ellos, por lo cual el arreglo no se altera en la misma proporción que ocurre para una base de menor número de nodos, por lo cual la perturbación no es tan marcada. Esto se puede observar al comparar las figuras VI.2 y VI.3, con lo cual se ve que la mayor perturbación la sufre la cúbica simple ya que su geometría inicial se ve afectada de manera importante al perturbar los nodos, esta perturbación se refleja en el cambio del módulo de elasticidad, por un lado la *cúbica simple* disminuye abruptamente su módulo de elasticidad mientras que la base *cúbica centrado en el cuerpo* lo hace pero de una manera suave, de manera análoga al cambio de su geometría.

Ahora, ésto satisface al considerar a la base *cúbica simple* y *cúbica centrada en el cuerpo,* sin embargo el comportamiento que sigue la estructura con base *cúbica centrada en las caras* tiene mayor complejidad ya que si bien, los cambios en el módulo de elasticidad no son tan marcados como en la base *cúbica simple* y pueden ser explicados a partir del número de nodos vecinos presentes en la estructura, el comportamiento del módulo de elasticidad con respecto al grado de perturbación no es el que hasta el momento se había presentado, ya que el módulo de elasticidad para la estructura sin perturbar se encuentra entre los valores más bajos, dentro de sus estructuras con perturbación, lo cual en este caso se atribuye, a que si se consideró que al aumentar el número de nodos al aplicar un desplazamiento en ellos la estructura no se veía afectada de manera importante, si el número se incrementa considerablemente como sucede en este caso, en el que se pasa de 9 nodos por milímetro cúbico (cúbica centrada en el cuerpo) a 14 nodos por milímetro cúbico (cúbica centrada en las caras) la perturbación nuevamente altera de manera substancial la geometría de la estructura, esto puede visualizarse en la figura VI.4 donde de la estructura sin perturbar a la estructura con una desviación estándar de 0.1, se observa que el cambio de su geometría es mas evidente que en las otras bases, esto también se hizo evidente en el momento en el cual se exportaban las estructuras a la paquetería de ABAQUS®, ya que esta base presentó la mayor cantidad de problemas para ser aceptada, por que al distorsionar a la estructura se presentaban problemas con la orientación de las vigas y abortaban los análisis.

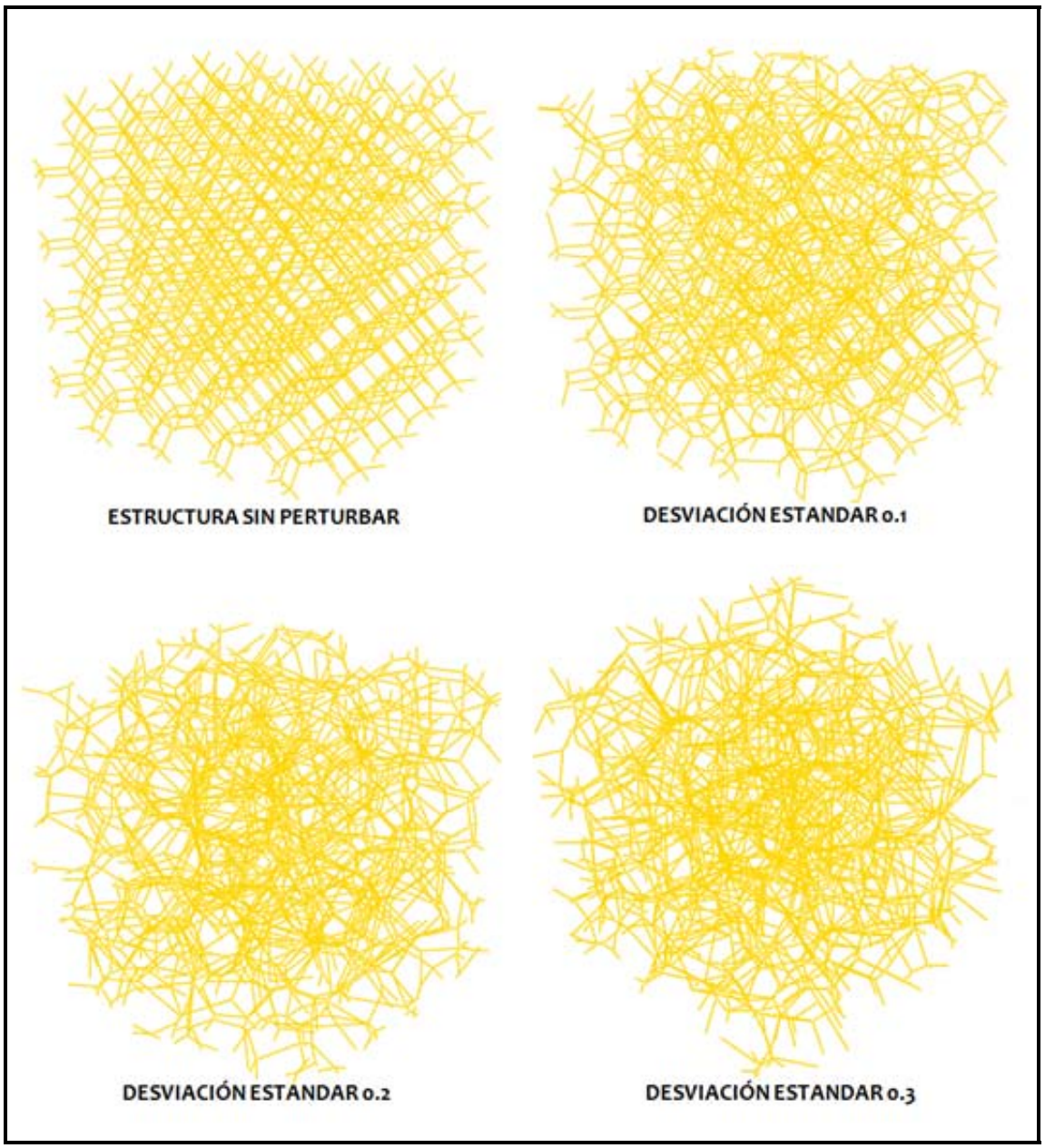


Figura VI.3 Comportamiento de la estructura (base cúbica centrada en el cuerpo) con respecto al grado de perturbación.

Esto por una lado explica por que el cambio en el comportamiento del módulo de elasticidad con respecto a la desviación estándar, junto con este factor, está el incremento del número de vigas que existe entre la estructura sin perturbar y la estructura perturbada que como se observa en la tabla IV.2 es un aumento considerable y contribuye al aumento en le módulo de elasticidad, ambos factores producen que para la estructura sin perturbar presente un menor módulo de elasticidad con respecto a las estructuras disturbadas.

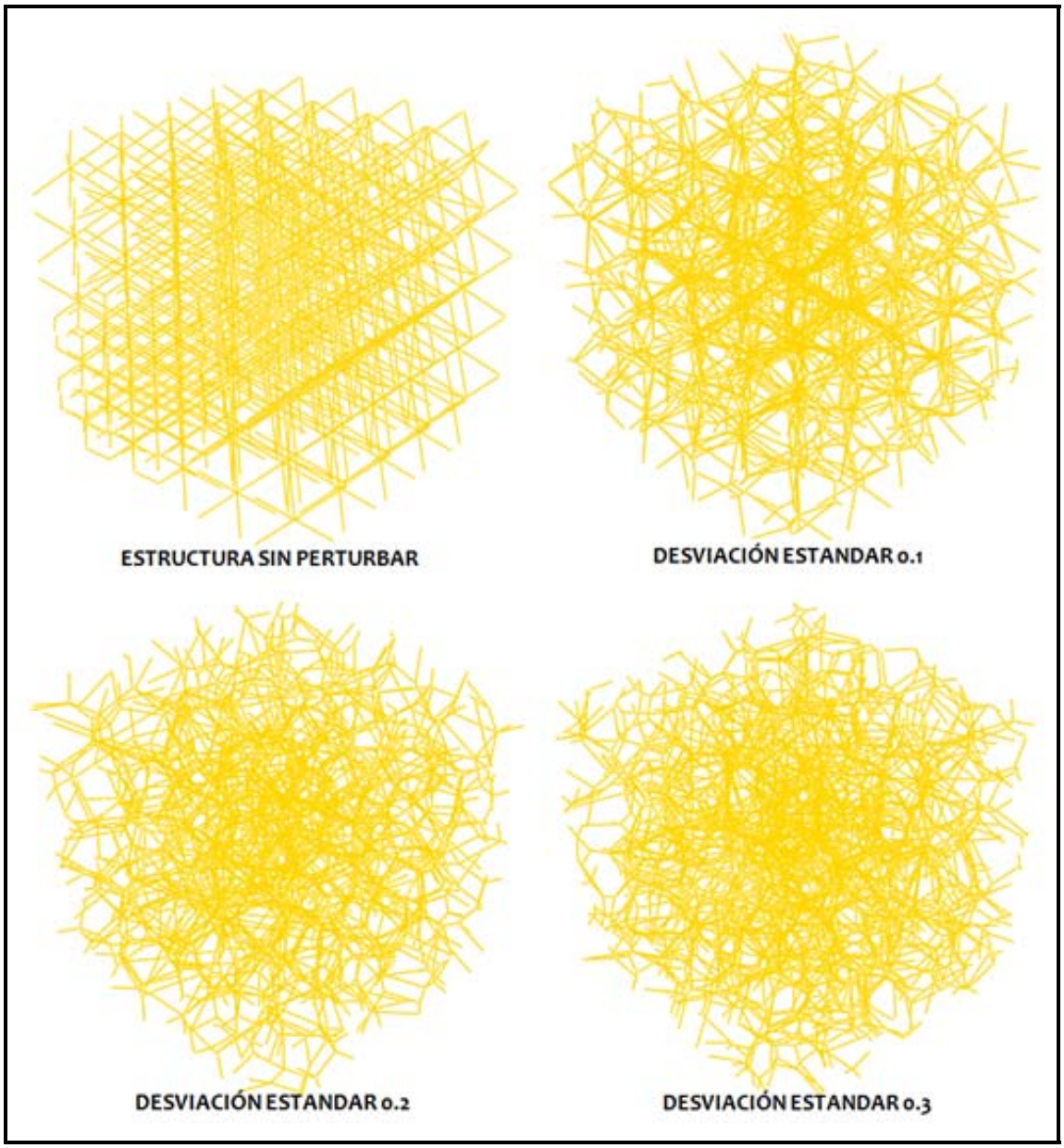


Figura VI.4 Comportamiento de la estructura (base cúbica centrada en el cuerpo) con respecto al grado de perturbación.

VI.2 Sobre el módulo de rigidez (G)

Para la determinación del módulo de rigidez, como ya se mencionó, se trabajó con una probeta virtual cúbica de 4.864 [mm] de arista, a la cual se le impuso un desplazamiento a los nodos de la parte superior del cubo correspondiente a 0.5[mm] en una dirección tangente a dicha cara. Al imponer estas condiciones se obtuvieron los valores promedio del módulo de rigidez para cada base y perturbación, los cuales se muestran en la siguiente tabla.

Tabla IV.3 Módulo de rigidez de las estructuras.

MÓDULO DE RIGIDEZ			
	CUBICO SIMPE	CUBICO CENTRADO EN EL CUERPO	CUBICO CENTRADO EN LAS CARAS
0	2.6192	27.97746711	18.91362116
0.1	12.82955583	30.62278247	46.09624714
0.2	15.96150045	27.59304796	43.1778695
0.3	12.54382159	27.75451158	41.4746049

De estos resultados se pueden inferir lo siguiente, los valores del módulo de rigidez obtenidos para las muestras se encuentran muy por debajo de lo reportado en la literatura en la cual se encuentran rangos de entre 90[MPa] y 170[MPa], esto se atribuye principalmente a la simplificación de hueso trabecular que en este trabajo se realizó, ya que al considerar el arreglo de vigas y eliminar a las placas se mermó la capacidad de la estructura para soportar esfuerzos cortantes y se muestra reflejado en el módulo de rigidez. Otro factor que se considera afectó al módulo de rigidez fue que en este trabajo se consideraron vigas con un espesor mínimo reportado experimentalmente.

Ahora el comportamiento que sigue el módulo de rigidez con respecto al grado de perturbación reflejado en la desviación estándar se muestra en la siguiente grafica.

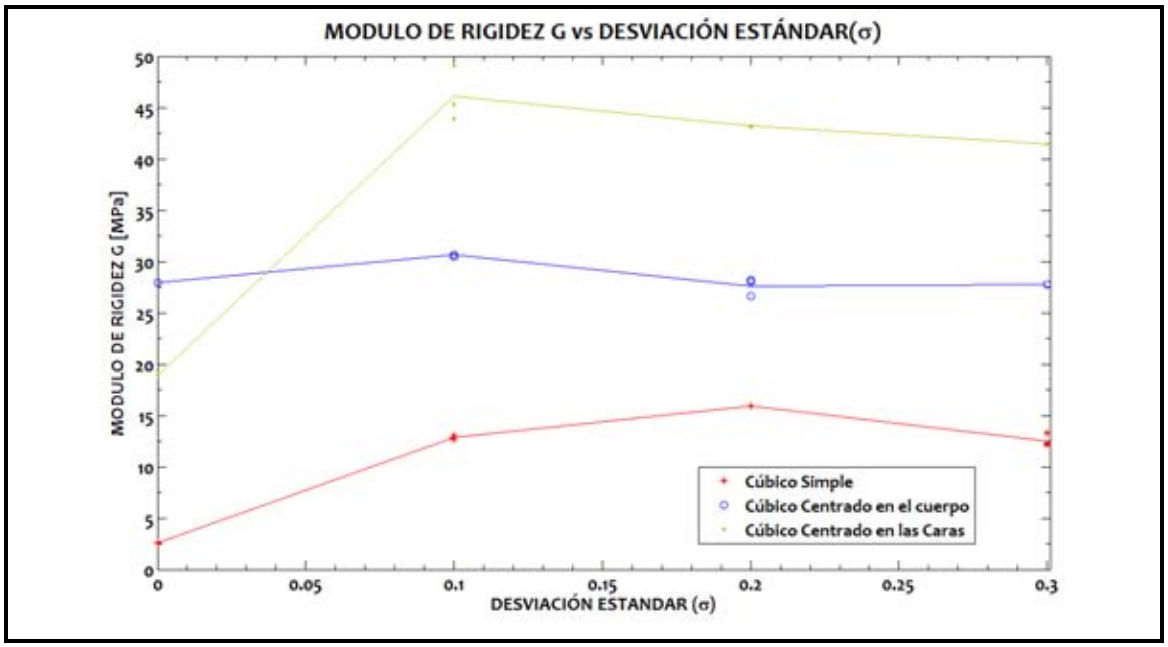


Figura VI.1 Módulo de rigidez (G) vs Desviación estándar (σ)

En esta gráfica se puede observar que para las bases cúbica simple y cúbica centrada en el cuerpo se sigue un comportamiento similar, donde al aumentar el grado de perturbación se aumenta también el módulo de rigidez, esto se debe a que cuando se distorsionan las estructuras las posiciones de las vigas cambian de tal manera que su posición no es perpendicular al desplazamiento aplicado, con lo cual su capacidad de reacción ante el desplazamiento cortante aumenta, este efecto se ve resaltado con la base cúbica simple la cual tiene la máxima capacidad de reacción para un desplazamiento en el dirección 3 y un

mínimo en una dirección cortante esto, principalmente por la dirección preferencial de sus vigas, para la base cúbica centrada en las caras se observa que nuevamente su comportamiento es diferente a los dos estructuras manejadas anteriormente, los factores a los cuales se atribuye este comportamiento son como se mencionó anteriormente la facilidad de distorsionarse de esta estructura dada la cantidad de nodos vecinos presentes en ella y la cantidad de vigas presente ella, la cual aumenta su rigidez dada la cantidad de material presente en la estructura lo cual es un factor importante para este aumento.

VI. 3 Sobre el coeficiente de Poisson (ν)

Los resultados obtenidos para el coeficiente de Poisson en todos los casos son cercanos a los reportados en la literatura. Para el hueso trabecular, el coeficiente de Poisson reportado es de 0.3 mientras que los valores obtenidos en las probetas de Voronoi oscilan entre 0.105 para CS con desviación estándar de 0.1 hasta 0.381 en CS sin perturbación. Los resultados obtenidos para las diferentes estructuras se muestran a continuación.

Tabla IV.4 Coeficiente de Poisson de las estructuras.

COEFICIENTE DE POISSON			
	CUBICO SIMPE	**CUBICO CENTRADO EN EL CUERPO**	**CUBICO CENTRADO EN LAS CARAS**
0	0.381556584	0.30513934	0.225620985
0.1	0.261685388	0.316587818	0.240694132
0.1	0.195621362	0.317474795	0.238845854
0.1	0.203114194	0.317760342	0.227481477
0.2	0.366087415	0.271934062	0.261530499
0.2	0.301520596	0.277824791	0.265127804
0.2	0.368141249	0.32140135	0.265127804
0.3	0.282012198	0.270412998	0.267761021
0.3	0.268721344	0.271864466	0.27394966
0.3	0.375077794	0.270445486	0.267761021

La desviación estándar no promueve una tendencia en el coeficiente de Poisson según los resultados obtenidos.

VI. 4 Sobre los coeficientes de anisotropía (δ y ξ)

Para la obtención de éstos coeficientes se determinaron las deformaciones ε_{12}, ε_{11}, ε_{22} y ε_{33} a través de Abaqus®, con el fin de observar la influencia de las perturbaciones sobre la anisotropía de las estructuras de Voronoi y así, poder aproximar el comportamiento de las probetas al del hueso trabecular.

Para las estructuras sin perturbación, se espera que el comportamiento sea lo más cercano a las estructuras cúbicas CS, CCC y CCF que son estructuras cúbicas. Por tanto, el coeficiente anisotrópico δ en dichas estructuras debería tender a cero ya que las deformaciones en las direcciones perpendiculares a la aplicación del desplazamiento, en un material isotrópico, tienen la misma magnitud. Los resultados obtenidos se muestran en la tabla siguiente.

Tabla IV.5 Coeficiente anisotrópico δ de las estructuras.

δ			
	CUBICO SIMPE	**CUBICO CENTRADO EN EL CUERPO**	**CUBICO CENTRADO EN LAS CARAS**
0	0.007631131	-0.002331334	-0.00538336
0.1	-0.024627686	0.014382764	0.00858897
0.1	0.028213435	0.014545918	0.011202031
0.1	0.013659566	0.009947497	-0.004634646
0.2	-0.021226372	0.040386136	-0.003636714
0.2	-0.02871114	0.041298457	-0.002091612
0.2	-0.021698622	0.021512916	-0.002091612
0.3	0.000324812	0.065165785	-0.036540709
0.3	-0.00936484	0.067246174	-0.038868485
0.3	0.037187859	0.06510081	-0.039006616

De estos resultados se puede observar que en las estructuras sin perturbación, el coeficiente δ presenta valores que sin ser enteramente cero, reflejan un alto grado de simetría entre las deformaciones en las direcciones 1 y 2, lo cual se asemeja al comportamiento de las estructuras cúbicas. Existen dos factores importantes a considerar que se atribuyen a ésta discrepancia en los resultados: por un lado, es posible que al generar las estructuras sin perturbación mediante la paquetería de MATLAB®, se produjeran errores de redondeo al momento de generar las celdas de Voronoi. Esto crea vigas oblicuas que alteran el comportamiento de la estructura sujeta a compresión uniaxial, por otro lado, en ABAQUS® se presentó un error de redondeo al importar la geometría de las estructuras de Voronoi ya que no se consideraron algunos decimales. Finalmente, se debe considerar el efecto de recortar las probetas mediante planos virtuales de manera arbitraria.

Ahora bien, como ya se mencionó, lo que se pretende al generar distorsiones en las semillas de las celdas de Voronoi es inducir anisotropía en el material y de esta forma, aproximar el comportamiento de las mismas al del hueso esponjoso. Analizando los resultados obtenidos con diferentes valores de desviación estándar se puede observar que el coeficiente δ refleja mayor anisotropía conforme la desviación estándar aumenta, pasando de un valor de 0.0023 para la estructura CCC sin perturbación a 0.065 para la misma estructura con desviación estándar de 0.3, conforme a lo esperado.

Cabe destacar que el coeficiente anisotrópico ζ será de mayor orden que δ ya que mientras el numerador de δ es una diferencia de deformaciones, ζ simplemente es el cociente de una deformación por otra, por lo tanto, se espera que el error de aproximación sea de mayor orden. De igual manera que en el coeficiente δ, para las estructuras sin perturbación el coeficiente ζ debe ser lo más cercano posible a cero, ya que en las estructuras isotrópicas las deformaciones cortantes son enteramente iguales a cero. Sin embargo, en los resultados obtenidos, éste factor varía entre 0.17 y 0.39 sin reflejar tendencia a disminuir en las estructuras no disturbadas. De nuevo se puede atribuir dicha dispersión a los errores de redondeo al exportar la geometría a ABAQUS® aunque en éste caso, no se puede suponer una tendencia similar al de δ ya que una estructura con poca perturbación podría favorecer mayores deformaciones cortantes que una con perturbación importante en la que el alto grado de anisotropía favorezca el empotramiento en algunas zonas y de esa forma, reducir la anisotropía de la estructura. Además, cabe mencionar que los ensayos no se realizaron sobre estructuras completamente regulares, es decir, las estructuras regulares CS, CCC, CCF se tomaron como referencia para las semillas con que se generaron las celdas de Voronoi por lo que las estructuras sin desviación estándar siguen siendo arreglos de celdas de Voronoi.

A continuación se anexan los resultados obtenidos.

Tabla IV.6 Coeficiente anisotrópico ξde las estructuras.

	ζ		
	CUBICO SIMPE	**CUBICO CENTRADO EN EL CUERPO**	**CUBICO CENTRADO EN LAS CARAS**
0	0.381556584	0.30513934	0.225620985
0.1	0.261685388	0.316587818	0.240694132
0.1	0.195621362	0.317474795	0.238845854
0.1	0.203114194	0.317760342	0.227481477
0.2	0.366087415	0.271934062	0.261530499
0.2	0.301520596	0.277824791	0.265127804
0.2	0.368141249	0.32140135	0.265127804
0.3	0.282012198	0.270412998	0.265718866
0.3	0.268721344	0.271864466	0.270289874
0.3	0.375077794	0.270445486	0.272263629

VII. CONCLUSIONES

A partir del trabajo desarrollado a lo largo de la investigación y de los resultados obtenidos de ella se puede concluir en tres puntos principales.

1. La generación de estructuras de Voronoi en tres dimensiones es un proceso factible de desarrollar. A lo largo de este trabajo se presentó la metodología para su generación así como su manipulación. La interacción entre MATLAB® y ABAQUS® es proceso que igualmente es viable mediante la interfase adeacuada, en este caso un archivo de *inp*, que permite exportar las estructuras de MATLAB® para poder practicar en ellas ensayos de compresión y deslizamiento con el fin de determinar las propiedades mecánicas de dichas estructuras. Con este primer punto se cumple con el objetivo de este trabajo, que era plantear la metodología para la creación de estructuras de Voronoi en tres dimensiones, la cual en este punto no se tenia disponible.
2. Las estructuras de Voronoi que siguieron un comportamiento más aproximado al hueso trabecular fueron aquellas a las que se les aplicaron un grado de perturbación, las estructuras sin perturbar si bien presentaron el módulo de elasticidad más elevado perdían la capacidad de soportar esfuerzos en otras direcciones, lo cual pone en evidencia su gran diferencia con un el comportamiento del hueso trabecular, ya que este debido a su capacidad de adaptarse a los requerimientos mecánicos a los cuales es sometido no puede perder la capacidad de soportar esfuerzos cortantes de la manera en la que ocurre en las estructuras sin perturbar.
3. Para las probetas virtuales de Voronoi generadas en MATLAB® y simuladas en ABAQUS®, con propiedades de hueso cortical medidas experimentalmente (Ramírez, 2007), se obtuvieron módulos de elasticidad con una buena aproximación dado que dichos resultados se encuentran dentro de los rangos reportado en la literatura. Sin embargo estos resultados encuentran en el limite inferior de estos rangos que en este caso se atribuye a que para las simulaciones realizadas se asignó un espesor de viga que se encuentra entre los más pequeños, lo cual llevó a una subestimación de los resultados.

De lo anterior se puede observar que para una primera aproximación de la mesoestructura del hueso trabecular se obtuvieron resultados sin embargo hay en este punto mucho en que trabajar. Dentro de las perspectivas que se tienen de este trabajo, está el avance de la automatización de este proceso, en el cual se obtenga la estructura de Voronoi en MATLAB®, se exporte dicha estructura a la paquetería ABAQUS®, se simulen los ensayos de compresión y deslizamiento, se obtengan de los resultados de dichas simulaciones los datos de mayor relevancia y se determinen de estos datos las propiedades mecánicas de esta estructura, dentro de este aspecto se han tenido ya avances al lograr correr en MATLAB®

diversas estructuras y generar archivos *.inp* con distinto nombre siendo esto el principal problema en un inicio, otro aspecto que se a logrado avanzar es encadenar eventos entre MATLAB® y ABAQUS® mediante archivo bach, si embargo no se ha logrado aún extraer datos de ABAQUS® de manera automática, lo cual es el trabajo principal que se tiene en este momento.

Finalmente ya automatizado el proceso se puede trabajar mucho mas con los modelos, variando la geometría de la sección transversal de las vigas, la densidad mediante el cambio de espesores de las vigas trabeculares, variar también las bases que generan el arreglo de Voronoi, considerar en el análisis elemento placa para observar su comportamiento y buscar de esta forma modelos que se apeguen más a la realidad.

REFERENCIAS

Ballesteros, Rafael. 2004. "**Traumatología y medicina deportiva"**. Tomo 1. Cengage Learning Editores. España.

Callister, W.D. 1997. **"Introducción a la Ciencia e Ingeniería de los Materiales**". Editorial Reverté, S.A. Barcelona.

Cuppone, M. ; Seedhom, B. ; Berry, E. ; Ostell, A. 2004. **"The Longitudinal Young's Modulus of Cortical Bone in the Midshaft of Human Femur and its Correlation with CT Scanning Data"**. Vol. 74. No. 3. Pags. 302-309.

Gibson, L. J. 1985. **"The mechanical behaviour of cancellous bone"**. Journal of Biomechanics. Vol. 18. No. 5. Pags 317-318

Mark de Berg; Marc van Kreveld; Mark Overmars. 2007. **"Computational Geometry: Algorithms and Applications**". Second Edition**.** Springer-Verlag.

Narvaez, Monica Y. 2004. **"Metodos para el análisis de la microestructura y propiedades mecánicas del hueso esponjoso".** Tesis licenciatura. UNAM; México D.F.

Ramirez, Edgar I. 2007. **"Desarrollo de un modelo micromecánico para la predicción de las propiedades de hueso mediante paquetería de elemento finito."** Tesis de maestría. UNAM; México D.F.

Weiss L. 1988. **"Cell and Tissue Biology. A Textbook of Histology.** Urban and Schwarzenbergn. EUA.

ANEXO I

```
*Heading
Hueso Trabecular
Caracterización de una estructura de Voronoi
*Preprint, echo=YES, model=YES, history=YES
**
** PARTS
**
*Part, name=alambre
*End Part
**
** ASSEMBLY
**
*Assembly, name=Assembly
**
*Instance, name=alambre-1, part=alambre
*Node
101,0.30272,0.8519,0.93839
102,0.16037,0.87313,1.0649
.
.

*Element, type=B33, elset=VORO1
101,101,102
102,101,760
.
.

*Beam General Section, elset=VORO1, poisson = 0.3, section=CIRC
0.08
0.0,0.0,-1.0
1000.,384.6154
*End Instance
**
*Nset, nset=EMPOTRADO
761,766

*Nset, nset=DESPLAZAMIENTO
1074,1075

*Nset, nset=X_inf
837,841

*Nset, nset=X_sup
764,765

*Nset, nset=Y_inf
807,811

*Nset, nset=Y_sup
762,763
```

```
*Nset, nset=pivote
764

*End Assembly
**
** ------------------
**
** STEP: paso-1
**
*Step, name=paso-1
*Static
0.1, 1., 1e-05, 0.1
**
** BOUNDARY CONDITIONS
**
*Boundary
DESPLAZAMIENTO, 3, -2.5
*Boundary
EMPOTRADO, 3, 0
*Boundary
pivote,encastre
**
** OUTPUT REQUESTS
**
**
*Output, field, variable=PRESELECT
**
*Output, history, variable=PRESELECT
*End Step
```

ANEXO II

```
%GENERA UNA RED DE VORONOI
clc
clear all
%SE GENERA LOS CENTROS DE LAS CELDAS DE VORONOI
%ESTRUCTURA CUBICA CENTRADA EN EL CUERPO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
d=[0:1:5];%CAMBIAR TERCER ELEMENTO PARA CAMBIAR
LA MAGNITUD DEL CUBO DE NODO VORONOI
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PARAMETROS DEL PROGRAMA
%DESVIACIÓN ESTANDAR
sigx=0.3;
sigy=0.3;
sigz=0.3;
%FACTORES DE ESCALA
ax=1;
ay=1;
az=1;
%LIMITES DE CORTE EN Z
planoinferior=0.123;
planosuperior=4.987;
%LIMITES DE CORTE EN X
planoinferior1=0.123;
planosuperior1=4.987;
%LIMITES DE CORTE EN Y
planoinferior2=0.123;
planosuperior2=4.987;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[x,y,z]=meshgrid(d,d,d);
B=[x(:),y(:),z(:)];
S=[B];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%CREANDO LA MATRIZ D DISTURBIO
for nn=1:1:size(S,1)

D(nn,:)=[random('norm',0,sigx,[1,1]),random('nor
m',0,sigy,[1,1]),random('norm',0,sigz,[1,1])];
end
D;
F=S+D;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MATRIZ DE TRANSFORMACIÓN T
T=[ax 0 0;0 ay 0;0 0 az];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MATRIZ DE SEMILLAS A
A=F*T;
%DETERMINA LOS VERTICES DE VORONOI FUNCION
'voronoin'
[V,C]=voronoin(A);
banda=0;
for a=1:size(C,1)
    X=V(C{a},:);
    if(C{a}(1)~=1)
        banda=banda+1;
        Z=X;
        matrizZ(banda).matrix1=Z;
        K=convhulln(Z);
        matrizK(banda).matrix1=K;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%GENERACIÓN DE LA GRAFICA MATLAB
        d=[1 2 3 1];
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DETERMINACIÓN DE LOS PUNTOS COPLANARES
marcador1=size(matrizK);
for q=1:1:marcador1(2)
    tols=10e-10;
    toli=-10e-10;
    marca4=size(matrizK(q).matrix1,1);
    for g=1:1:marca4

k1_1=matrizZ(q).matrix1(matrizK(q).matrix1(g,1),
:);

k1_2=matrizZ(q).matrix1(matrizK(q).matrix1(g,2),
:);

k1_3=matrizZ(q).matrix1(matrizK(q).matrix1(g,3),
:);
        u=(k1_1-k1_2);
        v=(k1_1-k1_3);
%DEFINICIÓN DE VECTOR NORMAL
        N=[u(1,2)*v(1,3)-
u(1,3)*v(1,2),u(1,3)*v(1,1)-
u(1,1)*v(1,3),u(1,1)*v(1,2)-u(1,2)*v(1,1)];
        N1=N(1,1);
        N2=N(1,2);
        N3=N(1,3);
%PROBAR CADA TRIANGULO DE LA MATRIZ K
        coplanares=[ , ];
        marca3=size(matrizK(q).matrix1,1);
        for i=1:1:marca3;
            j=1:3;
%ECUACIÓN DEL PLANO QUE DEFINE CADA ELEMENTO DE
LA MATRIZ Z

plano=N1*matrizZ(q).matrix1(matrizK(q).matrix1(i
,j),1)+N2*matrizZ(q).matrix1(matrizK(q).matrix1(
i,j),2)+N3*matrizZ(q).matrix1(matrizK(q).matrix1
(i,j),3)-N1*k1_1(1,1)-N2*k1_1(1,2)-N3*k1_1(1,3);

if((plano(1,1)>toli&plano(1,1)<tols)&(plano(2,1)
>toli&plano(2,1)<tols)&(plano(3,1)>toli&plano(3,
1)<tols))

coplanares=[coplanares;matrizK(q).matrix1(i,:)];
                    coplanares;
                else
                end
        end
        %eval(['coplanares', int2str(g),
'=coplanares']);
        romeo(g).matriz=coplanares;
    end
%DEFINICIÓN ESTRUCTURA ANIDADA
        koplanar(q).matrix=romeo;
        romeo=struct('matriz',[]);
end
%DETERMINACIÓN DE ARISTAS
marca6=size(koplanar);
for f=1:1:marca6(2)
marca=size(koplanar(f).matrix);
for n=1:1:marca(2);
    koplanar(f).matrix(n);
    h=0;
```

```
    limite=size(koplanar(f).matrix(n).matriz);
    for s=1:1:limite(1)

prueba=[koplanar(f).matrix(n).matriz(s,1:2);kopl
anar(f).matrix(n).matriz(s,1:2:3);koplanar(f).ma
trix(n).matriz(s,2:3)];
    for p=1:1:3
    band2=0;
    for r=1:1:limite(1)
        band=0;
        while band<2
            aux=prueba(p,2);
            prueba(p,2)=prueba(p,1);
            prueba(p,1)=aux;
            if s~=r
                if
prueba(p,:)==koplanar(f).matrix(n).matriz(r,1:2)
|
prueba(p,:)==koplanar(f).matrix(n).matriz(r,2:3)
|
prueba(p,:)==koplanar(f).matrix(n).matriz(r,1:2:
3)
                    band2=1;
                end
            end
            if band2==1
                band=2;
            else
                band=band+1;
            end
        end
    end
    if band2==0;
            h=h+1;
            aristas(h,:)=prueba(p,:);
    end
    end
    prueba;
    band2;
end
aristas;
julieta(n).matriz=aristas;
aristas=[];
end
edges(f).matrix2=julieta;
julieta=struct('matriz',[]);
end
%DETERMINAR COORDENADAS DE LAS ARISTAS REFERIDOS
A LA CORRESPONDIENTE K
coordenadas=[ , ];
marca7=size(matrizZ);
for k=1:1:marca7(2)
    marca1=size(edges(k).matrix2);
        for y=1:1:marca1(2)

marca2=size(edges(k).matrix2(y).matriz,1);
                for x=1:1:marca2

coordenadas=[coordenadas;matrizZ(k).matrix1(edge
s(k).matrix2(y).matriz(x,1),:),matrizZ(k).matrix
1(edges(k).matrix2(y).matriz(x,2),:)];
                end
                coordenadas;
        end
 koordinaten=coordenadas;
end
%SE ELIMINAN LOS REGLONES REPETIDOS
cont1=1;
for m1=1:1:size(koordinaten,1)
    renglon=koordinaten(m1,:);
    rep=0;
    for l=m1:-1:1
        if renglon==koordinaten(l,:)
            rep=rep+1;
        end
    end
    if rep==1
        os(cont1,:)=renglon;
        cont1=cont1+1;
    end
end
os;
%SE ELIMINAN LOS REGLONES ESPEJO
cont=1;
for m=1:1:size(os,1)
    renglon1=[os(m,1:6)];
    rep1=0;
        for w=m:-1:1
            if renglon1==[os(w,4:6),os(w,1:3)];
                rep1=rep1+1;
            end
        end
        if rep1==0
            bones(cont,:)=renglon1;
            cont=cont+1;
        end
end
bones;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%CREAR MATRIZ
PARA ADAPTAR CORTE DE PLANO Z
ossa4=bones;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PUNTO DE LA RECTA
for mm=1:1:size(ossa4,1)
    P0(mm,:)=[ossa4(mm,1:1:3)];
end
P0;
%VECTOR DIRECTOR DE LA RECTA
for mm1=1:1:size(ossa4,1)
    uu(mm1,:)=[ossa4(mm1,1)-
ossa4(mm1,4),ossa4(mm1,2)-
ossa4(mm1,5),ossa4(mm1,3)-ossa4(mm1,6)];
end
uu;
%PARAMETRO PARA DETERMINAR EL PUNTO DE
INTERSECCION INFERIOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
planoinferior; %DEPENDE DEL PLANO QUE INTERSECTA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for mm3=1:1:size(ossa4,1)
    tt(mm3,:)=[(planoinferior-
P0(mm3,3))/uu(mm3,3)];
end
tt;
%PUNTO DE INTERSECCION
for mm2=1:1:size(ossa4,1)

puntoii(mm2,:)=[P0(mm2,1)+(tt(mm2,:)*uu(mm2,1)),
P0(mm2,2)+(tt(mm2,:)*uu(mm2,2)),P0(mm2,3)+(tt(mm
2,:)*uu(mm2,3))];
end
puntoii;
%MATRIZ OSSA4/INTERSECCION
for mm4=1:1:size(ossa4,1)

intersec(mm4,:)=[ossa4(mm4,:),puntoii(mm4,:)];
end
intersec;
%DEFINIENDO MATRIZ CON PLANO INFERIOR CORTADO
marca101=0;
for mm5=1:1:size(intersec,1)
```

```
    if
intersec(mm5,9)>=intersec(mm5,3)&intersec(mm5,9)
<=intersec(mm5,6)|intersec(mm5,9)<=intersec(mm5,
3)&intersec(mm5,9)>=intersec(mm5,6)
        marca101=1;
        if intersec(mm5,3)>planoinferior

fronteraii(mm5,:)=[intersec(mm5,1:3),intersec(mm
5,7:9)];
        else

fronteraii(mm5,:)=[intersec(mm5,4:6),intersec(mm
5,7:9)];
        end
    else
        marca101=2;

fronteraii(mm5,:)=[intersec(mm5,1:1:3),intersec(
mm5,4:1:6)];
    end
end
fronteraii;
%PARAMETRO PARA DETERMINAR EL PUNTO DE
INTERSECCION SUPERIOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
planosuperior; %DEPENDE DEL PLANO QUE INTERSECTA
SUPERIOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for mm6=1:1:size(ossa4,1)
    tt1(mm6,:)=[(planosuperior-
P0(mm6,3))/uu(mm6,3)];
end
tt1;
%PUNTO DE INTERSECCION
for mm7=1:1:size(ossa4,1)

punto1ii(mm7,:)=[P0(mm7,1)+(tt1(mm7,:)*uu(mm7,1)
),P0(mm7,2)+(tt1(mm7,:)*uu(mm7,2)),P0(mm7,3)+(tt
1(mm7,:)*uu(mm7,3))];
end
punto1ii;
%MATRIZ OSSA4/INTERSECCION
for mm8=1:1:size(ossa4,1)

intersec1(mm8,:)=[fronteraii(mm8,:),punto1ii(mm8
,:)];
end
intersec1;
%DEFINIENDO MATRIZ CON PLANO INFERIOR Y SUPERIOR
CORTADO
marca102=0;
for mm9=1:1:size(intersec,1)
    if
intersec1(mm9,9)>=intersec1(mm9,3)&intersec1(mm9
,9)<=intersec1(mm9,6)|intersec1(mm9,9)<=intersec
1(mm9,3)&intersec1(mm9,9)>=intersec1(mm9,6)
        marca102=1;
        if intersec1(mm9,3)<planosuperior

fronterais(mm9,:)=[intersec1(mm9,1:3),intersec1(
mm9,7:9)];
        else

fronterais(mm9,:)=[intersec1(mm9,4:6),intersec1(
mm9,7:9)];
        end
    else
        marca102=2;

fronterais(mm9,:)=[intersec1(mm9,1:1:3),intersec
1(mm9,4:1:6)];
    end
end
fronterais;
%ELIMINAR LAS VIGAS QUE NO TIENEN INTERSECCION
CON EL PLANO DE CORTE PERO
%ESTAN SOBRE DICHO PLANO
marca103=0;
cont101=1;
for mm10=1:1:size(fronterais,1)
    if
fronterais(mm10,3)<=planosuperior&fronterais(mm1
0,6)<=planosuperior
        marca103=3;
    else
        marca103=1;
    end
    if marca103==3
        ossa2(cont101,:)=[fronterais(mm10,:)];
        cont101=cont101+1;
    end
end
ossa2;
%ELIMINAR LAS VIGAS QUE NO TIENEN INTERSECCION
CON EL PLANO DE CORTE PERO
%ESTAN DEBAJO DE DICHO PLANO
marca104=0;
cont102=1;
for mm11=1:1:size(ossa2,1)
    if
ossa2(mm11,3)>=planoinferior&ossa2(mm11,6)>=plan
oinferior
        marca104=3;
    else
        marca104=1;
    end
    if marca104==3
        ossa5(cont102,:)=[ossa2(mm11,:)];
        cont102=cont102+1;
    end
end
ossa5;
%ELIMINAR LOS SEGMENTOS DE RECTAS DEFINIDOS COMO
PUNTOS
marca105=0;
cont104=1;
for mm12=1:1:size(ossa5,1)
    if ossa5(mm12,1:1:3)==ossa5(mm12,4:1:6)
        marca105=1;
    else
        marca105=4;
    end
        if marca105==4
            ossa6(cont104,:)=[ossa5(mm12,:)];
            cont104=cont104+1;
        end
end
ossa6;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%CREAR MATRIZ PARA ADAPTAR CORTE DE PLANO X
%REINICIANDO VARIABLES
ossa7=ossa6;
mm=[];
mm1=[];
uu=[];
mm3=[];
tt=[];
mm2=[];
puntoii=[];
P0=[];
mm4=[];
intersec=[];
```

```
marca101=[];
mm5=[];
fronteraii=[];
mm6=[];
tt1=[];
mm7=[];
punto1ii=[];
mm8=[];
intersec1=[];
marca102=[];
mm9=[];
fronterais=[];
marca103=[];
cont101=[];
mm10=[];
ossa2=[];
marca104=[];
cont102=[];
mm11=[];
ossa5=[];
marca105=[];
cont104=[];
mm12=[];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PUNTO DE LA RECTA
for mm=1:1:size(ossa7,1)
    P0(mm,:)=[ossa7(mm,1:1:3)];
end
P0;
%VECTOR DIRECTOR DE LA RECTA
for mm1=1:1:size(ossa7,1)
    uu(mm1,:)=[ossa7(mm1,1)-
ossa7(mm1,4),ossa7(mm1,2)-
ossa7(mm1,5),ossa7(mm1,3)-ossa7(mm1,6)];
end
uu;
%PARAMETRO PARA DETERMINAR EL PUNTO DE
INTERSECCION INFERIOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
planoinferior1; %DEPENDE DEL PLANO QUE
INTERSECTA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for mm3=1:1:size(ossa7,1)
    tt(mm3,:)=[(planoinferior1-
P0(mm3,1))/uu(mm3,1)];
end
tt;
%PUNTO DE INTERSECCION
for mm2=1:1:size(ossa7,1)

puntoii(mm2,:)=[P0(mm2,1)+(tt(mm2,:)*uu(mm2,1)),
P0(mm2,2)+(tt(mm2,:)*uu(mm2,2)),P0(mm2,3)+(tt(mm
2,:)*uu(mm2,3))];
end
puntoii;
%MATRIZ OSSA7/INTERSECCION
for mm4=1:1:size(ossa7,1)

intersec(mm4,:)=[ossa7(mm4,:),puntoii(mm4,:)];
end
intersec;
%DEFINIENDO MATRIZ CON PLANO INFERIOR CORTADO
marca101=0;
for mm5=1:1:size(intersec,1)
    if
intersec(mm5,7)>=intersec(mm5,1)&intersec(mm5,7)
<=intersec(mm5,4)|intersec(mm5,7)<=intersec(mm5,
1)&intersec(mm5,7)>=intersec(mm5,4)
        marca101=1;
        if intersec(mm5,1)>planoinferior1

fronteraii(mm5,:)=[intersec(mm5,1:3),intersec(mm
5,7:9)];
        else

fronteraii(mm5,:)=[intersec(mm5,4:6),intersec(mm
5,7:9)];
        end
    else
        marca101=2;

fronteraii(mm5,:)=[intersec(mm5,1:1:3),intersec(
mm5,4:1:6)];
    end
end
fronteraii;
%PARAMETRO PARA DETERMINAR EL PUNTO DE
INTERSECCION SUPERIOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
planosuperior1; %DEPENDE DEL PLANO QUE
INTERSECTA SUPERIOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for mm6=1:1:size(ossa7,1)
    tt1(mm6,:)=[(planosuperior1-
P0(mm6,1))/uu(mm6,1)];
end
tt1;
%PUNTO DE INTERSECCION
for mm7=1:1:size(ossa7,1)

punto1ii(mm7,:)=[P0(mm7,1)+(tt1(mm7,:)*uu(mm7,1)
),P0(mm7,2)+(tt1(mm7,:)*uu(mm7,2)),P0(mm7,3)+(tt
1(mm7,:)*uu(mm7,3))];
end
punto1ii;
%MATRIZ OSSA4/INTERSECCION
for mm8=1:1:size(ossa7,1)

intersec1(mm8,:)=[fronteraii(mm8,:),punto1ii(mm8
,:)];
end
intersec1;
%DEFINIENDO MATRIZ CON PLANO INFERIOR Y SUPERIOR
CORTADO
marca102=0;
for mm9=1:1:size(intersec,1)
    if
intersec1(mm9,7)>=intersec1(mm9,1)&intersec1(mm9
,7)<=intersec1(mm9,4)|intersec1(mm9,7)<=intersec
1(mm9,1)&intersec1(mm9,7)>=intersec1(mm9,4)
        marca102=1;
        if intersec1(mm9,1)<planosuperior1

fronterais(mm9,:)=[intersec1(mm9,1:3),intersec1(
mm9,7:9)];
        else

fronterais(mm9,:)=[intersec1(mm9,4:6),intersec1(
mm9,7:9)];
        end
    else
        marca102=2;

fronterais(mm9,:)=[intersec1(mm9,1:1:3),intersec
1(mm9,4:1:6)];
    end
end
fronterais;
%ELIMINAR LAS VIGAS QUE NO TIENEN INTERSECCION
CON EL PLANO DE CORTE PERO
%ESTAN SOBRE DICHO PLANO
```

```
marca103=0;
cont101=1;
for mm10=1:1:size(fronterais,1)
    if
fronterais(mm10,1)<=planosuperior1&fronterais(mm
10,4)<=planosuperior1
        marca103=3;
    else
        marca103=1;
    end
    if marca103==3
        ossa2(cont101,:)=[fronterais(mm10,:)];
        cont101=cont101+1;
    end
end
ossa2;
%ELIMINAR LAS VIGAS QUE NO TIENEN INTERSECCION
CON EL PLANO DE CORTE PERO
%ESTAN DEBAJO DE DICHO PLANO
marca104=0;
cont102=1;
for mm11=1:1:size(ossa2,1)
    if
ossa2(mm11,1)>=planoinferior1&ossa2(mm11,4)>=pla
noinferior1
        marca104=3;
    else
        marca104=1;
    end
    if marca104==3
        ossa5(cont102,:)=[ossa2(mm11,:)];
        cont102=cont102+1;
    end
end
ossa5;
%ELIMINAR LOS SEGMENTOS DE RECTAS DEFINIDOS COMO
PUNTOS
marca105=0;
cont104=1;
for mm12=1:1:size(ossa5,1)
    if ossa5(mm12,1:1:3)==ossa5(mm12,4:1:6)
        marca105=1;
    else
        marca105=4;
    end
        if marca105==4
            ossa8(cont104,:)=[ossa5(mm12,:)];
            cont104=cont104+1;
        end
end
ossa8;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%CREAR MATRIZ PARA ADAPTAR CORTE DE PLANO Y
%REINICIANDO VARIABLES
ossa7=ossa6;
mm=[];
mm1=[];
uu=[];
mm3=[];
tt=[];
mm2=[];
puntoii=[];
P0=[];
mm4=[];
intersec=[];
marca101=[];
mm5=[];
fronteraii=[];
mm6=[];
tt1=[];
mm7=[];
punto1ii=[];
mm8=[];
intersec1=[];
marca102=[];
mm9=[];
fronterais=[];
marca103=[];
cont101=[];
mm10=[];
ossa2=[];
marca104=[];
cont102=[];
mm11=[];
ossa5=[];
marca105=[];
cont104=[];
mm12=[];
%ossa4=ossa7
%bones=ossa6
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PUNTO DE LA RECTA
for mm=1:1:size(ossa8,1)
    P0(mm,:)=[ossa8(mm,1:1:3)];
end
P0;
%VECTOR DIRECTOR DE LA RECTA
for mm1=1:1:size(ossa8,1)
    uu(mm1,:)=[ossa8(mm1,1)-
ossa8(mm1,4),ossa8(mm1,2)-
ossa8(mm1,5),ossa8(mm1,3)-ossa8(mm1,6)];
end
uu;
%PARAMETRO PARA DETERMINAR EL PUNTO DE
INTERSECCION INFERIOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
planoinferior2; %DEPENDE DEL PLANO QUE
INTERSECTA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for mm3=1:1:size(ossa8,1)
    tt(mm3,:)=[(planoinferior2-
P0(mm3,2))/uu(mm3,2)];
end
tt;
%PUNTO DE INTERSECCION
for mm2=1:1:size(ossa8,1)

puntoii(mm2,:)=[P0(mm2,1)+(tt(mm2,:)*uu(mm2,1)),
P0(mm2,2)+(tt(mm2,:)*uu(mm2,2)),P0(mm2,3)+(tt(mm
2,:)*uu(mm2,3))];
end
puntoii;
%MATRIZ OSSA7/INTERSECCION
for mm4=1:1:size(ossa8,1)

intersec(mm4,:)=[ossa8(mm4,:),puntoii(mm4,:)];
end
intersec;
%DEFINIENDO MATRIZ CON PLANO INFERIOR CORTADO
marca101=0;
for mm5=1:1:size(intersec,1)
    if
intersec(mm5,8)>=intersec(mm5,2)&intersec(mm5,8)
<=intersec(mm5,5)|intersec(mm5,8)<=intersec(mm5,
2)&intersec(mm5,8)>=intersec(mm5,5)
        marca101=1;
        if intersec(mm5,2)>planoinferior2

fronteraii(mm5,:)=[intersec(mm5,1:3),intersec(mm
5,7:9)];
        else
```

```
fronteraii(mm5,:)=[intersec(mm5,4:6),intersec(mm
5,7:9)];
        end
    else
        marca101=2;

fronteraii(mm5,:)=[intersec(mm5,1:1:3),intersec(
mm5,4:1:6)];
    end
end
fronteraii;
%PARAMETRO PARA DETERMINAR EL PUNTO DE
INTERSECCION SUPERIOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
planosuperior2; %DEPENDE DEL PLANO QUE
INTERSECTA SUPERIOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for mm6=1:1:size(ossa8,1)
    tt1(mm6,:)=[(planosuperior2-
P0(mm6,2))/uu(mm6,2)];
end
tt1;
%PUNTO DE INTERSECCION
for mm7=1:1:size(ossa8,1)

punto1ii(mm7,:)=[P0(mm7,1)+(tt1(mm7,:)*uu(mm7,1)
),P0(mm7,2)+(tt1(mm7,:)*uu(mm7,2)),P0(mm7,3)+(tt
1(mm7,:)*uu(mm7,3))];
end
punto1ii;
%MATRIZ OSSA4/INTERSECCION
for mm8=1:1:size(ossa8,1)

intersec1(mm8,:)=[fronteraii(mm8,:),punto1ii(mm8
,:)];
end
intersec1;
%DEFINIENDO MATRIZ CON PLANO INFERIOR Y SUPERIOR
CORTADO
marca102=0;
for mm9=1:1:size(intersec,1)
    if
intersec1(mm9,8)>=intersec1(mm9,2)&intersec1(mm9
,8)<=intersec1(mm9,5)|intersec1(mm9,8)<=intersec
1(mm9,2)&intersec1(mm9,8)>=intersec1(mm9,5)
        marca102=1;
        if intersec1(mm9,2)<planosuperior2

fronterais(mm9,:)=[intersec1(mm9,1:3),intersec1(
mm9,7:9)];
        else

fronterais(mm9,:)=[intersec1(mm9,4:6),intersec1(
mm9,7:9)];
        end
    else
        marca102=2;

fronterais(mm9,:)=[intersec1(mm9,1:1:3),intersec
1(mm9,4:1:6)];
    end
end
fronterais;
%ELIMINAR LAS VIGAS QUE NO TIENEN INTERSECCION
CON EL PLANO DE CORTE PERO
%ESTAN SOBRE DICHO PLANO
marca103=0;
cont101=1;
for mm10=1:1:size(fronterais,1)
    if
fronterais(mm10,2)<=planosuperior2&fronterais(mm
10,5)<=planosuperior2
        marca103=3;
    else
        marca103=1;
    end
    if marca103==3
        ossa2(cont101,:)=[fronterais(mm10,:)];
        cont101=cont101+1;
    end
end
ossa2;
%ELIMINAR LAS VIGAS QUE NO TIENEN INTERSECCION
CON EL PLANO DE CORTE PERO
%ESTAN DEBAJO DE DICHO PLANO
marca104=0;
cont102=1;
for mm11=1:1:size(ossa2,1)
    if
ossa2(mm11,2)>=planoinferior2&ossa2(mm11,5)>=pla
noinferior2
        marca104=3;
    else
        marca104=1;
    end
    if marca104==3
        ossa5(cont102,:)=[ossa2(mm11,:)];
        cont102=cont102+1;
    end
end
ossa5;
%ELIMINAR LOS SEGMENTOS DE RECTAS DEFINIDOS COMO
PUNTOS
marca105=0;
cont104=1;
for mm12=1:1:size(ossa5,1)
    if ossa5(mm12,1:1:3)==ossa5(mm12,4:1:6)
        marca105=1;
    else
        marca105=4;
    end
        if marca105==4
            ossa9(cont104,:)=[ossa5(mm12,:)];
            cont104=cont104+1;
        end
end
ossa9;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
planosuperior;
planoinferior;
planosuperior1;
planoinferior1;
planosuperior2;
planoinferior;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cont105=1;
for nn1=1:1:size(ossa9,1)
    element1=ossa9(nn1,1);
    element2=ossa9(nn1,6);
    element3=ossa9(nn1,3);
    element4=ossa9(nn1,4);
            if
element1==planosuperior1&element2==planosuperior
|element4==planosuperior1&element3==planosuperio
r

            else

ossa10(cont105,:)=[ossa9(nn1,:)];
                cont105=cont105+1;
```

```
            end
end
ossa10;
%%%%%%%
cont106=1;
for nn2=1:1:size(ossa10,1)
    element5=ossa9(nn2,3);
    element6=ossa9(nn2,5);
    element7=ossa9(nn2,2);
    element8=ossa9(nn2,6);
            if
element5==planosuperior&element6==planoinferior2
|element7==planoinferior2&element8==planosuperio
r

            else

ossa11(cont106,:)=[ossa10(nn2,:)];
                cont106=cont106+1;
            end
end
ossa11;
%%%%%%%
cont107=1;
for nn3=1:1:size(ossa11,1)
    element9=ossa11(nn3,3);
    element10=ossa11(nn3,4);
    element11=ossa11(nn3,1);
    element12=ossa11(nn3,6);
            if
element9==planosuperior&element10==planoinferior
1|element11==planoinferior1&element12==planosupe
rior

            else

ossa12(cont107,:)=[ossa11(nn3,:)];
                cont107=cont107+1;
            end
end
ossa12;
%%%%%%%
cont108=1;
for nn4=1:1:size(ossa12,1)
    element13=ossa12(nn4,3);
    element14=ossa12(nn4,5);
    element15=ossa12(nn4,2);
    element16=ossa12(nn4,6);
            if
element13==planosuperior&element14==planosuperio
r2|element15==planosuperior2&element16==planosup
erior

            else

ossa13(cont108,:)=[ossa12(nn4,:)];
                cont108=cont108+1;
            end
end
ossa13;
%%%%%%
cont109=1;
for nn5=1:1:size(ossa13,1)
    element17=ossa13(nn5,1);
    element18=ossa13(nn5,5);
    element19=ossa13(nn5,2);
    element20=ossa13(nn5,4);
            if
element17==planoinferior1&element18==planosuperi
or2|element19==planosuperior2&element20==planoin
ferior1

            else

ossa14(cont109,:)=[ossa13(nn5,:)];
                cont109=cont109+1;
            end
end
ossa14;
%%%%%%%%
cont110=1;
for nn6=1:1:size(ossa14,1)
    element21=ossa14(nn6,1);
    element22=ossa14(nn6,4);
    element23=ossa14(nn6,1);
    element24=ossa14(nn6,4);
            if
element21==planoinferior1&element22==planosuperi
or1|element23==planosuperior1&element24==planoin
ferior1

            else

ossa15(cont110,:)=[ossa14(nn6,:)];
                cont110=cont110+1;
            end
end
ossa15;
%%%%%%%
cont111=1;
for nn7=1:1:size(ossa15,1)
    element25=ossa15(nn7,1);
    element26=ossa15(nn7,5);
    element27=ossa15(nn7,2);
    element28=ossa15(nn7,4);
            if
element25==planoinferior1&element26==planoinferi
or2|element27==planoinferior2&element28==planoin
ferior1

            else

ossa16(cont111,:)=[ossa15(nn7,:)];
                cont111=cont111+1;
            end
end
ossa16;
%%%%%%%
cont112=1;
for nn8=1:1:size(ossa16,1)
    element29=ossa16(nn8,2);
    element30=ossa16(nn8,4);
    element31=ossa16(nn8,1);
    element32=ossa16(nn8,5);
            if
element29==planosuperior2&element30==planosuperi
or1|element31==planosuperior1&element32==planosu
perior2

            else

ossa17(cont112,:)=[ossa16(nn8,:)];
                cont112=cont112+1;
            end
end
ossa17;
%%%%%%%
cont113=1;
for nn9=1:1:size(ossa17,1)
    element33=ossa17(nn9,2);
    element34=ossa17(nn9,5);
    element35=ossa17(nn9,2);
```

```
    element36=ossa17(nn9,5);
            if
element33==planosuperior2&element34==planoinferi
or2|element35==planoinferior2&element36==planosu
perior2

            else

ossa18(cont113,:)=[ossa17(nn9,:)];
                cont113=cont113+1;
            end
end
ossa18;
%%%%%%%
cont114=1;
for nn10=1:1:size(ossa18,1)
    element37=ossa18(nn10,1);
    element38=ossa18(nn10,5);
    element39=ossa18(nn10,2);
    element40=ossa18(nn10,4);
            if
element37==planosuperior1&element38==planoinferi
or2|element39==planoinferior2&element40==planosu
perior1

            else

ossa19(cont114,:)=[ossa18(nn10,:)];
                cont114=cont114+1;
            end
end
ossa19;
%%%%%%%
cont115=1;
for nn11=1:1:size(ossa19,1)
    element41=ossa19(nn11,3);
    element42=ossa19(nn11,4);
    element43=ossa19(nn11,1);
    element44=ossa19(nn11,6);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Condición
revisar
            if
element41==planoinferior&element42==planosuperio
r1|element43==planosuperior1&element44==planoinf
erior
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            else

ossa20(cont115,:)=[ossa19(nn11,:)];
                cont115=cont115+1;
            end
end
ossa20;
%%%%%%%
cont116=1;
for nn12=1:1:size(ossa20,1)
    element45=ossa20(nn12,3);
    element46=ossa20(nn12,5);
    element47=ossa20(nn12,2);
    element48=ossa20(nn12,6);
            if
element45==planoinferior&element46==planoinferio
r2|element47==planoinferior2&element48==planoinf
erior

            else

ossa21(cont116,:)=[ossa20(nn12,:)];
                cont116=cont116+1;
            end
end
ossa21;
%%%%%%%
cont117=1;
for nn13=1:1:size(ossa21,1)
    element49=ossa21(nn13,3);
    element50=ossa21(nn13,4);
    element51=ossa21(nn13,1);
    element52=ossa21(nn13,6);
            if
element49==planoinferior&element50==planoinferio
r1|element51==planoinferior1&element52==planoinf
erior

            else

ossa22(cont117,:)=[ossa21(nn13,:)];
                cont117=cont117+1;
            end
end
ossa22;
%%%%%%%%
cont118=1;
for nn14=1:1:size(ossa22,1)
    element53=ossa22(nn14,3);
    element54=ossa22(nn14,5);
    element55=ossa22(nn14,2);
    element56=ossa22(nn14,6);
            if
element53==planoinferior&element54==planosuperio
r2|element55==planosuperior2&element56==planoinf
erior

            else

ossa23(cont118,:)=[ossa22(nn14,:)];
                cont118=cont118+1;
            end
end
ossa23;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
bones=[];
bones=ossa23;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ASIGNANDO NUMERO DE NODO PARA EXPORTAR A ABAQUS
othon=zeros(2*size(bones,1),4);
othon(1:size(bones,1),2:4)=[bones(:,1:3)];
othon(size(bones,1)+1:2*size(bones,1),2:4)=bones
(:,4:6);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nodoabaq=101; %nodoabaq DEPENDIENDE DEL TIPO DE
SISTEMA QUE SE ESTE UTILIZANDO nodoabaq COMIENZA
EN 101 O 201
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for aa=1:size(othon,1)
    bandera4=0;
    renglon4=othon(aa,2:4);
    for bb=aa-1:-1:1
        if renglon4==othon(bb,2:4)
            bandera4=1;
            repet=othon(bb,1);
        end
    end
    if bandera4==1
        othon(aa,1)=repet;
    else
        othon(aa,1)=nodoabaq;
        nodoabaq=nodoabaq+1;
    end
end
```

```
othon;
ossa=zeros(0.5*size(othon,1),8);
for aa1=1:1:0.5*size(othon,1)
    ossa(aa1,1:4)=othon(aa1,:);
end
ossa(1:0.5*size(othon,1),5:8)=othon(0.5*size(oth
on,1)+1:size(othon,1),:);
ossa;
othon;
%DETERMIINACIÓN DE LA MATRIZ DE NODOS
contador1=1;
for ii1=1:1:size(othon,1)
    renglon5=othon(ii1,:);
    rep5=0;
    for jj1=ii1:-1:1
        if renglon5==othon(jj1,:)
            rep5=rep5+1;
        end
    end
    if rep5==1
        nodos(contador1,:)=renglon5;
        contador1=contador1+1;
    end
end
nodos
%DETERMINANDO LA CONECTIVIDAD A PARTIR DE LOS
INDICE EN NOTACIÓN DE ABAQUS
ossa;
for x1=1:1:size(ossa,1)
    coneccion(x1,1)=ossa(x1,1);
end
for y1=1:1:size(ossa,1)
    coneccion(y1,2)=ossa(y1,5);
end
coneccion;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elementabaq=101;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for x2=1:1:size(coneccion,1)

elemento(x2,:)=[elementabaq,coneccion(x2,:)];
    elementabaq=elementabaq+1;
end
elemento
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%CALCULO DE LONGITUDES DE ELEMENTOS
ossa;
for n1=1:1:size(ossa,1)
longitudes(n1,:)=[((ossa(n1,2)-
ossa(n1,6))^2+(ossa(n1,3)-
ossa(n1,7))^2+(ossa(n1,4)-ossa(n1,8))^2)^(1/2)];
end
longitudes;
longitud_total=sum(longitudes);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%CONDICIONES DE FRONTERA "CARGA O
DESPLAZAMIENTO"
cont2=1;
for m2=1:1:size(nodos,1)
    renglon2=[nodos(m2,4)];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if renglon2==planosuperior %DEFINIENDO
LIMITE SUPERIOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            frontera1(cont2,:)=nodos(m2,1);
            cont2=cont2+1;
        end
end
frontera1;
for m3=1:1:size(frontera1,1)
    frontera(1,m3)=frontera1(m3,:);
end
frontera
%CONDICIONES DE FRONTERA "EMPOTRADO"
cont3=1;
for m4=1:1:size(nodos,1)
    renglon3=[nodos(m4,4)];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if renglon3==planoinferior %DEFINIENDO
LIMITE INFERIOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            frontera2(cont3,:)=nodos(m4,1);
            cont3=cont3+1;
        end
end
frontera2;
for m6=1:1:size(frontera2,1)
    boundary(1,m6)=frontera2(m6,:);
end
boundary
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%NODOS ADICIONALES EN DIRECCIÓN X
%LIMITE INFERIOR X
cont4=1;
for m7=1:1:size(nodos,1)
    renglon4=[nodos(m7,2)];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if renglon4==planosuperior1 %DEFINIENDO
LIMITE SUPERIOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            frontera3(cont4,:)=nodos(m7,1);
            cont4=cont4+1;
        end
end
frontera3;
for m8=1:1:size(frontera3,1)
    limiteinf_X(1,m8)=frontera3(m8,:);
end
limiteinf_X;
%LIMITE SUPERIOR X
cont5=1;
for m9=1:1:size(nodos,1)
    renglon5=[nodos(m9,2)];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if renglon5==planoinferior1 %DEFINIENDO
LIMITE INFERIOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            frontera4(cont5,:)=nodos(m9,1);
            cont5=cont5+1;
        end
end
frontera4;
for m10=1:1:size(frontera4,1)
    limitesup_X(1,m10)=frontera4(m10,:);
end
limitesup_X;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%NODOS ADICIONALES EN DIRECCIÓN y
%LIMITE INFERIOR Y
cont6=1;
for m11=1:1:size(nodos,1)
    renglon6=[nodos(m11,3)];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if renglon6==planosuperior2 %DEFINIENDO
LIMITE SUPERIOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            frontera5(cont6,:)=nodos(m11,1);
            cont6=cont6+1;
        end
end
frontera5;
for m12=1:1:size(frontera5,1)
```

```
    limiteinf_Y(1,m12)=frontera5(m12,:);
end
limiteinf_Y;
%LIMITE SUPERIOR Y
cont7=1;
for m13=1:1:size(nodos,1)
    renglon7=[nodos(m13,3)];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if renglon7==planoinferior2 %DEFINIENDO
LIMITE INFERIOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            frontera6(cont7,:)=nodos(m13,1);
            cont7=cont7+1;
        end
end
frontera6;
for m14=1:1:size(frontera6,1)
    limitesup_Y(1,m14)=frontera6(m14,:);
end
limitesup_Y;
%%%%%%%%%%%%%%%
limiteinf_X=limiteinf_X'
limitesup_X=limitesup_X'
limiteinf_Y=limiteinf_Y'
limitesup_Y=limitesup_Y'
%COORDENADAS DE LAS FRONTERAS
%%%%%LIMITES INFERIOR X
contador2=1;
for aa2=1:1:size(limiteinf_X,1)
    for aa3=1:1:size(nodos,1)
        mixteco=nodos(aa3,1);
        mexica=nodos(aa3,2:4);
        if mixteco==limiteinf_X(aa2,1)
            maya=1;

inferior_X(contador2,:)=[limiteinf_X(aa2,1),mexi
ca];
            contador2=contador2+1;
        else
            maya=2;
        end
    end
end
inferior_X
%%%%%
%%%%% LIMITE SUPERIOR X
contador3=1;
for aa3=1:1:size(limitesup_X,1)
    for aa4=1:1:size(nodos,1)
        mixteco2=nodos(aa4,1);
        mexica2=nodos(aa4,2:4);
        if mixteco2==limitesup_X(aa3,1)
            maya2=1;

superior_X(contador3,:)=[limitesup_X(aa3,1),mexi
ca2];
            contador3=contador3+1;
        else
            maya2=2;
        end
    end
end
superior_X
%%%%%%%
%%%%%%% LIMITE INFERIOR Y
contador4=1;
for aa4=1:1:size(limiteinf_Y,1)
    for aa5=1:1:size(nodos,1)
        mixteco3=nodos(aa5,1);
        mexica3=nodos(aa5,2:4);
        if mixteco3==limiteinf_Y(aa4,1)
            maya3=1;

inferior_Y(contador4,:)=[limiteinf_Y(aa4,1),mexi
ca3];
            contador4=contador4+1;
        else
            maya3=2;
        end
    end
end
inferior_Y
%%%%%%%%%% LIMITE SUPERIOR Y
contador5=1;
for aa5=1:1:size(limitesup_Y,1)
    for aa6=1:1:size(nodos,1)
        mixteco4=nodos(aa6,1);
        mexica4=nodos(aa6,2:4);
        if mixteco4==limitesup_Y(aa5,1)
            maya4=1;

superior_Y(contador5,:)=[limitesup_Y(aa5,1),mexi
ca4];
            contador5=contador5+1;
        else
            maya4=2;
        end
    end
end
superior_Y
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
longitud_total;
dim_Xinf=size(limiteinf_X,1);
dim_Xsup=size(limitesup_X,1);
dim_Yinf=size(limiteinf_Y,1);
dim_Ysup=size(limitesup_Y,1);
limitesup_X=limitesup_X';
limiteinf_Y=limiteinf_Y';
limitesup_Y=limitesup_Y';
limiteinf_X=limiteinf_X';
%%%%%%%%%%%%%%%%%%%%%%%%%%%% AÑADIR TEXTO AL INP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
intro = char('*Heading','** Job name: pruebacero
Model name: Model-1','*Preprint, echo=NO,
model=NO, history=NO, contact=NO','**','**
PARTS','**','*Part, name=alambre','*End
Part','**','**','** ASSEMBLY','**','*Assembly,
name=Assembly','**','*Instance, name=alambre-1,
part=alambre','*Node');
elem = char('*Element, type=B33, elset=VORO1');
section = char('** Section: seccion-a  Profile:
transversal-a','*Beam General Section,
elset=VORO1, poisson = 0.3,
section=CIRC','0.08','0.,0.,-
1.','1000.,384.6154','*End Instance','**');
nset_b = char('*Nset, nset=EMPOTRADO, internal,
instance=alambre-1');
nset_f = char('*Nset, nset=DESPLAZAMIENTO,
internal, instance=alambre-1');
nset_xinf = char('*Nset, nset=X_inf, internal,
instance=alambre-1');
nset_xsup = char('*Nset, nset=X_sup, internal,
instance=alambre-1');
nset_yinf = char('*Nset, nset=Y_inf, internal,
instance=alambre-1');
nset_ysup = char('*Nset, nset=Y_sup, internal,
instance=alambre-1');
pivote = char('*Nset, nset=pivote, internal,
instance=alambre-1');
materials = char('*End Assembly','**','**
MATERIALS','**','*Material, name=material-
a','*Elastic','1000. , 0.3','** ----------
```

```
---','**','** STEP: paso-a','**','*Step,
name=paso-a','*Static','0.1, 1., 1e-05,
0.1','**');
boundary_cond = char('** BOUNDARY
CONDITIONS','**','** Name: desplazamiento Type:
Displacement/Rotation','*Boundary','DESPLAZAMIEN
TO, 3, 3, -2.5','** Name: empotrado Type:
Symmetry/Antisymmetry/Encastre','*Boundary','EMP
OTRADO, 3,3,0','*Boundary','pivote,encastre','**
');
outputs = char('** OUTPUT
REQUESTS','**','*Restart, write,
frequency=0','**','** FIELD OUTPUT: F-Output-
1','**','*Output, field,
variable=PRESELECT','**','** HISTORY OUTPUT: H-
Output-1','**','*Output, history,
variable=PRESELECT','*End Step');
longitud = char('*longitud total');
Xinf = char('*Longitud Xinf');
Xsup = char('*Longitud Xsup');
Yinf = char('*Longitud Yinf');
Ysup = char('*Longitud Ysup');
%%%%%%%%%%%%%%%%%%%%%%%%   Exportar en formato INP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dlmwrite('CS_1_01.inp', intro,'')
dlmwrite('CS_1_01.inp', nodos,'-
append','roffset', 0, 'delimiter',
',','newline','pc')
dlmwrite('CS_1_01.inp', elem,'-
append','roffset', 1, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', elemento,'-
append','roffset', 0, 'delimiter',
',','newline','pc')
dlmwrite('CS_1_01.inp', section,'-
append','roffset', 1, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', nset_b,'-
append','roffset', 0, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', boundary,'-
append','roffset', 0, 'delimiter',
',','newline','pc')
dlmwrite('CS_1_01.inp', nset_f,'-
append','roffset', 1, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', frontera,'-
append','roffset', 0, 'delimiter',
',','newline','pc')
dlmwrite('CS_1_01.inp', nset_xinf,'-
append','roffset', 1, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', limiteinf_X,'-
append','roffset', 0, 'delimiter',
',','newline','pc')
dlmwrite('CS_1_01.inp', nset_xsup,'-
append','roffset', 1, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', limitesup_X,'-
append','roffset', 0, 'delimiter',
',','newline','pc')
dlmwrite('CS_1_01.inp', nset_yinf,'-
append','roffset', 1, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', limiteinf_Y,'-
append','roffset', 0, 'delimiter',
',','newline','pc')
dlmwrite('CS_1_01.inp', nset_ysup,'-
append','roffset', 1, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', limitesup_Y,'-
append','roffset', 0, 'delimiter',
',','newline','pc')
dlmwrite('CS_1_01.inp', pivote,'-
append','roffset', 1, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', materials,'-
append','roffset', 2, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', boundary_cond,'-
append','roffset', 0, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', outputs,'-
append','roffset', 0, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', longitud,'-
append','roffset', 3, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', longitud_total,'-
append','roffset', 0, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', Xinf,'-
append','roffset', 1, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', dim_Xinf,'-
append','roffset', 0, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', Xsup,'-
append','roffset', 1, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', dim_Xsup,'-
append','roffset', 0, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', Yinf,'-
append','roffset', 1, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', dim_Yinf,'-
append','roffset', 0, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', Ysup,'-
append','roffset', 1, 'delimiter',
'','newline','pc')
dlmwrite('CS_1_01.inp', dim_Ysup,'-
append','roffset', 0, 'delimiter',
'','newline','pc')
%%%%%%%%%%%%%%%%%%%%%%%%%%%% OSSA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dlmwrite('CS_1_01_OSSA.out', ossa, ',')
dlmwrite('inferior_X.out', inferior_X, ',')
dlmwrite('superior_X.out', superior_X, ',')
dlmwrite('inferior_Y.out', inferior_Y, ',')
dlmwrite('superior_Y.out', superior_Y, ',')
```