Simulación de Tejido Blando basada en el Método de los Elementos Finitos

Pablo Valera

3 de diciembre de 2002

UNIVERSIDAD DE CARABOBO

Facultad de Ciencias y Tecnología

Departamento de Computación

Simulación de Tejido Blando basada en el Método de los Elementos Finitos

Autor: Pablo J. Valera G.

Tutor: Prof. Carlos Cadenas

Cotutor: Dr. Guillermo Montilla

Trabajo Especial de Grado presentado ante la ilustre Universidad de Carabobo como credencial de mérito para optar al título de Licenciado en Computación.

Bárbula, 3 de diciembre de 2002

Resumen

En esta investigación se proporcionan un conjunto de estructuras de datos y subrutinas implementadas en el lenguaje de programación C++ para simular la deformación de tejido blando simple (venas). Se utiliza un modelo para deformaciones lineales sobre un material isotrópico homogéneo, donde se aproxima la geometría a un conjunto finito de elementos y se calcula la fuerza ejercida sobre cada partícula y sus vecinas a través del método de los elementos finitos, para después simular su evolución usando modelos incrementales iterativos. Cada subrutina implementada esta basada sobre el enfoque de la API Virtual Vision Machine (VVM), de la cual se aprovechan al máximo las diferentes funciones y estructuras para generar las visualizaciones.

A mi esposa, mi hijo y mis padres;

Por haber sido la mayor inspiración para la culminación de este trabajo y los mejores regalos que Dios me ha dado.

Agradecimiento

Resulta un poco difícil tratar de mencionar en estas pocas líneas a todas las personas que de una manera u otra contribuyeron con la realización de esta tesis, pero trataré de que no me falte nadie.

Para empezar quiero darle Gracias! a Dios, por haber estado siempre conmigo ayudándome a no flaquear, a no desesperarme, por haber insistido tanto en la Matemática y a enseñarme a creer que con fé todo se puede realizar (Jesus en tí Confio). A mi Esposa, Marilin por haberme ayudado tanto, por esas noches de trasnocho, por tu amor, tu animo, tu comprensión, por soportarme cuando me levantabas. A mi hijo, Pablo Daniel, por enseñarme que no debemos complicarnos la vida y que siempre tenemos que tener una sonrisa al frente aún después de haber llorado, por ser mi reloj despertador en las mañanas y por no dejarme dormir en las noches, te quiero muchachito. A mi papá, por su gran apoyo y sus consejos, sin tí viejito esto tampoco hubiera sido realidad. A mi mamá, por su forma de ser, su animo y su apoyo. A mi abuela Thais, por haber rezado tanto, estoy seguro que los Santos participaron en esto y todavia te debo la del cumpleaños. A mi abuelo Pablo y a mi abuela Luisa, esto también es suyo abuelos. A mi suegra, por su comprensión y su gran ayuda, realmente yo creía que las suegras eran malas. A mi familia y a la familia de mi esposa, por su apoyo. A mi tutor Carlos, por haber dedicado tantas horas para enseñarme y ayudarme, por el computador y los libros, por haber sido como un papá para mi en la Universidad que siempre me apoyo y me ayudó, por su comprensión y gran paciencia, tienes que sacar el Doctorado!. A el profesor Guillermo, por sus horas de consulta, su preocupación, su ayuda, su

apoyo, sus cafecitos y sus exigencias, en realidad me dejo mucho. A la profesora Desiree Delgado, por sus constantes ayudas y orientación. A la profesora Jackeline Loyo, por su personalidad, su ayuda, por haber entrado a mi seminario de avance aunque se me olvidó invitarla, por los caramelos en IO, vaya que sí comí, por haberme dejado tanto como profesora, la admiro. A el profesor Joel Rivas, por ser como es, sus chistes malos, su manera de explicar y de ver la vida, por haberme dejado tanto como profesor. A las profesoras Elsa Tovar y Mariana Souto, sus clases nunca se me olvidarán. Al profesor Saba Infante, por haberme enseñado que siempre debemos dar la pelea hasta el final aunque a veces creamos que es imposible. Al profesor Orestes por las consultas continuas de LaTex. A Dany por sus constantes consejos y ayudas, por su manera práctica de ver las cosas, vaya que me pudistes orientar. Al profesor Edwin Peña, por su ayuda académica. Al profesor Castañeda, por enseñarme que la perfección no existe y la terquedad sí. A Carla y León, por sus consultas en línea y los correos que tanto animo me dieron. A mi primo Hernán, por su ayuda allá en Caracas. A todo el personal del Centro de Procesamiento de Imágenes, por haberme ayudado y haberme tratado tan bien. A Adolfo, por sus múltiples segundas y el pase de estacionamiento. A el profesor Germán por sus consejos para la optimización del algoritmo. Al Consejo de Desarrollo Científico y Humanístico de la Universidad de Carabobo (CDCH) por la subvención para realizar esta investigación.

Gracias! a todos por haberme ayudado, cada uno tiene un granito de arena en esta tesis.

Pablo Valera

Índice general

Índice de cuadros

Índice de figuras

Introducción

La simulación computacional de objetos tridimensionales deformables ha sido estudiada en computación gráfica por más de dos décadas, de las cuales han resultado una amplia variedad de modelos para efectuar simulaciones, cada una con sus pro y sus contra, dependiendo del problema a resolver. En la mayoría de los casos se necesita hacer un análisis del comportamiento del sistema a través del tiempo, es decir, en tres dimensiones más el tiempo (llamado estudio en 4D), el cual es modelado en el ámbito físico-mecánico por una ó varias ecuaciones, llamadas ecuaciones constitutivas, porque dependen de las características y las propiedades del material. Las ecuaciones constitutivas suelen representar ecuaciones diferenciales que generalmente son resueltas haciendo uso de métodos de aproximación como diferencias finitas, el método de los elementos de frontera ó el método de los elementos finitos.

En este trabajo se proponen un conjunto de subrutinas para simular la deformación de venas, haciendo uso del método de los elementos finitos, con el cual se consiguen aproximar los valores de la fuerza ejercida sobre cada partícula del sólido y sus partículas vecinas.

En el primer capítulo se presenta una descripción del problema a resolver, los objetivos de esta investigación, su justificación y trabajos realizados sobre esta área anteriormente. Luego, en el capítulo dos se hará mención de algunos conceptos básicos de la mecánica y resistencia de materiales, para la comprensión del modelo utilizado, las ecuaciones planteadas y sus métodos de resolución, los cuales serán explicados de manera breve y poco formal, ya que la investigación engloba diversas áreas, que por su profundidad y diversidad, escapan del alcance del trabajo. En el capítulo tres se explicará la manera como se resolvió el problema presentado en el primer capítulo, así como también las subrutinas desarrolladas, para posteriormente mencionar en el cuarto capítulo los resultados y conclusiones obtenidos.

Capítulo 1

PLANTEAMIENTO DEL PROBLEMA

1.1. Planteamiento del Problema

El problema consiste en simular y deformar tejido blando. Este trabajo proveerá un módulo de software que actuará basicamente como una "caja negra", al que le entra un objeto sintético (un cilindro, una esfera, etc.) y del que sale el mismo objeto, pero ahora con propiedades elásticas. Este módulo estará orientado a una API desarrollada en la Universidad de Carabobo llamada *"Virtual Vision Machine"* (VVM). El VVM permite la reconstrucción del cuerpo de cualquier persona a través de la recolección de datos por tomografía axial computarizada y resonancia magnética nuclear, pero no permite visualizar las diferentes deformaciones que se pueden realizar a la piel (tejido blando) del individuo a través de la aplicación de una fuerza. Con el desarrollo de este módulo se persigue lograr en principio las deformaciones sobre "venas".

El desarrollo de este software antes mencionado será implementado bajo el lenguaje C++ y estará basado en el Método de los Elementos Finitos (MEF). La selección de este método está justificada por su amplio uso en la mayoría de los trabajos realizados sobre esta área, dando los mejores resultados, por permitir un mejor desempeño en tiempo de ejecución de los mismos y lograr deformaciones realísticas apreciables, criterios principales para la evaluación de trabajos en esta área.

El procedimiento general consistirá en observar las diversas estructuras de tejido blando como un espacio continuo, describiendo su evolución a través de la teoría mecánica continua, aproximando esta geometría a un conjunto finito de elementos, y simulando esta evolución usando procedimientos incrementales iterativos. Partiendo de esta idea, las propiedades mecánicas de tal material deberán ser provistas en la forma de una relación constitutiva esfuerzo-deformación.Tomando en cuenta que el tejido blando es considerado: no lineal y un material visco-elástico de gran deformación.

Antes de continuar es preciso mencionar que de aquí en adelante cuando nos refiramos a la frase "tejido blando"nos estaremos refiriendo a tejido blando simple (venas).

1.2. Objetivo General y Específicos

1.2.1. Objetivo General

Desarrollar un módulo de software que permita la simulación de tejido blando simple (venas) a través del uso del método de los elementos finitos para la API

"Virtual Vision Machine"(VVM).

1.2.2. Objetivos Específicos

1. Estudiar las diversas variantes del método de los elementos finitos utilizadas en la modelación y simulación de tejido blando.

   a) Analizar las ventajas y desventajas de cada uno de estos métodos.

   b) Analizar las distintas implementaciones realizadas sobre el método de los elementos finitos a nivel computacional.

   c) Comparar la eficiencia y rendimiento de cada una de las implementaciones encontradas, basándose en antecedentes anteriores.

2. Estudiar las diferentes estructuras de datos utilizadas para crear imágenes tridimensionales sobre aplicaciones.

   a) Analizar las ventajas y desventajas de cada una de estas estructuras.

   b) Comparar la eficiencia y rendimiento de cada una estas estructuras basándose en antecedentes anteriores.

3. Diseñar cada una de las estructuras de datos a utilizar para la implementación de las subrutinas de simulación pertenecientes al módulo.

4. Diseñar un modelo matemático a través del método de los elementos finitos que permita realizar las subrutinas para la simulación de tejido blando.

5. Diseñar e implementar cada una de las subrutinas para la deformación de tejido blando sobre la aplicación VVM.

6. Realizar diferentes pruebas a cada una de las subrutinas implementadas, para observar sus resultados.

1.3. Justificación

El desarrollo de este módulo aportaría a la API VVM del Centro de Procesamiento de Imágenes(CPI) de la Universidad de Carabobo, una nueva estructura de datos que permitiría la creación de imágenes a partir de arreglos no estructurados de sólidos (volúmenes), con los cuales se podrían realizar simulaciones de fluído, deformación, comportamiento de gases y transmisión de calor. Además, se aportarían nuevas herramientas para la simulación de deformaciones que permitirían apreciar la deformación que ofrece un sólido cuando le es aplicada una fuerza.

Las herramientas de simulación y manipulación de tejido blando pueden servir para una variedad de propósitos. En nuestro contexto, pueden ser usadas para especificar composición de tejido y su relación, por ejemplo, músculos y ligamentos podrían ser creados y conectados para especificar partes sin huesos y dar propiedades que caracterizarían su comportamiento.

La mayoría de las aplicaciones que utilizan imágenes médicas podrían utilizar estas herramientas para facilitar la investigación y experimentación. Estas podrían incluir diagnósticos de anormalidades (estructural y de movimiento), investigación quirúrgica interactiva, examinación de terapias ortopédicas, diseño y simulación de prótesis. Adicionalmente a esto, estas herramientas podrían ser usadas en la educación para la visualización del tejido humano y en el entretenimiento para el

desarrollo de juegos.

En la gran mayoría de las ramas de la ciencia médica, es utilizada la imagenología como complemento fundamental a la hora de establecer un diagnóstico. Con los avances de la tecnología se han ido creando nuevas técnicas radiográficas con las cuales no sólo se consigue observar el tejido óseo, sino que también se puede observar tejido blando con una exactitud mayor, hablamos de las tomografías y las resonancias, con las cuales se logran imágenes tridimensionales.

Sabemos que las radiografías son de gran ayuda para la realización de cualquier actividad quirúrgica, pero las mismas presentan un margen de error considerable si se habla en milímetros; ahora bien, la idea es conseguir que dicho margen sea mínimo, para obtener imágenes confiables, lo que se obtiene por medio de imágenes tridimensionales (tridimensionalidad virtual y real).

Sería fantástico poder realizar cualquier tratamiento quirúrgico dos veces, primero sobre un modelo o réplica exacta de la zona a operar (VIRTUAL) y una vez satisfecho con los resultados obtenidos, realizarlo sobre el paciente (REAL); al lograr esto, aplicaríamos un factor importante en cada tratamiento aunque no sea quirúrgico, que es la PREVISIÓN.

1.4. Antecedentes

La amplia demanda de visualización y creación de imágenes científicas, esta proporcionando técnicas y capacidades avanzadas para la adquisición, procesamiento,

visualización y análisis de imágenes biomédicas para habilitar la información confiable de información clínica y científica. Se han realizado considerables esfuerzos sobre esta vertiente en la última década. Estos esfuerzos han sido primeramente direccionados a la visualización y despliegue (vistas tridimensionales de órganos humanos) de la información obtenida de diversos medios, tales como: Tomografía Axial Computarizada (TAC) y Resonancia Magnética Nuclear (RMN).

En la actualidad existen muchos grupos trabajando sobre la simulación y reconstrucción por computadora de tejido blando humano a partir de datos tomográficos, tanto en Universidades como en instituciones privadas. A continuación se mencionan algunos trabajos publicados en esta área:

- En 1978, Z. Wesolowski [30], publica un trabajo explicando la elasticidad de los cuerpos a través de la dinámica no lineal usando el Método de los Elementos Finitos (MEF).
- En 1982, F. Parke [22], publica un trabajo que propone un modelo parametrizado para la animación de imágenes faciales haciendo uso del MEF.
- En 1993, fue iniciado un proyecto de investigación por la comisión Europea llamado CHARM. El objetivo de éste es desarrollar un modelo de recurso de animación para la compresión del cuerpo humano (Comprehensive Human Animation Resource Model) que permita la reconstrucción del cuerpo humano a través de imágenes médicas y simulación dinámica de esta compleja músculo-esquelética estructura. Incluyendo la simulación de con-

tracción muscular y la deformación de tejido blando a través del Método de Elementos Finitos (MEF).

- En 1995, P. Kalra y otros [15] describen una metodología utilizada para modelar la topología anatómica humana y cómo fue empleada ésta sobre el modelado topológico presentado por los mismos.

- En 1996, P. Gingins y otros [13] describen una metodología y herramientas a utilizar para la construcción de un Modelo Humano Comprensible a partir de un conjunto visible de características humanas (Visible Human Dataset ó VHD) concentrando su esfuerzo específicamente en la parte superior izquierda del cuerpo.

- En 1998, W. Maurel y otros [18] publican un libro en el que realizan una síntesis de todos los modelos biomécanicos conocidos en la simulación realística de músculos humanos, tendones y piel, basados en la teoría mecánica y el método de los elementos finitos.

- En 2000, W. Maurel y otros [16] presentan un trabajo para mejorar la modelación del hombro humano a través del uso de la teoría cinemática inversa y el MEF, basándose en investigaciones biomecánicas realizadas. Este trabajo permitía apreciar las diferentes deformaciones y movimientos que describía un hombro humano a través de la coacción scapulo toráxica.

  Como se ha visto existen muchas investigaciones sobre este tema y podemos encontrar muchas más en la red ó en la referencia bibliográfica citada más adelante. Sin embargo, a través de las múltiples investigaciones realizadas

se encontró que en latinoamérica y más específicamente en nuestro país (Venezuela), existen muy pocos trabajos que han contribuido a esta área.

Actualmente, en la Universidad de Carabobo se está desarrollando una API llamada VVM la cual sigue dicha vertiente. Esta puede reconstruir diversas partes del cuerpo humano utilizando como entrada los datos de la tomografía de una persona. Sin embargo, la aplicación carece de herramientas en la simulación de tejido blando. Es por esta razón que la investigación presente contribuiría al enriquecimiento de esta API y por ende al enriquecimiento de investigaciones en esta área a nivel nacional, por ser dicha aplicación una de las pocas creadas en Venezuela.

Capítulo 2

MARCO TEÓRICO

Los computadores han resultado una herramienta indispensable en la modelación y simulación. Tan pronto como se incrementa el poder computacional, los usuarios y aplicaciones demandan incrementar aún más los niveles de realismo sobre estas áreas. Esta tendencia se presenta particularmente en computación gráfica donde las más sofisticadas formas geométricas y objetos físicos están siendo modelados en el contexto de ambientes físicos complejos.

En particular, la habilidad para modelar y manipular objetos deformables es esencial para muchas aplicaciones, por ello es importante saber escoger el modelo a utilizar de acuerdo a nuestros requerimientos. Para algunas aplicaciones es muy importante precisar la exactitud y realismo de la simulación que se esté haciendo; sin embargo, para otras podría ser más importante la rapidez en que se presentan los resultados de tal simulación. Algunas veces tendremos que sacrificar tiempo por exactitud, dependiendo del tipo de aplicación que estemos desarrollando. Generalmente los modelos que ofrecen mayor rapidez en tiempo de respuesta ofrecen

menor definición y viceversa.

En este capítulo se introducen algunos conceptos básicos de mecánica continua y análisis estructural, para después describir las propiedades del tejido blando y los modelos que se pueden utilizar para modelar el mismo. Finalmente, se mostrarán las formulaciones del modelo que permite realizar las deformaciones, como los métodos empleados para resolver el mismo.

2.1. Conceptos Básicos

2.1.1. Fuerza Axial

Corresponde a la acción de tirar (o de empujar) sobre la sección. Tirar (o jalar) representa una fuerza de extensión o tracción que tiende a alargar el sólido, mientras que empujar representa una fuerza de compresión que tiende a acortarlo. Se representa generalmente por P.

2.1.2. Fuerzas Cortantes

Son componentes de la resistencia total al deslizamiento de una porción del sólido a un lado de la sección de exploración respecto de la otra sección. La fuerza cortante se suele representar por V y sus componentes, V_x,V_y,V_z, determinan su dirección y sentido.

2.1.3. Esfuerzo

A nivel general podemos ver dos tipos de esfuerzos:

Esfuerzo Simple

La fuerza por unidad de área que soporta un material se suele denominar *esfuerzo* en el material, y se expresa matemáticamente en la forma:

$$\sigma = \frac{P}{A} \tag{2.1}$$

en donde σ es el esfuerzo o fuerza por unidad de área, P es la carga aplicada y A es el área de la sección transversal. Sin embargo, hasta una expresión tan sencilla como la anterior requiere un cuidadoso examen. Dividiendo la carga entre el área de la sección no se obtiene el valor del esfuerzo en todos los puntos de aquélla, sino solamente el valor medio del esfuerzo. Una determinación más exacta del esfuerzo exige dividir la fuerza diferencial dP entre el elemento de área diferencial sobre el que actúa y escribir:

$$\sigma = \frac{dP}{dA} \tag{2.2}$$

La situación en la que el esfuerzo es constante uniforme se llama estado de esfuerzo simple. Una distribución uniforme de esfuerzos sólo puede existir si la resultante de las fuerzas aplicadas pasa por el centroide o centro de gravedad de la sección considerada. No se puede inferir, sin embargo, que sí la fuerza es tal que su línea de acción pasa por el centroide de la sección, resulte siempre una distribución uniforme de esfuerzos.

Esfuerzo Cortante

El esfuerzo cortante (o de cizallamiento), a diferencia del axial (o de tensión o compresión), es producido por fuerzas que actúan paralelamente al plano que las resiste, mientras que los de tensión o de compresión lo son por fuerzas normales al plano sobre el que actúan. Por esta razón, los esfuerzos de tensión y de compresión se llaman también esfuerzos normales, mientras que el esfuerzo cortante puede denominarse esfuerzo tangencial. Aparecen esfuerzos cortantes siempre que las fuerzas aplicadas obliguen a que una sección del sólido tienda a deslizar sobre la sección adyacente. Puede existir esfuerzo cortante uniforme si la fuerza cortante resultante pasa por el centroide de la sección sometida al esfuerzo cortante. Si ocurre así, el esfuerzo de corte viene dado por

$$\tau = \frac{V}{A} \tag{2.3}$$

En realidad, la distribución del esfuerzo cortante en una sección no es uniforme prácticamente en ningún caso y por ello la expresión debe interpretarse solamente como el esfuerzo cortante medio. Esto no restringe su empleo en modo alguno, siempre que el valor del esfuerzo cortante admisible para un material dado tenga en cuenta este hecho de que la distribución real no es uniforme. Además, cuando la distancia entre las fuerzas que la producen sea muy pequeña, o el ancho de la sección que lo soporta sea igualmente pequeño, la distribución del esfuerzo cortante tiende a ser uniforme.

2.1.4. Deformación

También en las deformaciones podemos encontrarnos con dos tipos:

Deformación Simple

El valor de la deformación (unitaria) ϵ es el cociente del alargamiento (deformación total) δ y la longitud L en la que se ha producido. Por tanto,

$$\epsilon = \frac{\delta}{L} \tag{2.4}$$

Sin embargo, de este modo sólo se obtiene el valor medio de la deformación. La expresión correcta de la deformación en cualquier punto es

$$\epsilon = \frac{d\delta}{dL} \tag{2.5}$$

que determina el valor de la deformación en una longitud tan pequeña (dL) que puede considerarse constante en dicha longitud. No obstante, en ciertas condiciones, se puede suponer que la deformación es constante y aplicar la expresión (2.4). Estas condiciones son:

1. El elemento sometido a tensión debe tener una sección transversal constante.
2. El material debe ser homogéneo.
3. La fuerza o carga debe ser axial.

Por último, obsérvese que, como la deformación representa un cambio de longitud dividida entre la longitud inicial, la deformación es una cantidad sin dimensiones.

No obstante, cuando se habla de deformaciones se emplean unidades de metro por metro (m/m). En la práctica es frecuente encontrar deformaciones del orden de 1.0 x 10^{-3} m/m.

Deformación Cortante

Las fuerzas cortantes producen una deformación angular o distorsión, de la misma manera que las fuerzas axiales originan deformaciones longitudinales, pero con una diferencia fundamental. Un elemento sometido a tensión experimenta un alargamiento, mientras que un elemento sometido a una fuerza cortante no varía la longitud de sus lados, manifestándose por el contrario un cambio de forma.

El proceso puede imaginarse como producido por el desplazamiento infinitesimal o resbalamiento de capas infinitamente delgadas del elemento unas sobre otras, siendo la suma de estos infinitos resbalamientos infinitesimales la deformación transversal total δ, en una longitud L.

La deformación angular media se obtiene dividiendo δ entre L. Por tanto

$$\gamma = \frac{\delta}{L} \tag{2.6}$$

2.1.5. Diagrama Esfuerzo-Deformación

Un diagrama esfuerzo-deformación se representa por las deformaciones axiales en el eje horizontal y los esfuerzos correspondientes en las ordenadas. Estos diagramas ilustran el comportamiento de diversos materiales a medida que se les

va aplicando una carga, o son cargados, a tracción. Un ejemplo de un diagrama esfuerzo-deformación puede ser visto en la figura 2.1, obtenido de [14].

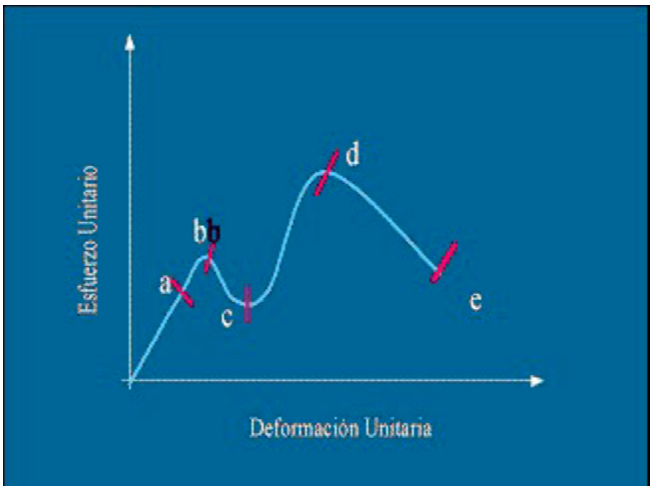


Figura 2.1: Ejemplo de un Diagrama Esfuerzo-Deformación

Existen cinco propiedades que podemos observar en un diagrama esfuerzo-deformación:

Límite de Proporcionalidad

El límite de proporcionalidad es el punto final del segmento de recta que se presenta en el diagrama esfuerzo-deformación. Ocurre cuando el esfuerzo deja de ser proporcional a la deformación, y deja de aplicarse la tan conocida relación de proporcionalidad enunciada en el año 1678 por Robert Hooke [14].

Límite de Elasticidad

Es el esfuerzo más allá del cual el material no recupera totalmente su forma original al ser descargado, sino que queda con una deformación residual llamada deformación permanente.

Punto de Fluencia

Es aquél en el que aparece un considerable alargamiento o fluencia del material sin el correspondiente aumento de carga, que incluso puede disminuir mientras dura la fluencia.

El Esfuerzo Último

Es la máxima ordenada de la curva esfuerzo deformación. Al haber más alargamiento el esfuerzo aumenta y alcanza su valor máximo en un punto de la curva, más allá de este punto el alargamiento adicional esta acompañado por una reducción en la carga y la ruptura del material.

Punto de Ruptura

Es el que ocurre próximo a tener lugar la ruptura, el material se alarga muy rápidamente y al mismo tiempo se estrecha en una parte muy localizada, de forma que la carga, en el instante de la ruptura, se distribuye realmente sobre una sección mucho más pequeña. Suele ser medido dividiendo la carga entre el área inicial de la barra.

2.1.6. Módulo de Elasticidad

El módulo de elasticidad se llama a veces módulo de Young, en honor al científico inglés Thomas Young (1773-1829) que estudió el comportamiento elástico de barras.

$$E = \frac{\sigma}{\varepsilon} \tag{2.7}$$

Este módulo es la pendiente de la gráfica deformación-esfuerzo en la región linealmente elástica y es diferente para varios materiales. Para la mayor parte de los materiales el módulo de elasticidad a la compresión es el mismo que a la tracción. En los cálculos, el esfuerzo y la deformación de tracción suelen considerarse positivos y el esfuerzo y la deformación de compresión, negativos.

2.1.7. Módulo de Poisson

Cuando una barra se carga a tracción, el alargamiento axial es acompañado por una contracción lateral, esto es, el ancho de la barra se hace menor a medida que su longitud aumenta. La razón de la deformación en dirección lateral a la deformación en dirección axial o longitudinal, es constante dentro del intervalo elástico y se conoce por *relación de Poisson* v; así pues,

$$v = \frac{\text{deformación axial}}{\text{deformación lateral}} \tag{2.8}$$

Esta constante recibe su nombre en honor al matemático francés S.D. Poisson (1781-1840), quién intentó calcular esa relación por una teoría molecular de los materiales.

El cambio del volumen de una barra sometida a una tracción puede calcularse sí se conocen la relación de Poisson y el módulo de elasticidad del material.

2.1.8. Los Tensores y su significado

Un tensor puede ser visto como una definición alternativa de un vector de N componentes que se transforman de acuerdo a una ley para llevarlas a su equivalente en otro sistema. Las componentes pueden ser vistas como un conjunto de n variables $x_1, x_2, x_3..., x_n$ que son transformadas a un nuevo conjunto de variables $\bar{x}_1, \bar{x}_2, \bar{x}_3, ..., \bar{x}_n$ a través de la siguiente ecuación:

$$\bar{x}_i = f_i(x_1, x_2, x_3..., x_n) \tag{2.9}$$

o viceversa,

$$x_i = g_i(\bar{x}_1, \bar{x}_2, \bar{x}_3, ..., \bar{x}_n) \tag{2.10}$$

Cuando un tensor posee 3 componentes, se dice es de primer orden, a nivel general cuando el orden de un tensor es mayor que 1, deja de ser un vector y se convierte en una matriz de $(O + 1) * 3$ componentes, donde O es el orden del tensor.

Las propiedades de los tensores vienen dadas de la siguiente manera:

Igualdad: dos tensores T y S que pertenecen a un espacio vectorial V^n, son iguales sí y sólo sí,

$$Sv = Tv \quad \forall v \epsilon V^n \tag{2.11}$$

Suma: dados T y S que pertenecen a un espacio vectorial V^n, la suma $T + S$ que

pertenece a V^n se define como,

$$(S+T)v = Sv + Tv \quad \forall v \epsilon V^n \tag{2.12}$$

Producto por un Escalar: dado S que pertenece a un espacio vectorial V^n y α que pertenece $\mathbb{R}$, se define el producto αS que pertenece a V^n por,

$$(\alpha S)v = \alpha(Sv) \quad \forall v \epsilon V^n \tag{2.13}$$

La importancia del análisis tensorial puede ser vista sobre las ecuaciones. La forma de una ecuación puede tener validez general con respecto a cualquier punto de referencia sólo sí todo término en la ecuación tiene las mismas características tensoriales. Sí esta condición no es satisfecha, un simple cambio en el sistema de referencia destrozaría la forma de la relación. Además, el análisis dimensional expresa que dos cantidades físicas no pueden ser iguales a menos que estas tengan las mismas dimensiones; cualquier ecuación física no puede ser correcta a menos que esta sea invariante con respecto al cambio fundamental de sus unidades.

El Tensor de Green-St. Venant

A nivel general existen muchos tensores para modelar el comportamiento de algunos fenómenos físicos, entre ellos podemos citar: el tensor de Almansi para esfuerzos finitos, el tensor de Euclides, el tensor de Cauchy para esfuerzos infinitos y el tensor de Green - Saint Venant entre otros. Este último tensor, mide el cambio en la longitud de los elementos de un material de tipo lineal y el cambio

en los ángulos de este objeto con respecto al sistema de referencia, es decir, su rotación.

El Tensor de Green-Saint Venant esta definido de la siguiente manera:

$$E_{ij} = \frac{1}{2}\left[\frac{\partial u_j}{\partial a_i} + \frac{\partial u_i}{\partial a_j} + \frac{\partial u_j}{\partial a_i}\frac{\partial u_i}{\partial a_j}\right] \tag{2.14}$$

o,

$$E = \frac{1}{2}\left[\nabla U + \nabla U^t + \nabla U \nabla U^t\right] \tag{2.15}$$

donde U representa un vector de desplazamiento, a el eje sobre el cual se este derivando U y ∇ el gradiente de un vector. Este tensor es utilizado para modelar grandes desplazamientos de materiales hiperelásticos. Sin embargo, existe una versión de este mismo tensor linealizada para materiales homogéneos isotrópicos que se define de la siguiente manera:

$$E = \frac{1}{2}\left[\nabla U + \nabla U^t\right] \tag{2.16}$$

el cual es utilizado para modelar pequeños desplazamientos y deformaciones, típicamente menores al 10 por ciento del tamaño de la malla.

2.2. Biomecánica del Tejido Blando

2.2.1. Fisiología

El tejido blando puede ser distinguido de otros tejidos del cuerpo por su flexibilidad y propiedades mecánicas. Todo el tejido blando esta esencialmente compuesto de colágeno. Informes han reportado que el colágeno representa entre el 75 por

ciento del peso seco de los tendones humanos. El peso restante es compartido entre elastina, reticulina, y gel hidrofólico.

Este tejido se caracteriza por contener células y también sustancias extracelulares, en su mayor parte secretadas por uno de los tipos celulares (los fibroplastos) y que, en condiciones normales representan una proporción del tejido mayor que las células. En conjunto, las sustancias extracelulares se denominan matriz extracelular, compuestas por fibras incluidas en una matriz amorfa que contiene líquido celular. Las fibras del tejido blando se dividen en tres tipos, fibras de colágeno, reticulares y elásticas. La matriz amorfa está compuesta por glucosaminoglucanos y proteoglucanos que forman geles muy hidratados en los cuales están incluidos los demás componentes. En la matriz celular también hay glucoproteínas adhesivas, como por ejemplo fibronectina y laminina.

Como quiera que sea, el tejido blando exhibe grandes diferencias en sus propiedades mecánicas. Esta observación ha permitido llegar a la conclusión que las propiedades del tejido blando son debido a su estructura más que a su cantidad relativa de constituyentes. Existen muchos tipos de tejido blando envueltos en los movimientos del cuerpo y las deformaciones, por ejemplo, tendones, ligamentos, piel y venas.

2.2.2. Venas

Las venas conducen la sangre de regreso al corazón. Por lo general acompañan las arterias correspondientes, pero tienen mayores diámetros. A menudo una arteria

es acompañada por varias venas, que drenan la sangre de la zona irrigada por la arteria. La superficie transversal ocupada por el conjunto de estas venas se caracteriza por ser mucho mayor que la arterial. Las venas también tienen paredes mucho más delgadas que las arterias del mismo tamaño, lo cual se debe considerar a la luz de la presión venosa mucho menor. La presión hidrostática algo superior en las extremidades inferiores, comparada con las extremidades superiores, se refleja en el espesor de la pared venosa, que por lo general es algo más gruesa en las venas de las extremidades inferiores. Por último, la pared contiene más tejido conectivo en las arterias pero las venas en humanos son bastante ricas en tejido elástico, por lo que se dilatan con relativa facilidad. Las venas suelen agruparse en grandes, medianas y pequeñas. Las pequeñas tienen un diámetro de 0,1-1 mm, mientras que las medianas varían entre 1 y 10 mm e incluyen, por ejemplo, la mayoría de las venas superficiales y profundas de los brazos y las piernas. En las grandes se incluyen todas las de diámetro superior a 10 mm, por ejemplo, las venas cava.

2.2.3. Propiedades Mecánicas del Tejido Blando

Elasticidad No Lineal

Bajo una tensión uniaxial paralela al tejido de las fibras de colágeno se observa una relación no lineal esfuerzo-deformación caracterizada por una región de un módulo de elasticidad inicial bajo, una región intermedia de incremento gradual en el módulo, una región de maximización del módulo, y una región de decrecimiento del módulo antes de completar la ruptura del tejido, situación que puede ser ilustrada por la figura 2.2, obtenida de [18]. La región de bajo módulo es atribuída

al traslado de las ondulaciones de las fibras de colágeno que normalmente existen en el tejido relajado. Como las fibras comienzan a resistir la carga de tensión, el módulo del tejido se incrementa. Cuando todas las fibras resultan cargadas, el módulo del tejido alcanza su máximo valor, y de allí en adelante, el esfuerzo de tensión se incrementa linealmente con el incremento de la deformación. Con una carga adicional, los grupos de fibras comienzan a fallar, causando el decrecimiento del módulo hasta completar la ruptura del tejido.

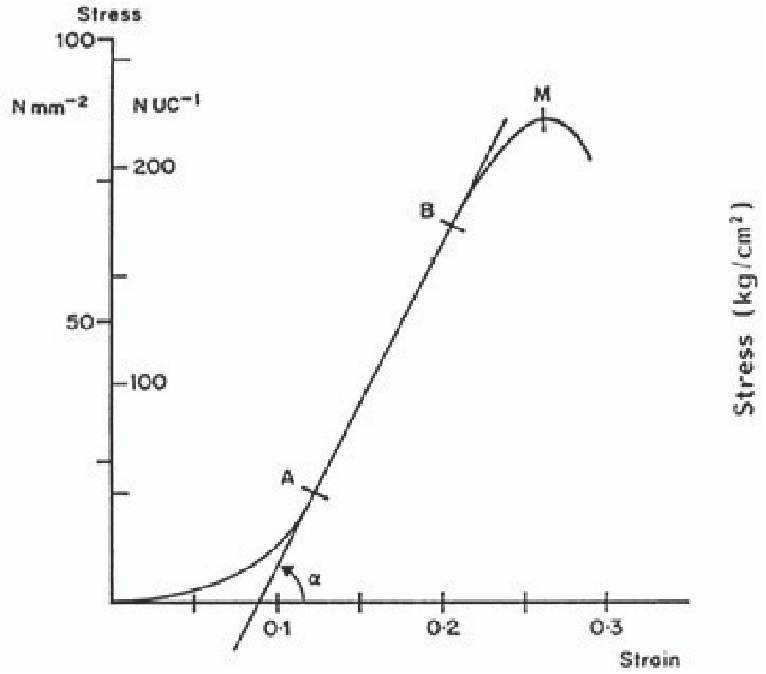


Figura 2.2: Diagrama Esfuerzo-Deformación Tejidos Blandos

Viscoelasticidad

Todos los experimentos relacionados acerca de este punto revelan una relación entre el esfuerzo y la deformación en el caso estático, es decir, cuando el cuerpo

está en equilibrio, pero cuando el mismo no es alcanzado, existe un componente dependiente de la historia en el estudio del comportamiento mecánico del tejido. Al medir una extensión dinámica y una estática con la misma deformación, encontramos que el esfuerzo resultante en el estudio dinámico resultó mayor que el esfuerzo en el estudio estático, aunque ambas mediciones hayan tenido la misma deformación. En general, cuando el esfuerzo en cualquier instante del tiempo no depende sólo de la deformación en el tiempo, sino también de la "historia"de la misma, podemos apreciar el fenómeno físico de la viscoelasticidad.

Otras Propiedades

La compresibilidad del tejido blando ha sido investigada muy poco, usualmente se asume que este es un material incompresible. Cuando los ciclos de carga-descarga son aplicados sobre el tejido sucesivamente al mismo nivel de esfuerzo, la curva esfuerzo-deformación se mueve gradualmente hacia la derecha. Después de un número de ciclos, la respuesta mecánica del tejido entra en una fase estacionaria y el resultado se convierte en reproducible de un ciclo al siguiente. Este fenómeno es debido a los cambios que ocurren en la estructura interna del tejido, hasta que un estado estable es alcanzado. Esta fase inicial de comportamiento común de todos los tejidos vivos es usualmente usada como una *precondición del tejido* anterior a la experimentación.

2.3. Modelos de Deformación

A nivel general, de acuerdo a las características que exhiba un material podemos utilizar diferentes modelos. Básicamente estos modelos pueden ser divididos en dos clases:

- Modelos Lineales.
- Modelos No Lineales.

2.3.1. Modelos Lineales

Son los que están basados en la hipótesis de comportamiento elástico lineal de los materiales constituyentes y en la consideración del equilibrio de la estructura sin deformar.

El análisis lineal asume que la relación entre cargas y desplazamientos resultantes es lineal, es decir, se cumple el principio de superposición: si se dobla la magnitud de la carga se dobla la respuesta del modelo (desplazamientos, deformaciones y tensiones resultantes). Todas las estructuras reales se comportan de forma no lineal a partir de un cierto nivel de carga. En muchos casos, un análisis lineal puede ser adecuado, pero en otros la solución lineal producirá resultados erróneos, en cuyo caso se deberá realizar un análisis no lineal.

Entre los modelos lineales podemos mencionar:

Lineal Elástico Isotrópico

Definidos por el módulo de Young y el coeficiente de Poisson, para pequeñas deformaciones cuando el material tiene un comportamiento elástico uniforme a lo largo de todos los ejes. Es usado en escenarios en los que el objeto de interés está diseñado para no llegar al rango no lineal de un material isótropo. Generalmente es aplicado en el estudio de test de caída o ensayos de pandeo de paneles de aluminio sin plasticidad.

Lineal Elástico Ortotrópico

Definido por el módulo de Young, coeficiente de Poisson y el módulo cortante, para pequeñas deformaciones, cuando el modelo no tiene propiedades elásticas uniformes en todos los ejes. Usado en escenarios en los que el objeto de interés está diseñado para evitar el rango no lineal de un objeto ortotrópico.

Termo-Elástico

Definido por el coeficiente de expansión térmica dependiente de la temperatura, módulo de Young y coeficiente de Poisson para pequeñas deformaciones, cuando las estructuras experimentan variaciones de temperatura que producen dilataciones y contracciones (y por tanto cargas adicionales).

2.3.2. Modelos No Lineales

Es el que tiene en cuenta la no linealidad mecánica, esto es, el comportamiento tenso-deformacional de los materiales y la no linealidad geométrica, es decir, la consideración del equilibrio de la estructura en su situación deformada.

En el análisis no lineal, una importante fuente de no linealidades se debe al efecto de los grandes desplazamientos en la configuración geométrica global de la estructura. Las estructuras y componentes mecánicos con grandes desplazamientos pueden experimentar importantes cambios en la geometría debido a que las cargas inducidas por la deformación pueden provocar una respuesta no lineal de la estructura en forma de rigidización (stress stiffening) o ablandamiento (stress softening).

Otra importante causa de no linealidad se debe a la relación lineal existente entre tensión y deformación. Esta situación ocurre cuando el material no sigue la Ley de Hooke, es decir, las tensiones no son directamente proporcionales a las deformaciones. Algunos materiales se comportan linealmente sólo sí las deformaciones son muy pequeñas, otros materiales en cambio siguen comportamientos totalmente diferentes.

Entre los modelos no lineales podemos citar:

Elástico por Puntos

Definido por curva por puntos de deformación unitaria-tensiones, para modelar miembros estructurales que siguen una curva determinada deformación tensión. Tanto la carga como la descarga son siguiendo la curva definida. Generalmente es usado en análisis de cables de catenaria para predecir su deformada y en el pandeo de estructuras elásticas reticuladas.

Von Misses con endurecimiento isotrópico

Definido en la parte elástica por el Módulo de Young y en la parte plástica por el limite elástico y módulo de endurecimiento por deformación, en materiales con pequeñas o grandes deformaciones, que tienen un endurecimiento isotrópico al pasar al limite elástico. Permite simular un material plástico por dos rectas, una para el tramo lineal y otra para el plano plástico (forma estándar bilineal). Se produce un incremento del límite elástico si se entra en la zona plástica, y la descarga se realiza de forma paralela a la recta elástica definida. En el formato multilineal se puede especificar la curva deformación-tensión mediante puntos.

Termo Plástico

Definido por el coeficiente de expansión térmica, Módulo de Young, coeficiente de Poisson, límite elástico uniaxial y endurecimiento por deformación en función de la temperatura. Usado cuando las tensiones pueden exceder el límite elástico, y el comportamiento mecánico es afectado por cambios de temperatura. Adecuado

para estructuras donde la carga de la temperatura está lejos del punto de fusión de los materiales, no se dan efectos o movimientos masivos de dislocaciones en el material.

Mooney-Rivlin

Definido por la curva de deformación-tensión a través de las constantes de Mooney-Rivlin o por una curva por puntos. Es utilizado en el análisis de pequeñas o grandes deformaciones de materiales hiperelásticos incompresibles. La carga y descarga siguen la curva definida por el usuario. No se produce deformación plástica. Un ejemplo de su aplicación es el análisis de apoyos de gomas de motores sujetos a vibración, sellos de goma para la industria de válvulas y escobillas de limpieza de parabrisas.

Visco-Elástico

Define su parte elástica a través del Modulo de Young y su parte plástica a través del módulo de endurecimiento por deformación, función general de la ley potencial de cedencia. Es aplicado cuando las tensiones están por debajo del límite elástico con un comportamiento de cedencia representado por una ley potencial.

2.4. Ecuación Diferencial y Condiciones de Frontera

Una ecuación diferencial es una ecuación que involucra derivadas de una función desconocida de una ó más variables. Sí la función desconocida depende sólo de una variable (de tal modo que las derivadas son ordinarias) la ecuación se llama una ecuación diferencial ordinaria. Sin embargo, si la función desconocida depende de más de una variable (de tal modo que las derivadas son derivadas parciales) la ecuación se llama una ecuación diferencial parcial. El orden de una ecuación diferencial es el orden de la derivada más alta que aparece en la ecuación. En su forma más general una ecuación diferencial se puede escribir como:

$$A(u) = f \quad \text{en } \Omega, \tag{2.17}$$

donde A es un operador diferencial, u y f son funciones suficientemente diferenciables para satisfacer las restricciones del operador A, y u se conoce como función solución o variable dependiente. El operador A se dice que es lineal en su argumento u, sí y sólo sí se cumple que

$$A(\alpha u + \beta u) = \alpha A(u) + \beta A(v) \tag{2.18}$$

para todo escalar α,β, y funciones u,v. Sí $f \equiv 0$ en (2.17), se dice que la ecuación es *homogénea*, en caso contrario, *no homogénea*. Además, Ω en (2.17) se conoce como el *dominio* de la ecuación diferencial, y para este contexto siempre se definirá real.

Cuando la variable dependiente es función de una variable, el dominio es un segmento de recta y su frontera son los dos puntos extremos. Cuando la variable

dependiente es función de dos variables, el dominio es, comúnmente, una región del plano y su frontera es la curva que encierra (o delimita) la región.

Para garantizar la unicidad de la solución de la ecuación diferencial, es necesario imponer algunas condiciones sobre la variable dependiente. Cuando estas condiciones se aplican a los valores de la función solución y/o a sus derivadas en los puntos de la frontera, se conoce como un *problema de valor de frontera*. Las condiciones de frontera se denotan como

$$B(u) = g \quad \text{en } \Gamma, \tag{2.19}$$

donde B es un operador diferencial, u la variable dependiente y g es una función definida en la frontera Γ. Dependiendo de como se formulan las condiciones de frontera se pueden tener diferentes tipos, entre ellas: condición de *Dirichlet*, condición de *Newmann*, condición de *Robin-Newmann* generalizada o mixta.

2.5. Formulación General de la Ecuación para el Comportamiento del Tejido Blando

El comportamiento del tejido blando puede ser modelado como un sistema masa-resorte, donde un objeto (un sólido ó volumen) puede ser visto como una colección de puntos de masa conectados por resortes de *fuerza lineal* a través de una estructura de malla, la cual puede ser representada como se muestra en la figura 2.3. Los resortes conectan los puntos de masa ejerciendo fuerza sobre los puntos vecinos cuando una masa es desplazada de su posición de equilibrio. En vista de esto, la ecuación del movimiento de una masa fija a un resorte amortiguado puede

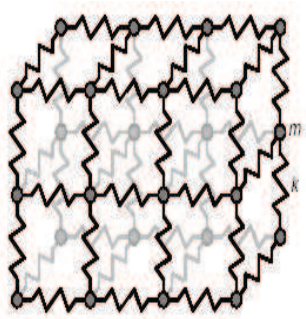


Figura 2.3: Estructura modelo masa-resorte.

ser expresada de la siguiente manera:

$$m_i \frac{d^2 P_i}{dt^2} = \gamma_i \frac{dP_i}{dt} + F_i \tag{2.20}$$

la cual no es más que la misma ecuación del movimiento armónico simple con una fuerza $\gamma_i \frac{dP_i}{dt}$ que tiende a extraer la energía de un sistema en movimiento, y con ello lo desacelera o lo amortigua, lo que es comúnmente llamado *Movimiento Armónico Amortiguado* [10]. Esta ecuación puede ser deducida de la segunda ley de Newton

$$\sum F_i = F_{ext} + F_{resist} = m_i.a_i \tag{2.21}$$

sí en la ecuación (2.21), sustituimos la aceleración por su equivalente

$$a = \frac{d^2 P_i}{dt^2} \tag{2.22}$$

y

$$F_{resist} = -\gamma_i.v = -\gamma_i \frac{dP_i}{dt} \tag{2.23}$$

obtenemos:

$$F_i = m_i \frac{d^2 P_i}{dt^2} - \gamma_i \frac{dP_i}{dt} \tag{2.24}$$

la cual es una ecuación diferencial de segundo orden lineal homogénea de coeficientes constantes, que expresa que la fuerza F ejercida sobre una partícula i es

igual a la masa m_i de esa partícula por la segunda derivada del desplazamiento P_i con respecto al tiempo dos veces, menos el coeficiente de amortiguamiento γ_i por la derivada de este desplazamiento con respecto al tiempo.

La ecuación del sistema completo es obtenida a través de (2.24) y viene dada por:

$$M\ddot{U} + C\dot{U} + F(U) = R \tag{2.25}$$

donde M y C son las matrices de masa y amortiguamiento respectivamente de tamaño $3.N$ x $3.N$ (N= número de puntos que conforman el objeto), $\dot{U}$ y $\ddot{U}$ son las derivadas de los vectores de posición con respecto al tiempo una y dos veces respectivamente, $F(U)$ es el vector de fuerzas de los resortes y R es el vector de las fuerzas externas.

Es importante distinguir que aunque se está haciendo la analogía de que la piel puede ser vista como un sistema masa-resorte y se están utilizando estas ecuaciones para definir su comportamiento, no se está usando un *modelo de simulación* masa-resorte para realizar las simulaciones. El modelo masa-resorte, a nivel general, abarca mucho más que el uso de las ecuaciones constitutivas de mecánica continua que describen el comportamiento de un sistema. Para más detalle sobre las diferencias entre los modelos de simulación consultar [12].

2.6. Aproximación de la derivada

El cálculo de la derivada de una función puede ser un proceso *difícil* a nivel computacional ya sea por lo complicado de la definición analítica de la función

o por que esta se conoce únicamente en un número discreto de puntos, es por ello que se recurre a diferentes métodos para tratar de aproximar la misma. Uno de los métodos ó técnicas más usadas es la aproximación polinomial, ya que es bien conocido que los *polinomios algebraicos*, o sea el conjunto de funciones de la forma:

$$P_n(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0, \tag{2.26}$$

donde n es un entero no negativo y $a_0, ..., a_n$ son constantes reales, aproximan de manera uniforme a las funciones continuas. Dada una función cualquiera, definida y continua en un intervalo cerrado y acotado, existe un polinomio que está tan cerca de la función como se desee (**Teorema de aproximación de Weirstrass**) [3]. Así que las integrales definidas y las derivadas de un polinomio son fáciles de aproximar ya que estas a su vez son también polinomios.

La definición de la derivada de una función $f(x)$ en el punto "x" esta dada en términos del límite:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{2.27}$$

de esta definición podemos decir que si h es pequeño entonces:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \tag{2.28}$$

esto nos da inmediatamente la primera fórmula numérica para aproximar la derivada

$$D_h f'(x) = \frac{f(x+h) - f(x)}{h} \tag{2.29}$$

que normalmente se conoce como *aproximación lineal* o *aproximación de la recta tangente* de f en a y que nos dice que $D_h f'(x)$ aproxima a $f'(x)$ con un error

proporcional a h, i.e. $O(h)$, siempre y cuando la convergencia aritmética sea exacta [28].

Sin embargo, existen otros métodos que ofrecen mejores resultados que el anterior, como por ejemplo: la interpolación a través de polinomios de Lagrange, de Taylor, de Hermite, diferencias finitas, etc. Para un tratamiento más completo de estos métodos consultar [3].

2.6.1. Método de las Diferencias Finitas

La diferenciación numérica, o aproximación por diferencias, se utiliza para evaluar las derivadas de una función por medio de sus valores dados en los puntos de una retícula. Entre los métodos más utilizados se pueden mencionar: *método de las diferencias finitas hacia adelante*, *hacia atrás* y *centrada*.

Aproximación de la primera y segunda derivada de una función usando diferencias centradas

Sí se desea evaluar la primera derivada de $f'(x)$ y $f''(x)$ en $x = x_0$, se deben conocer los valores de f en $x_0 - h$, x_0 y $x_0 + h$, donde h es el tamaño del intervalo entre dos puntos consecutivos sobre el eje x (o cualquier otro eje que se esté estudiando). De esta manera podemos aproximar la primera y segunda derivada de f por diferencias *centradas*. A saber,

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} + O(h^2) \tag{2.30a}$$

$$f''(x_0) = \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} + O(h^2) \tag{2.30b}$$

2.7. Discretización del Tiempo para la Ecuación del Tejido Blando

Como vimos en la sección (2.5) obtuvimos una ecuación diferencial de segundo orden para la descripción del comportamiento del sistema a través del tiempo, sin embargo, a nivel computacional el cálculo de las derivadas que implica la ecuación (2.24) no se lleva a cabo de manera analítica, sino más bien por aproximaciones de las mismas. En vista de esto, se procederá a aproximar la primera y segunda derivada de la ecuación (2.24) con respecto al tiempo por el método de las diferencias finitas centradas presentado anteriormente.

Aplicando las ecuación (2.30a) a la primera derivada de (2.24) y tomando en cuenta que derivaremos con respecto al tiempo, es decir, que nuestro h será igual a $\triangle t$, nos queda:

$$\frac{dP_i}{dt} = \frac{P_i^{t+1} - P_i^{t-1}}{2\triangle t} \tag{2.31}$$

después, utilizando la ecuación (2.30b) para la segunda derivada de (2.24) obtenemos,

$$\frac{d^2P_i}{dt^2} = \frac{P_i^{t+1} - 2P_i^{t} + P_i^{t-1}}{\triangle t^2} \tag{2.32}$$

sustituyendo (2.31) y (2.32) en (2.24)

$$m_i(\frac{P_i^{t+1} - P_i^{t-1}}{2\triangle t}) = \gamma_i(\frac{P_i^{t+1} - 2P_i^{t} + P_i^{t-1}}{\triangle t^2}) + F_i \tag{2.33}$$

resolviendo finalmente obtenemos,

$$(\frac{m_i}{\triangle t^2} - \frac{\gamma_i}{2\triangle t})P_i^{t+1} = F_i + \frac{2m_i}{\triangle t^2}P_i^{t} - (\frac{m_i}{\triangle t^2} + \frac{\gamma_i}{2\triangle t})P_i^{t-1} \tag{2.34}$$

la cual será la ecuación empleada para calcular la posición próxima (x,y,z) en un tiempo $(t+1)$ de un punto P_i^{t+1} a partir de una fuerza F_i, y su posición actual P_i^t y anterior P_i^{t-1}.

2.8. Ley de Hooke

La pendiente de la recta es la relación entre el esfuerzo y la deformación; se llama módulo de elasticidad y se representa por la letra E, que se suele escribir:

$$\sigma = E.\epsilon \tag{2.35}$$

que no expresa otra cosa que la conocida Ley de Hooke. En principio, Hooke sólo anunció la ley de que el esfuerzo es proporcional a la deformación. Fué Thomas Young, en el año 1807, quién introdujo la expresión matemática con una constante de proporcionalidad que se llamó módulo de Young.

De la ley de Hooke, podemos ver que las unidades para el módulo de elasticidad E son idénticas a las unidades para el esfuerzo σ.

Otra forma de la expresión de la ley de Hooke, muy conveniente a veces, es la llamada *Ley de Hooke Generalizada*. Con la introducción de los conceptos de deformación y esfuerzo, Cauchy generalizó esta ley en una ecuación tensorial de esfuerzo que puede ser escrita de la siguiente manera:

$$\sigma^{ij} = E^{ijkl}\epsilon^{kl} \tag{2.36}$$

Recalquemos que la validez de la expresión (2.35) y (2.36) implica tener en cuenta las hipótesis siguientes:

1. La carga ha de ser axial.

2. El cuerpo es continuo y permanece continuo bajo la acción de fuerzas externas.

3. El esfuerzo no debe sobrepasar el límite de proporcionalidad. Existe un único estado de equilibrio al cual el cuerpo retorna cuando todas las fuerzas externas son removidas.

2.9. Formulación de la Ecuación de Energía Lineal Elástica

La energía lineal elástica W_{lineal}, para un material isotrópico homogéneo [5], está definida por la siguiente fórmula:

$$W_{lineal} = \frac{\lambda}{2}(trE_l)^2 + \mu(trE_l^2) \tag{2.37}$$

donde λ(módulo de Poisson) y μ(módulo de rígidez) son conocidas como las constantes de Lamé y

$$E_l = \frac{1}{2}(\nabla U + \nabla U^t) \tag{2.38}$$

como el tensor de esfuerzo de *Green-St.Venant* [11] de dimensión 3x3 con sus invariantes principales

$$l_1 = trE_l \tag{2.39a}$$

$$l_2 = trE_l^2 \tag{2.39b}$$

donde

$$U = \begin{bmatrix} U_x \\ U_y \\ U_z \end{bmatrix} \tag{2.40}$$

$$U^t = \begin{bmatrix} U_x U_y U_z \end{bmatrix} \tag{2.41}$$

y ∇ es el gradiente de un vector, definido de la siguiente manera:

$$\nabla U = \begin{bmatrix} \frac{\partial U_x}{\partial x} & \frac{\partial U_x}{\partial y} & \frac{\partial U_x}{\partial z} \\ \frac{\partial U_y}{\partial x} & \frac{\partial U_y}{\partial y} & \frac{\partial U_y}{\partial z} \\ \frac{\partial U_z}{\partial x} & \frac{\partial U_z}{\partial y} & \frac{\partial U_z}{\partial z} \end{bmatrix} \tag{2.42}$$

y,

$$\nabla U^t = \begin{bmatrix} \frac{\partial U_x}{\partial x} & \frac{\partial U_y}{\partial x} & \frac{\partial U_z}{\partial x} \\ \frac{\partial U_x}{\partial y} & \frac{\partial U_y}{\partial y} & \frac{\partial U_z}{\partial y} \\ \frac{\partial U_x}{\partial z} & \frac{\partial U_y}{\partial z} & \frac{\partial U_z}{\partial z} \end{bmatrix} \tag{2.43}$$

Ahora bien, para fines de la investigación, la ecuación (2.37) será modificada de manera conveniente y equivalente para facilitar la obtención de la ecuación de fuerza en las siguientes secciones, veamos:

Sustituyendo en la ecuación (2.38), las ecuaciones (2.42) y (2.43) obtenemos,

$$E_l = \begin{bmatrix} \frac{\partial U_x}{\partial x} & \frac{1}{2}(\frac{\partial U_x}{\partial y} + \frac{\partial U_y}{\partial x}) & \frac{1}{2}(\frac{\partial U_x}{\partial z} + \frac{\partial U_z}{\partial x}) \\ \frac{1}{2}(\frac{\partial U_x}{\partial y} + \frac{\partial U_y}{\partial x}) & \frac{\partial U_y}{\partial y} & \frac{1}{2}(\frac{\partial U_y}{\partial z} + \frac{\partial U_z}{\partial y}) \\ \frac{1}{2}(\frac{\partial U_x}{\partial z} + \frac{\partial U_z}{\partial x}) & \frac{1}{2}(\frac{\partial U_y}{\partial z} + \frac{\partial U_z}{\partial y}) & \frac{\partial U_z}{\partial z} \end{bmatrix} \tag{2.44}$$

cuya *traza* sería,

$$tr(E_l) = \frac{\partial U_x}{\partial x} + \frac{\partial U_y}{\partial y} + \frac{\partial U_z}{\partial z} = div(U) \tag{2.45}$$

donde $div(U)$ es la divergencia de U.

Luego tenemos que,

$$\begin{aligned} tr(E_l^2) = &\left(\frac{\partial U_x}{\partial x}\right)^2 + \left(\frac{\partial U_y}{\partial y}\right)^2 + \left(\frac{\partial U_z}{\partial z}\right)^2 + \frac{1}{2}\left(\frac{\partial U_x}{\partial y} + \frac{\partial U_y}{\partial x}\right)^2 \\ &+ \frac{1}{2}\left(\frac{\partial U_x}{\partial z} + \frac{\partial U_z}{\partial x}\right)^2 + \frac{1}{2}\left(\frac{\partial U_y}{\partial z} + \frac{\partial U_z}{\partial y}\right)^2 \end{aligned} \tag{2.46}$$

a la cual le aplicamos la siguiente propiedad de producto notable,

$$a^2 + b^2 - \frac{1}{2}(a-b)^2 = \frac{1}{2}(a+b)^2 \tag{2.47}$$

para obtener,

$$\begin{aligned} tr(E_l^2) = &\left(\frac{\partial U_x}{\partial x}\right)^2 + \left(\frac{\partial U_x}{\partial y}\right)^2 + \left(\frac{\partial U_x}{\partial z}\right)^2 + \left(\frac{\partial U_y}{\partial x}\right)^2 \\ &+ \left(\frac{\partial U_y}{\partial y}\right)^2 + \left(\frac{\partial U_y}{\partial z}\right)^2 + \left(\frac{\partial U_z}{\partial x}\right)^2 + \left(\frac{\partial U_z}{\partial y}\right)^2 + \left(\frac{\partial U_z}{\partial z}\right)^2 \\ &- \frac{1}{2}\left[\left(\frac{\partial U_x}{\partial y} - \frac{\partial U_y}{\partial x}\right)^2 + \left(\frac{\partial U_x}{\partial z} - \frac{\partial U_z}{\partial x}\right)^2 + \left(\frac{\partial U_y}{\partial z} - \frac{\partial U_z}{\partial y}\right)^2\right] \end{aligned} \tag{2.48}$$

ahora, se hará un breve paréntesis para realizar las siguientes definiciones:

Norma Frobenius:

Sea A una matriz de dimensiones NxN, entonces se define la norma Frobenius de la siguiente manera:

$$\|A\|_F^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}^2 \tag{2.49}$$

Rotor de un Vector:

Sea U un vector de dimensión tres, definido como:

$$U = \begin{bmatrix} U_x \\ U_y \\ U_z \end{bmatrix} \tag{2.50}$$

el rotor de U esta definido de la siguiente manera:

$$rot(U) = \begin{bmatrix} \frac{\partial U_z}{\partial y} - \frac{\partial U_y}{\partial z} \\ \frac{\partial U_z}{\partial x} - \frac{\partial U_x}{\partial z} \\ \frac{\partial U_y}{\partial x} - \frac{\partial U_x}{\partial y} \end{bmatrix} \tag{2.51}$$

Norma 2:

Sea X un vector de dimensión N, la norma 2 se define como:

$$\|X\|_2^2 = \sum_{i=1}^{N} x_i^2 \tag{2.52}$$

donde x_i con $i = 1...N$ son los elementos del vector.

Volviendo a la deducción se puede demostrar que la norma Frobenius del gradiente de U es igual a:

$$\begin{aligned} \|\nabla U\|_F^2 = &\left(\frac{\partial U_x}{\partial x}\right)^2 + \left(\frac{\partial U_x}{\partial y}\right)^2 + \left(\frac{\partial U_x}{\partial z}\right)^2 + \left(\frac{\partial U_y}{\partial x}\right)^2 + \left(\frac{\partial U_y}{\partial y}\right)^2 + \left(\frac{\partial U_y}{\partial z}\right)^2 \\ &+ \left(\frac{\partial U_z}{\partial x}\right)^2 + \left(\frac{\partial U_z}{\partial y}\right)^2 + \left(\frac{\partial U_z}{\partial z}\right)^2 \end{aligned} \tag{2.53}$$

por (2.49) y utilizando (2.51) y (2.52) se puede deducir que

$$\|rot(U)\|_2^2 = \left[\left(\frac{\partial U_x}{\partial y} - \frac{\partial U_y}{\partial x}\right)^2 + \left(\frac{\partial U_x}{\partial z} - \frac{\partial U_z}{\partial x}\right)^2 + \left(\frac{\partial U_y}{\partial z} - \frac{\partial U_z}{\partial y}\right)^2\right] \tag{2.54}$$

entonces por las ecuaciones (2.37), (2.48), (2.53) y (2.54) se tiene que:

$$\mu tr E_l^2 = \mu \|\nabla U\|_F^2 - \frac{\mu}{2}\|rot(U)\|_2^2 \tag{2.55}$$

logrando finalmente encontrar una expresión equivalente para (2.37) dada por:

$$W_{lineal} = \frac{\lambda}{2}(tr E_l)^2 + \mu(tr E_l^2) = \frac{\lambda}{2}(divU)^2 + \mu\|\nabla U\|_F^2 - \frac{\mu}{2}\|rot(U)\|_2^2 \tag{2.56}$$

resultado que se puede verificar sustituyendo las ecuaciones (2.45) y (2.55) en (2.37).

2.10. El Método de los Elementos Finitos (MEF)

El método de los elementos finitos es una de las técnicas más generales para la solución numérica de ecuaciones diferenciales.

Este método no opera directamente sobre las ecuaciones diferenciales; en lugar de esto, aproxima a las funciones colocando la frontera continua y los problemas de valor inicial dentro de formas de aproximación variacional o de peso residual [6] y [26], equivalentes, como por ejemplo Rayleigh-Ritz, Galerkin, Mínimos cuadrados, Colocación, Petrov-Galerkin, etc. La integral de una función sobre un dominio arbitrario puede ser particionada en sumas de integrales sobre una colección arbitraria de subdominios llamados elementos finitos. Tan pronto como los subdominios son suficientemente pequeños, las funciones polinómicas pueden adecuadamente representar el comportamiento local de la solución. En cada elemento las funciones de aproximación son, frecuentemente, polinomios algebraicos,

y los parámetros incógnitas representan el valor de la función solución en un número preseleccionado de puntos, llamados *nodos*, sobre el borde y el interior del elemento. Las funciones de forma son seleccionadas usando conceptos de la teoría de la interpolación, por lo que también son llamadas funciones de interpolación. El grado de las funciones de interpolación depende del número de nodos definidos en el elemento y el orden de la ecuación diferencial a ser resuelta.

Para el MEF es necesario definir la función de aproximación como

$$u_N = \sum_{j=1}^{N} U_j \phi_j \tag{2.57}$$

donde N representa el número de nodos en la malla, U_j es lo que se conoce como el j-ésimo valor nodal, que corresponde al valor de la función solución en el nodo j, y ϕ_j es la j-ésima función de interpolación, la cual tiene la propiedad, que para cualquier nodo de la malla p_i

$$\phi_j(p_i) = \delta_{ij} = \begin{cases} 1, & \text{si } i = j \\ 0, & \text{si } i \neq j \end{cases} \tag{2.58}$$

A nivel general, de acuerdo a las observaciones hechas por Zienkiewicz (1975) y Oden (1991), las características más importantes de elementos finitos pueden ser resumidas como:

1. *Geometrías Arbitrarias*: es independiente de la geometría. Puede ser aplicado a dominios de forma compleja y con condiciones de frontera completamente arbitrarias. Un dominio geométricamente complejo se representa

como una colección de subdominios geométricamente más simples, llamados elemento finito. Sobre cada elemento finito, las funciones de aproximación se escogen utilizando la idea básica de que cualquier función continua puede ser representada como una combinación lineal de polinomios algebraicos.

2. *Mallado No Estructurado*:

   2.1 No necesita el análisis de transformación de coordenadas globales.

   2.2 Puede ser ubicado en cualquier dominio físico.

   2.3 Se pueden borrar o sumar elementos sin cambiar la estructura global de datos.

3. *Programas de propósitos generales y flexibles*: la clara estructura y versatilidad del método de los elementos finitos hace posible construir softwares para aplicaciones de propósito general.

4. *Fundamentación Matemática*: esta fundamentado en más de seis décadas de trabajo extenso sobre teoría matemática. Esto le ha proporcionado amplia confianza y en muchos casos hace posible el análisis matemático y la estimación de la exactitud de los elementos finitos.

El método del elemento finito puede ser visto, como una aplicación de los métodos de residuos ponderados o variacionales sobre cada elemento. Esto lo podemos apreciar si observamos que las incógnitas (valores nodales) se obtienen satisfaciendo la ecuación de gobierno en un sentido integral pesado, sobre cada elemento. Por ello, es importante resaltar que el método del elemento finito es el procedimiento para dividir el dominio, definir las funciones de forma en cada subdominio

y aplicar las condiciones necesarias de ensamblaje para satisfacer la ecuación de gobierno en un sentido variacional o MRP.

2.10.1. El Mallado en el MEF

La representación de una región dada por un conjunto de elementos es lo que se conoce como *discretización*, *generación de la malla* o *mallado*, y es un paso importante en el análisis de elemento finito. La elección del tipo de elemento, el número de elementos y su densidad, dependen de la geometría del dominio y el grado de precisión deseado. Al hacer una discretización del dominio se incurre en lo que se conoce como *error de discretización* del dominio, el cual es importante tener en cuenta. Las funciones de interpolación dependen, no solo del número de nodos en el elemento, sino también de su forma. Esta última debe ser tal, que su geometría esté únicamente definida por un conjunto de puntos, los cuales sirven como *nodos del elemento* o *nodos elementales* para la construcción de las funciones de interpolación.

Elemento Tetraédrico de Cuatro Nodos

A nivel general, todo cuerpo volumétrico puede ser dividido en tetraedros. El tetraedro de cuatro nodos, conocido también como tetraedro lineal, es el elemento tridimensional más sencillo, tal como el triángulo de tres nodos lo es entre los elementos bidimensionales. Estos elementos polinómicos según su orden generan polinomios de aproximación completos. La aproximación se escribe dentro de cada

tetraedro por el polinomio lineal siguiente:

$$\phi_i^{(e)} = \alpha_0 + \alpha_1 x_i^e + \alpha_2 y_i^e + \alpha_3 z_i^e \tag{2.59}$$

donde ϕ^e es el valor nodal del j-ésimo elemento de e de la función de aproximación. De la expresión anterior se deduce

$$\phi_1^e = \alpha_0 + \alpha_1 x_1^e + \alpha_2 y_1^e + \alpha_3 z_1^e \tag{2.60a}$$

$$\phi_2^e = \alpha_0 + \alpha_1 x_2^e + \alpha_2 y_2^e + \alpha_3 z_2^e \tag{2.60b}$$

$$\phi_3^e = \alpha_0 + \alpha_1 x_3^e + \alpha_2 y_3^e + \alpha_3 z_3^e \tag{2.60c}$$

$$\phi_4^e = \alpha_0 + \alpha_1 x_4^e + \alpha_2 y_4^e + \alpha_3 z_4^e \tag{2.60d}$$

Resolviendo el sistema anterior y sustituyendo los α_i en (2.59) se obtiene, tras agrupar términos

$$\phi = \sum_{i=1}^{4} N_i^e(x,y,z)\phi_i^e = N_1^e(x,y,z)\phi_1^e + N_2^e(x,y,z)\phi_2^e + ... + N_4^e(x,y,z)\phi_4^e \tag{2.61}$$

siendo

$$N_i^e(x,y,z) = \frac{1}{6V^e}(a_i^e + b_i^e x + c_i^e y + d_i^e z) \tag{2.62}$$

las funciones de forma del elemento tridimensional, con

$$a_i^e = det \begin{vmatrix} x_j^e & y_j^e & z_j^e \\ x_k^e & y_k^e & z_k^e \\ x_l^e & y_l^e & z_l^e \end{vmatrix}$$

$$b_i^e = -det \begin{vmatrix} 1 & y_j^e & z_j^e \\ 1 & y_k^e & z_k^e \\ 1 & y_l^e & z_l^e \end{vmatrix}$$

$$c_i^e = det \begin{vmatrix} x_j^e & 1 & z_j^e \\ x_k^e & 1 & z_k^e \\ x_l^e & 1 & z_l^e \end{vmatrix}$$

$$d_i^e = -det \begin{vmatrix} x_j^e & y_j^e & 1 \\ x_k^e & y_k^e & 1 \\ x_l^e & y_l^e & 1 \end{vmatrix}$$

el resto de las constantes se obtiene rotando cíclicamente los subíndices (i,j,k,l) aplicando la regla de la mano derecha.

El volumen del elemento V^e se halla a partir del determinante

$$6V^e = det \begin{vmatrix} 1 & x_i^e & y_i^e & z_i^e \\ 1 & x_j^e & y_j^e & z_j^e \\ 1 & x_k^e & y_k^e & z_k^e \\ 1 & x_l^e & y_l^e & z_l^e \end{vmatrix}$$

Es importante destacar que los nodos de un elemento finito poseen una numeración global y una interna, que es utilizada para el ensamblaje de las ecuaciones.

La *numeración interna* consiste en que todos los nodos de un elemento son numerados de la misma forma. Esto lo podemos apreciar en la figura 2.4.

Esto significa que los 4 vértices de cualquier tetraedro se numeran con la secuencia 1,2,3,4 en dirección contraria a las agujas del reloj, de manera de identificar las coordenadas de área correspondientes.

La *numeración global* consiste en numerar todos los nodos de todos los elementos de una manera global. Por ejemplo, sí un cuerpo sólido consta de n elementos

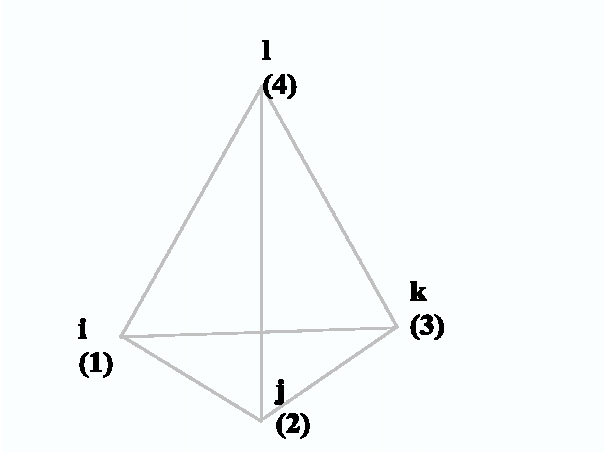


Figura 2.4: Numeración Interna de un Elemento Tetraédrico de 4 nodos

finitos con t nodos, el número total de nodos que tendrá este sólido será de $n.t$ y su numeración global irá de 1 a $n.t$. Un ejemplo de numeración global para dos elementos hexaédricos puede ser visto en la figura 2.5.

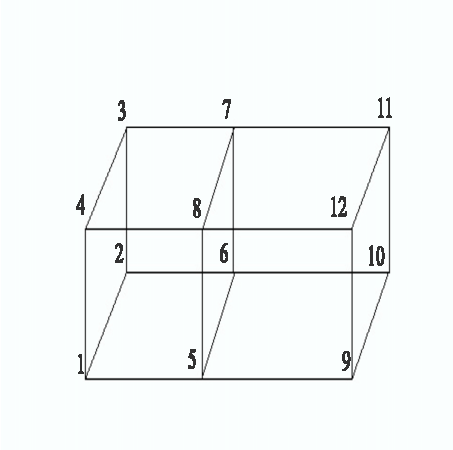


Figura 2.5: Ejemplo de Numeración Global de dos elementos hexaédricos de 8 nodos

2.11. Formulación de la Fuerza a través del MEF

En la sección (2.7) se obtuvo la ecuación (2.34) para modelar el comportamiento del cuerpo a través del tiempo, en la cual aparece la cantidad física F_i que representa la fuerza. En este apartado se presenta una formulación para dicha cantidad, fundamentada en el concepto de elemento finito a través de la ecuación de energía lineal (2.56) W_{lineal}, ya encontrada en la sección (2.9).

El procedimiento consistirá en llevar la ecuación de W_{lineal} a la forma de elemento finito, para luego derivarla en función del desplazamiento y obtener la fuerza.

Se adoptará la estructura y propiedades del elemento tetraédrico de cuatro nodos para realizar la discretización del dominio tridimensional, utilizando las funciones de forma presentadas en la sección (2.10.1). Se aproximarán los desplazamientos U a través de la siguiente ecuación:

$$U(x, y, z) = \sum_{j=0}^{3} U_j \Lambda_j(x, y, z) \tag{2.63}$$

donde Λ_j será la j-ésima función de forma, definida como

$$\Lambda_j = a_j + b_j x + c_j y + d_j z \tag{2.64}$$

la cual es equivalente a decir

$$\Lambda_j = \begin{bmatrix} b_j & c_j & d_j \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + a_j = \alpha_j^t X + \beta_j \tag{2.65}$$

de la que se obtiene

$$\alpha_j = \frac{(-1)^j}{6V(T_i)}(P_{j+1} \times P_{j+2} + P_{j+2} \times P_{j+3} + P_{j+3} \times P_{j+1}) \tag{2.66}$$

siendo P_{j+i} un vector de posición (con i=0,1,2,3), $V(T_i)$ el volumen del tetraedro i y $\times$ el producto cruz entre dos vectores.

Ahora bien, definidas ya las funciones de aproximación y de forma, se trabajará sobre la ecuación de energía lineal elástica

$$W_{lineal} = \frac{\lambda}{2}(divU)^2 + \mu\|\nabla U\|^2 - \frac{\mu}{2}\|rot(U)\|_2^2 \tag{2.67}$$

sustituyendo (2.63) en U y derivando la ecuación con respecto a esta. Empezaremos paso a paso con cada uno de los elementos de la ecuación. Veamos:

Sustituyendo (2.63) en (2.45) y derivando, se obtiene

$$\begin{aligned} div(U) =& \frac{\partial\left(\sum_{j=0}^{3} U_{jx}\Lambda_j(x,y,z)\right)}{\partial x} + \frac{\partial\left(\sum_{j=0}^{3} U_{jy}\Lambda_j(x,y,z)\right)}{\partial y} \\ &+ \frac{\partial\left(\sum_{j=0}^{3} U_{jz}\Lambda_j(x,y,z)\right)}{\partial z} \end{aligned} \tag{2.68}$$

y, por propiedades de sumatoria,

$$\begin{aligned} div(U) =& \sum_{j=0}^{3} \frac{U_{jx}\partial\Lambda_j(x,y,z)}{\partial x} + \frac{U_{jy}\partial\Lambda_j(x,y,z)}{\partial y} \\ &+ \frac{U_{jz}\partial\Lambda_j(x,y,z)}{\partial z} \end{aligned} \tag{2.69}$$

si ahora derivamos Λ_j, resulta

$$div(U) = \sum_{j=0}^{3} (U_{jx}b_j + U_{jy}c_j + U_{jz}d_j) \tag{2.70}$$

la cual elevando al cuadrado

$$(div(U))^2 = \sum_{j=0}^{3} (U_{jx}b_j + U_{jy}c_j + U_{jz}d_j) \sum_{j=0}^{3} (U_{jx}b_j + U_{jy}c_j + U_{jz}d_j) \tag{2.71}$$

y efectuando operaciones es igual a

$$(div(U))^2 = \sum_{j=0}^{3}\sum_{k=0}^{3} (U_{jx}b_j + U_{jy}c_j + U_{jz}d_j)(U_{jx}b_j + U_{jy}c_j + U_{jz}d_j) \tag{2.72}$$

$$(div(U))^2 = \sum_{j,k=0}^{3} (U_{jx}b_j + U_{jy}c_j + U_{jz}d_j)(U_{jx}b_j + U_{jy}c_j + U_{jz}d_j) \tag{2.73}$$

la cual se puede expresar como

$$(div(U))^2 = \sum_{j,k=0}^{3} \left(\begin{bmatrix} U_{jx} & U_{jy} & U_{jz} \end{bmatrix} \begin{bmatrix} b_j \\ c_j \\ d_j \end{bmatrix} \begin{bmatrix} b_k & c_k & d_k \end{bmatrix} \begin{bmatrix} U_{kx} \\ U_{ky} \\ U_{kz} \end{bmatrix} \right) \tag{2.74}$$

que es igual a

$$(div(U))^2 = \sum_{j,k=0}^{3} (U_j^t \alpha_j \alpha_k^t U_k) \tag{2.75}$$

obteniendo finalmente,

$$(div(U))^2 = \sum_{j,k=0}^{3} (U_j^t \alpha_j \otimes \alpha_k^t U_k) \tag{2.76}$$

por definición de $u \otimes v = u.v^t$ (producto tensorial).

Ahora, pasaremos a analizar el siguiente miembro de la ecuación (2.56), es decir, $\|\nabla U\|^2$. En esta ecuación sustituiremos (2.63) y lo derivaremos, para obtener:

$$\begin{aligned} \|\nabla U\|^2 = \sum_{j,k=0}^{3} [&(U_{jx}b_jb_kU_{kx}) + (U_{jx}c_jc_kU_{kx}) + (U_{jx}d_jd_kU_{kx}) \\ &+ (U_{jy}b_jb_kU_{ky}) + (U_{jy}c_jc_kU_{ky}) + (U_{jy}d_jd_kU_{ky}) \\ &+ (U_{jz}b_jb_kU_{kz}) + (U_{jz}c_jc_kU_{kz}) + (U_{jz}d_jd_kU_{kz}) \end{aligned} \tag{2.77}$$

Quedando sólo por analizar el último miembro de (2.56), es decir, $\|rot(U)\|^2$, el cual esta definido por (2.49), donde sustituimos (2.63) y derivamos, resultando

$$\begin{aligned}\|rot(U)\|^2 = \sum_{j,k=0}^{3} (&U_{jz}c_jc_kU_{kz} - U_{jz}c_jd_kU_{ky} - U_{jy}d_jc_kU_{kz} + U_{jy}d_jd_kU_{ky} \\ &+ U_{jz}b_jb_kU_{kz} - U_{jz}b_jd_kU_{kx} - U_{jx}d_jb_kU_{kz} + U_{jx}d_jd_kU_{kx} \\ &+ U_{jy}b_jb_kU_{ky} - U_{jy}b_jc_kU_{kx} - U_{jx}c_jb_kU_{ky} + U_{jx}c_jc_kU_{kx}\end{aligned} \tag{2.78}$$

Finalmente, después de haber encontrado formas equivalentes para los 3 miembros de la ecuación de la energía lineal elástica W_{lineal}, sustituimos (2.76), (2.77) y (2.78) en (2.56) y se obtiene:

$$W_{lineal}(T_i) = \sum_{j,k=0}^{3} U_j^t[B_{jk}^{T_i}]U_k \tag{2.79}$$

donde

$$B_{jk}^{Ti} = \frac{\lambda}{2}(\alpha_j \otimes \alpha_k) + \frac{\mu}{2}[(\alpha_j \otimes \alpha_k) + (\alpha_j.\alpha_k)Id_3] \tag{2.80}$$

es una matriz de dimensiones 3×3. Siendo Id_3 una matriz identidad de 3×3 y $u \otimes v = u.v^t$.

Luego de esto, utilizando el *primer teorema de Castigliano* que establece que la derivada parcial de la energía W en función del desplazamiento U es igual a la fuerza F, es decir,

$$F = \frac{\partial W}{\partial U} \tag{2.81}$$

derivamos (2.79) en función de U y obtenemos:

$$F_p^{T_i} = 2\sum_{j=0}^{3}[B_{pj}^{T_i}]U_j \tag{2.82}$$

ecuación que será utilizada para calcular la fuerza en (2.24). Para más detalles sobre la demostración de esta ecuación consultar el apéndice B de este trabajo.

2.12. Métodos Iterativos para Resolver Sistemas de Ecuaciones Lineales

La resolución de un sistema lineal $Ax = b$ a través de una factorización como: LU, LDL^T, Choleski, etc., suelen consumir muchos recursos del computador sí el sistema es de grandes dimensiones. Por ello, se han desarrollado otros métodos como los métodos iterativos que suelen presentar mejores resultados en tiempo de cómputo, sobre todo sí la matriz es esparcida.

Un método iterativo con el cual se resuelve el sistema lineal $Ax = b$ comienza con una aproximación inicial $x^{(0)}$ a la solución x y genera una sucesión de vectores $\{x_k\}_{k=0}^{\infty}$ que converge en x. Los métodos iterativos conllevan un proceso que convierte el sistema $Ax = b$ en otro equivalente de la forma $x = Tx + c$ para alguna matriz fija T y un vector c. Luego de seleccionar el vector inicial $x^{(0)}$ la sucesión de los vectores de la solución aproximada se genera calculando

$$x^{(k)} = Tx^{(k-1)} + c \qquad \text{para cada } k = 1, 2, 3, ... \tag{2.83}$$

El tiempo necesario para conseguir una exactitud satisfactoria a través de un método iterativo es rebasado cuando los sistemas son de dimensión pequeña. Sin embargo, en sistemas grandes con un alto porcentaje de elementos cero, son eficientes tanto en almacenamiento como en el tiempo de cómputo [3].

2.12.1. Método Iterativo de Gauss-Seidel

Para resolver un sistema iterativo $Ax = b$ dada una aproximación inicial $x^{(0)}$, sea $x^{(k)}$ la incognita que deseamos calcular, utilizamos las componentes de $x^{(k-1)}$. Como para $x^{(k-1)}$, ya se calcularon $x^{(k-1)}...x^{(1)}$, se calcula $x^{(k)}$ por medio de los valores calculados más recientemente, o sea,

$$x_i^{(k)} = \frac{-\sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)} + b_i}{a_{ii}} \tag{2.84}$$

donde los a_{ij} son los elementos de la matriz A, los b_i son los elementos del vector b y x_i^k es la incognita i en la iteración k, para $1 \leq i, j \leq n$.

Diversos criterios de parada pueden ser utilizados para detener el algoritmo de Gauss-Seidel, en este trabajo se usará el siguiente:

$$\frac{\|x^k - x^{k-1}\|}{\|x^k\|} \tag{2.85}$$

para alguna tolerancia prescrita $\varepsilon > 0$. Donde será empleada la norma infinito l_∞.

Capítulo 3

DESARROLLO

En el capítulo anterior se presentó la base teórica empleada para la comprensión y resolución del problema planteado y se formuló la ecuación de mecánica continua que describe el comportamiento de la deformación de un objeto a través del tiempo, sobre la cual se discretizó el tiempo a través del método de las diferencias finitas y se aproximó la fuerza usando el método de los elementos finitos. En este capítulo se llevaran las ecuaciones anteriormente estudiadas a expresiones computacionalmente manejables para luego presentar la implementación de las diferentes funciones que permiten realizar las simulaciones y deformaciones sobre el tejido blando. También se tratará un poco sobre la metodología de desarrollo del software implementada y de la API *Virtual Vision Machine* utilizada como soporte en el desarrollo de las subrutinas.

3.1. Resultados de La Metodología Utilizada

Para llevar a cabo los objetivos trazados en este trabajo, se realizaron a nivel general las etapas que involucran el modelo del ciclo de vida clásico del software, según [24], por tratarse de un módulo de software que persigue especificaciones bien definidas y que se adapta a las diferentes estructuras y clases de la API VVM . Este modelo divide el ciclo de vida del producto de programación en una serie de actividades sucesivas; cada fase requiere información de entrada, procesos y resultados, bien definidos. En este trabajo, el modelo de fases considera las siguientes partes:

- *Análisis*: en esta fase se realizó una revisión bibliográfica sobre varios trabajos realizados en el área de simulación de tejido blando, así como también todos los conceptos referentes a la teoría de elasticidad, el método de los elementos finitos, resistencia y mecánica de materiales. Logrando de esta manera plantear los objetivos de esta investigación, del software a realizar y la base teórica para la solución del problema, que pueden ser vistos en los capítulos I y II.

- *Diseño*: se formuló la ecuación general que modela el comportamiento de un objeto a través del tiempo, haciendo uso del método de las diferencias finitas para discretizar el tiempo y del método de los elementos finitos para aproximar la fuerza, y se realizaron los algoritmos en alto nivel para la simulación, los cuales se pueden apreciar en los capítulos II y III.

- *Codificación*: el diseño fué traducido en forma legible para la máquina. Esta

fase la podemos observar en el apéndice al final de este trabajo.

- *Prueba*: una vez generado el código, se hicieron las pruebas de las diferentes subrutinas. Inicialmente se validó que los resultados obtenidos con respecto a las deformaciones fueran similares a la realidad y además se probó la eficiencia del módulo de software con respecto a otros algoritmos similares. Estos resultados pueden ser observados en el capítulo IV.

3.2. La API *Virtual Vision Machine*(VVM)

Una Interfaz para la Programación de Aplicaciones ó "*Aplication Program Interface*" es un conjunto de subrutinas sobre las cuales se apoya un programador para realizar nuevas aplicaciones. A nivel general existen muchas API's, cada una orientada a un área en especial. En el caso de la computación gráfica existen muchas API's pero entre las más conocidas tenemos: Direct3D de Microsoft y OpenGl de Silicon Graphics, sobre las cuales están basadas la mayoría de los juegos, aplicaciones y tarjetas gráficas en el mercado.

El Virtual Vision Machine (VVM) es una API que ha sido diseñada para el desarrollo rápido de prototipos en campos que utilizan la Visualización Científica como la medicina 3D. Fue construida sobre OpenGl e implementada usando la "Programación Orientada a Objeto" bajo el lenguaje de programación C++.

VVM es un conjunto de entidades que se interconectan para construir mundos virtuales, con procesos estáticos o dinámicos. Se entiende como "Mundo virtual" una aplicación de visualización que simule una realidad o que extienda su percepción.

Tal es el caso de una planificación preoperatoria, la cirugía asistida por ordenador o una maqueta para la enseñanza de la cirugía. VVM se utiliza también en las ingenierías, en procesos que requieren herramientas avanzadas de visualización.

A nivel general, cuando se realiza un programa de visualización tridimensional se deben tomar en cuenta ciertos factores que son básicos a la hora de presentar el objeto de estudio, los cuales la mayoría de las veces son independientes de la investigación que se esté realizando, pero son necesarios para ver adecuadamente al objeto. Los factores a los que se hace referencia son, por ejemplo, la iluminación del objeto, propiedades de textura, movimientos de traslación, rotación, etc., los cuales a veces no se detallan cuando se observa un objeto, pero están ahí permitiendo que el objeto se visualice de una manera realística. El VVM ofrece todas estas características y muchas más, por ello, en este trabajo se aprovechan las potencialidades y estructura de esta API para realizar el mallado geométrico, darle propiedades de iluminación al objeto, movimientos de traslación, rotación, color e interacción con el usuario.

Un esquema básico de los elementos que debe tener una aplicación simple en el VVM, es la siguiente:

1. *Objetos en escena*: son los objetos con los que estamos trabajando en la ventana (visibles ó no). Estos pueden ser objetos como los obtenidos a través de una tomografía axial ó una resonancia magnética, a través de la codificación de ecuaciones matemáticas, o por manipulación de software a través de una interfaz con el usuario que los modela en el espacio y en el tiempo.

2. *Árbol de Procesos:* los procesos son los encargados de la actividad de los objetos. Ellos se organizan en árboles de procesos, con un proceso raíz que activa los demás procesos del árbol.

3. *Visualización:* este elemento a nivel global comprende la configuración de las características de visualización del objeto, por ejemplo, los tamaños de la ventana donde van a ser presentados los objetos, los modelos de iluminación, el tipo de cámara, etc. La visualización en el VVM se comparte entre tres entidades: VVM-Window, VVM-Scene y VVM-Model.

4. *Comunicación*: este elemento comprende las variables y los mensajes que van a compartir los objetos, entidades e interfaces.

5. *Interfaz*: es la comunicación del usuario con la aplicación. Esta es diseñada como una unidad totalmente independiente que utiliza un objeto VVM-CommunicationChannel para suplir la comunicación que se necesita en la cadena usuario-interfaz-escena.

Estos elementos se pueden apreciar a través del diagrama mostrado en la Figura 3.1. En este diagrama la figura en forma de elipse representa estructuras de datos, los rectángulos funciones y las flechas, la interacción o el flujo de datos entre los elementos. Las subrutinas implementadas en este trabajo fueron ubicadas en el árbol de procesos, ya que las mismas pueden ser vistas como una "caja negra" a la cual entra un objeto sin deformar y sale un objeto deformado, es decir, un proceso, el cual efectúa operaciones sobre los objetos. Para mayor referencia sobre el VVM puede consultarse [19] ó la página web *http://www.cpi.uc.edu.ve* del Centro de Procesamiento de Imágenes de la Universidad de Carabobo.

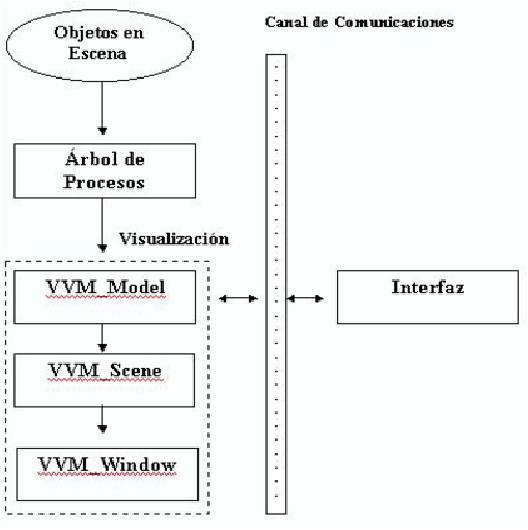


Figura 3.1: Esquema Básico de una aplicación simple en el VVM

3.3. El Mallado

Todo objeto tridimensional puede ser representado computacionalmente como la unión de un conjunto de figuras geométricas que dispuestas de manera organizada aproximan al objeto que esta siendo estudiado. Las figuras geométricas suelen ser todas de la misma forma, como por ejemplo, una malla de triángulos, una malla de hexaedros ó una malla de cubos, etc.

Para la representación del mallado de la vena, se utilizó la forma geométrica de un cilindro hueco de pared delgada debido a las similitudes que posee éste con respecto a una vena (sección 2.2). Al cilindro se le dan propiedades de elasticidad y rigidez a través de la ecuación (2.34) introduciendo los valores de las constantes de Lamé adecuados que modelan las propiedades del tejido. El mallado del cilindro consiste en una discretización a nivel general de tetraedros de cuatro nodos. Los

tetraedros son agrupados en cuñas formadas por tres de estos elementos. En la

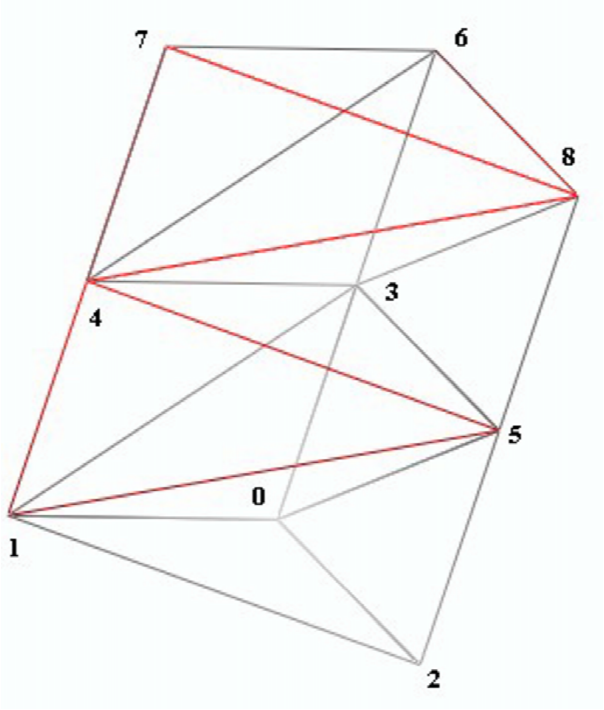


Figura 3.2: Elemento cuña dividido en 3 tetraedros

Figura 3.2 podemos apreciar dos cuñas una encima de la otra, cada una dividida en tres tetraedros. Los vértices de cada cuña son numerados de abajo hacia arriba y en sentido contrario de las agujas del reloj. Las cuñas son agrupadas como lo indica la Figura 3.3 para crear el cilindro y para identificar la posición (x,y,z) de cada uno de sus vértices, donde n y m representan el número de cortes verticales y el número de cortes horizontales con los que se forma el cilindro, respectivamente. La descomposición de la cuña en tetraedros puede verse en el Cuadro 3.1. En una vista desde arriba del cilindro podemos apreciar que la unión total de estas cuñas por cada capa horizontal queda como lo indica la Figura 3.4. Para finalmente obtener el mallado total mostrado por la Figura 3.5.

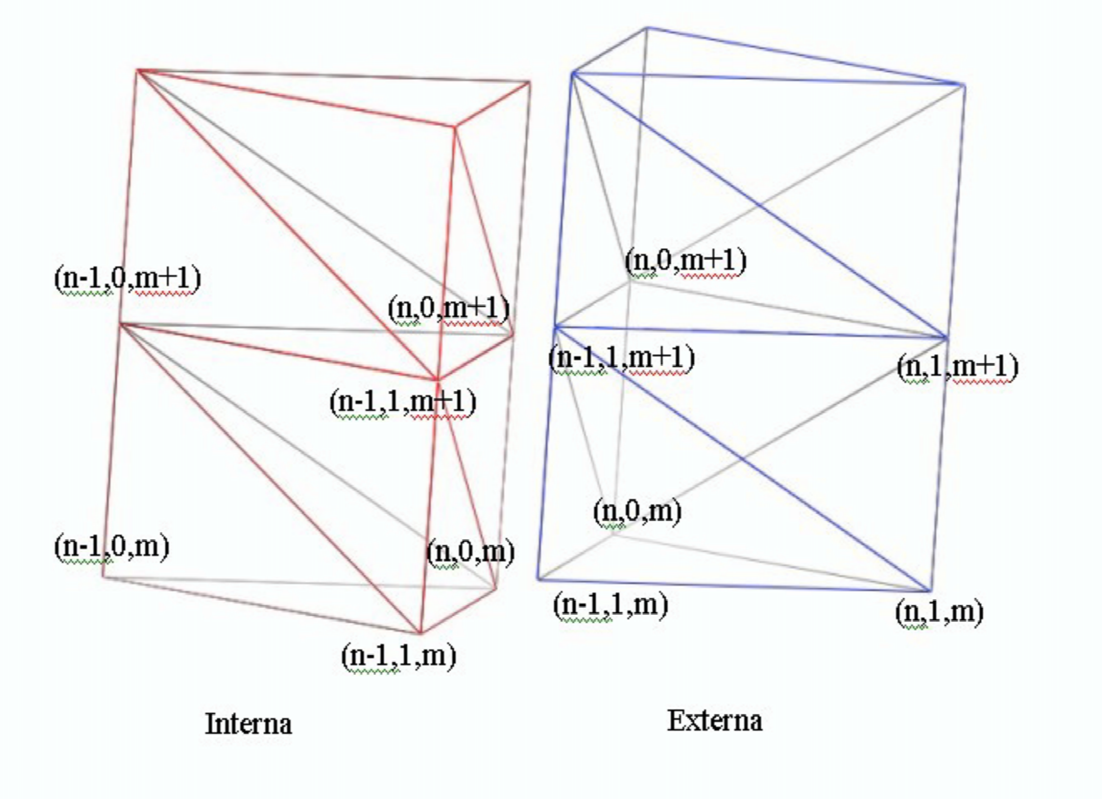


Figura 3.3: Dos cuñas adyacentes

Cuña	vértice 1	vértice 2	vértice 3	vértice 4
Interna Tetraedro 1	(n,0,m)	(n,0,m+1)	(n-1,0,m)	(n-1,1,m)
Interna Tetraedro 2	(n,0,m+1)	(n-1,0,m+1)	(n-1,0,m)	(n-1,1,m)
Interna Tetraedro 3	(n,0,m+1)	(n-1,0,m+1)	(n-1,1,m)	(n-1,1,m+1)
Externa Tetraedro 1	(n,1,m)	(n-1,1,m)	(n,0,m+1)	(n,0,m)
Externa Tetraedro 2	(n,1,m)	(n-1,1,m)	(n-1,1,m+1)	(n,0,m+1)
Externa Tetraedro 3	(n,1,m)	(n-1,1,m+1)	(n,1,m+1)	(n,0,m+1)

Cuadro 3.1: Descomposición de la cuña en tetraedros

3.4. Implementación de las Subrutinas de Deformación

Para implementar las subrutinas de deformación primero debemos llevar las ecuaciones que modelan el comportamiento del sistema a expresiones lógicamente

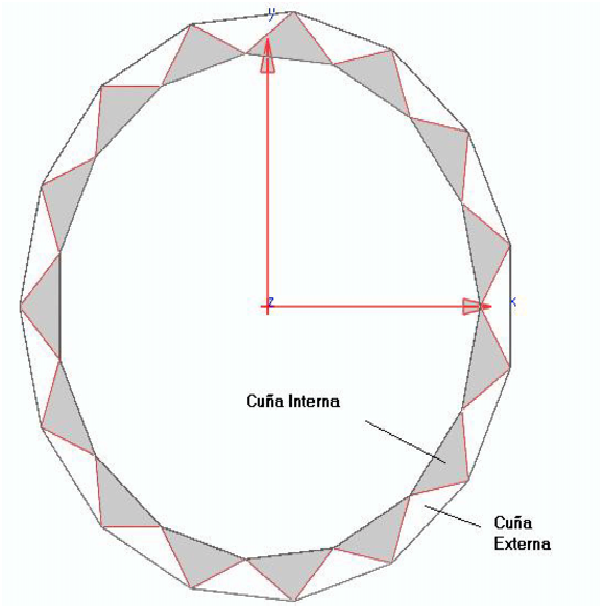


Figura 3.4: Organización de las cuñas para formar el cilindro

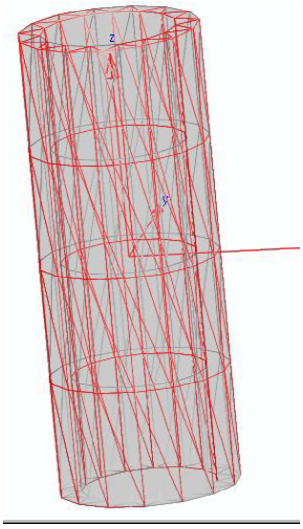


Figura 3.5: Malla del Cilindro

manejables por el computador. Para llevar a cabo esto, sustituiremos la ecuación (2.82) en (2.34), para obtener:

$$(\frac{m_i}{\triangle t^2} - \frac{\gamma_i}{2\triangle t})P_i^{t+1} = 2\sum_{j=0}^{3}([B_{pj}^{T_i}]U_j) + \frac{2m_i}{\triangle t^2}P_i^t - (\frac{m_i}{\triangle t^2} + \frac{\gamma_i}{2\triangle t})P_i^{t-1} \tag{3.1}$$

como vemos esta ecuación no es más que la ecuación principal que modela el comportamiento del sistema, sobre la cual desarrollaremos la sumatoria, quedando

$$\begin{aligned}(\frac{m_i}{\triangle t^2} - \frac{\gamma_i}{2\triangle t})P_i^{t+1} =& 2[B_{i0}^{T_i}]U_0 + 2[B_{i1}^{T_i}]U_1 + 2[B_{i2}^{T_i}]U_2 + 2[B_{i3}^{T_i}]U_0 + \frac{2m_i}{\triangle t^2}P_i^t \\ & - (\frac{m_i}{\triangle t^2} + \frac{\gamma_i}{2\triangle t})P_i^{t-1}\end{aligned} \tag{3.2}$$

la cual se lleva a forma matricial en vista que la mayoría de sus componentes son matrices y vectores, donde podemos obtener la siguiente expresión equivalente

$$\begin{aligned}(\frac{m_i}{\triangle t^2} - \frac{\gamma_i}{2\triangle t})P_{ik}^{t+1} =& 2[B_{k1}^{i0}]U_{01} + 2[B_{k2}^{i0}]U_{02} + 2[B_{k3}^{i0}]U_{03} + 2[B_{k1}^{i1}]U_{11} + 2[B_{k2}^{i1}]U_{12} \\ & + 2[B_{k3}^{i1}]U_{13} + 2[B_{k1}^{i2}]U_{21} + 2[B_{k2}^{i2}]U_{22} + 2[B_{k3}^{i2}]U_{23} + 2[B_{k1}^{i3}]U_{31} \\ & + 2[B_{k2}^{i3}]U_{32} + 2[B_{k3}^{i3}]U_{33} + \frac{2m_i}{\triangle t^2}P_i^t - (\frac{m_i}{\triangle t^2} + \frac{\gamma_i}{2\triangle t})P_i^{t-1}\end{aligned} \tag{3.3}$$

lo que es igual a

$$(\frac{m_i}{\triangle t^2} - \frac{\gamma_i}{2\triangle t})P_{ik}^{t+1} = \sum_{j=0}^{3}\sum_{z=0}^{3} 2[B_{kz}^{ij}]U_{jz} + \frac{2m_i}{\triangle t^2}P_i^t - (\frac{m_i}{\triangle t^2} + \frac{\gamma_i}{2\triangle t})P_i^{t-1} \tag{3.4}$$

ahora, se define U_{jz} como

$$U_{jz} = P_{jz}^{t+1} - P_{jz}^t \tag{3.5}$$

y se sustituye en (3.4), para obtener

$$(\frac{m_i}{\triangle t^2} - \frac{\gamma_i}{2\triangle t})P_{ik}^{t+1} = \sum_{j=0}^{3}\sum_{z=0}^{3} 2[B_{kz}^{ij}](P_{jz}^{t+1} - P_{jz}^t) + \frac{2m_i}{\triangle t^2}P_i^t - (\frac{m_i}{\triangle t^2} + \frac{\gamma_i}{2\triangle t})P_i^{t-1} \tag{3.6}$$

la cual puede ser vista como un sistema lineal de ecuaciones $Ax = b$ en su forma equivalente $x = Tx + c$ que será resuelto por el método iterativo de *Gauss-Seidel*, mencionado en la sección (2.12.1). Haciendo una analogía con la ecuación de este método (2.84) y (3.6), se puede observar que:

$$P_{ik}^{t+1} = x_i^{(k)} \tag{3.7}$$

$$\sum_{j=0}^{3}\sum_{z=0}^{3} 2[B_{kz}^{ij}](P_{jz}^{t+1}) = -\sum_{j=1}^{i-1}(a_{ij}x_j^{(k)}) \tag{3.8}$$

$$-\sum_{j=0}^{3}\sum_{z=0}^{3} 2[B_{kz}^{ij}](P_{jz}^{t}) + \frac{2m_i}{\triangle t^2}P_i^t = -\sum_{j=i+1}^{n}(a_{ij}x_j^{k-1}) \tag{3.9}$$

$$-(\frac{m_i}{\triangle t^2} + \frac{\gamma_i}{2\triangle t})P_i^{t-1} = b_i \tag{3.10}$$

$$(\frac{m_i}{\triangle t^2} - \frac{\gamma_i}{2\triangle t}) - 2[B_{kk}^{ii}] = a_{ii} \tag{3.11}$$

demostrando de esta manera, que la ecuación (3.6) representa la misma forma de la ecuación de Gauss-Seidel (2.84), excepto por el denominador, que puede ser obtenido de despejar P_{ik}^{t+1} de (3.6) . Haciendo la salvedad que en la sumatoria izquierda expresada en (3.8) no debe ser incluido el elemento $2[B_{kk}^{ii}]$, perteneciente al denominador de la ecuación general, lo cual podemos verificar fácilmente sí desarrollamos esta sumatoria y despejamos P_{ik}^{t+1} de (3.6). A través de todo esto hemos logrado conseguir una expresión manejable computacionalmente que describe el comportamiento del sistema.

3.4.1. Estructuras de Datos

Las estructuras de datos utilizadas por las subrutinas básicamente fueron diseñadas entorno a la matriz $B_{jk}^{T_i}$, definida en el segundo capítulo a través de

la ecuación (2.80). Se podría decir que la mayoría de los resultados arrojados por las deformaciones, dependen en gran parte de esta matriz. La función de $B_{jk}^{T_i}$ en la ecuación es representar una "constante" que describa las características de rigidez y elasticidad del elemento T_i, para nuestro caso un tetraedro. Sí en principio se parte de la idea de que el objeto a deformar es un tetraedro, $B_{jk}^{T_i}$ representará la contribución de rigidez que tiene el nodo k sobre el nodo j en el tetraedro T_i, que cambiará sus valores cada vez que se haga una deformación sobre el elemento. Como el tetraedro posee 4 nodos, tendremos en principio una matriz de $B's$ de dimensiones 4×4, donde cada elemento de esta matriz será a su vez una submatriz de dimensiones 3×3.

$$B = \begin{bmatrix} B_{00}^{T_i} & B_{01}^{T_i} & B_{02}^{T_i} & B_{03}^{T_i} \\ B_{10}^{T_i} & B_{11}^{T_i} & B_{12}^{T_i} & B_{13}^{T_i} \\ B_{20}^{T_i} & B_{21}^{T_i} & B_{22}^{T_i} & B_{23}^{T_i} \\ B_{30}^{T_i} & B_{31}^{T_i} & B_{32}^{T_i} & B_{33}^{T_i} \end{bmatrix} \tag{3.12}$$

Cada fila de esta matriz almacena la contribución de rigidez del nodo con respecto a él mismo y a sus vecinos. De aquí la importancia de esta matriz. Pero esto no termina aquí, ya se mencionó lo que pasa cuando es un sólo tetraedro, pero ¿que pasa cuando son varios y todos están unidos, uno adyacente al otro?, el cual es el caso en la malla del cilindro hueco (vena). Cuando existen varios tetraedros se parte del mismo principio expuesto anteriormente, existirán "por cada tetraedro" una matriz de coeficientes como (3.12), sólo con una consideración adicional, sí se tiene por ejemplo dos tetraedros T_1 y T_2 que comparten aunque sea un nodo j, el cálculo del coeficiente $B_{ij}^{T_1}$ para el nodo i con respecto a j del tetraedro T_1 , será la suma del coeficiente $B_{ij}^{T_1}$ para el tetraedro 1 y del coeficiente $B_{ij}^{T_2}$ para el tetraedro 2. Esta suma en realidad lo que significa es que ahora aparte de

sumar la contribución de todos los vecinos en el elemento (tetraedro), también se sumará la contribución de los vecinos adyacentes en otros tetraedros. Con esto lo que se consigue es que cuando se ejerza una fuerza sobre algún nodo de algún tetraedro, sean afectados también todos sus vecinos, de acuerdo a las propiedades que indique el coeficiente. Esto es lo que normalmente resulta en el MEF de ensamblar todas las ecuaciones de elemento finito [21].

En vista de lo anteriormente descrito se deben tener, para calcular los B de cada tetraedro mecanismos que nos provean de:

- La posición interna de cada nodo que estemos calculando dentro de cada tetraedro.
- La identificación de los nodos a los que pertenece cada tetraedro, es decir, dado un tetraedro saber cuales son los nodos que lo conforman.
- La identificación de los tetraedros a los que pertenece cada nodo, es decir, dado un nodo saber a que tetraedros pertenece.
- La identificación de los vecinos de cada nodo.

Estructura de Datos Voxel-Vert

Esta estructura provee información de los vértices a los cuales pertenece un tetraedro, es decir, dado un tetraedro los vértices que lo conforman. Está representada como un arreglo de enteros de dimensión $4N$, donde N es el número de tetraedros que conforman el sólido. En el arreglo cada elemento es un vértice,

el cual es representado por el "número global" que le corresponde a éste en la malla. Cada 4 elementos en el arreglo representa un tetraedro, comenzando de la posición 0 a la $N-1$. Esta cuatro-úpla de elementos se introduce organizada en el arreglo, donde el primer elemento de la misma representa el nodo interno 0 del tetraedro hasta el último elemento de la cuatro-úpla, que representa el nodo interno 3.

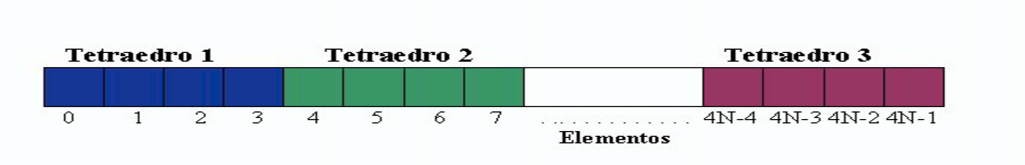


Figura 3.6: Estructura Voxel-vert

Estructura de Datos Vert-Tetra

Esta estructura nos provee información de los tetraedros a los que pertenece un vértice, es decir, dado un vértice cuales son los tetraedros a los que pertenece. Está representada por un arreglo de M elementos, donde M es el número de vértices que posee el objeto en la malla. Cada elemento del arreglo está conformado por una estructura que está dividida entre un entero que representa el número global de un vértice y un apuntador a una lista dinámica de enteros, que representa la lista de tetraedros a los que pertenece el vértice. Cada elemento de está lista dinámica es también una estructura que está dividida entre un entero que representa el número de un tetraedro y un apuntador al siguiente elemento de la lista.

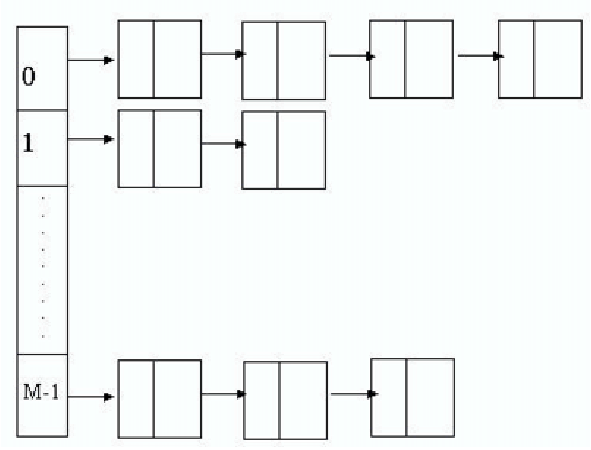


Figura 3.7: Estructura Vert-Tetra

Estructura de Datos Vert-enlaces

Esta estructura nos provee información acerca del número de enlaces que posee un nodo. Los enlaces de un nodo pueden ser vistos como la cantidad de segmentos de línea que están conectados a él. Para esta estructura se implementó también un arreglo de M elementos, donde M es el número de vértices que posee la malla. Cada elemento del arreglo es una estructura que está dividida entre un entero que representa un vértice a través de su número global en la malla y un apuntador a una lista dinámica de enteros. Cada elemento de la lista dinámica es una estructura de un apuntador a otro elemento de la lista y un entero. El elemento entero de esta lista es trabajado a nivel de bits, los primeros 28 bits son usados para representar el número global del vértice que posee un enlace con el vértice de estudio y los otros 4 bits representan la posición interna que tiene este vértice-enlace dentro del mismo tetraedro del vértice de estudio.

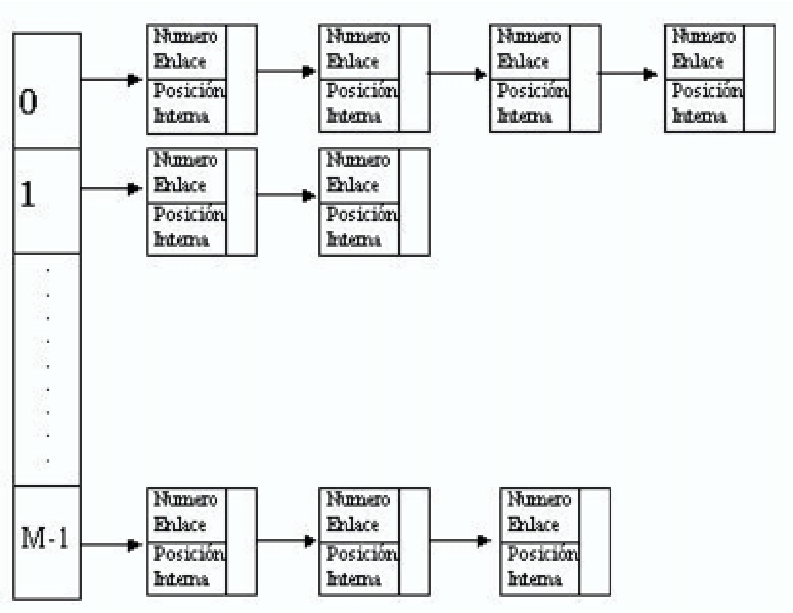


Figura 3.8: Estructura Vert-enlaces

Matriz de Coeficientes B^{Global}

Como ya se ha presentado al inicio del capítulo, la matriz B representa la matriz de rigidez. Esta matriz está definida a través de un arreglo de tamaño N, donde N es igual a la cantidad de tetraedros que posea la malla del sólido. Cada elemento de este arreglo es una submatriz de dimensiones 4×4 definida como en (3.12). Cada elemento de esta submatriz es a su vez una matriz $B_{jk}^{T_i}$ de dimensiones 3×3.

Matrices de Posición P

Las matrices de posición P, son aquellas mencionados en la ecuación (3.4), estos vienen dadas por: P_{ik}^{t+1},P_{ik}^{t},P_{ik}^{t-1}, los cuales representan las coordenadas (x, y, z)

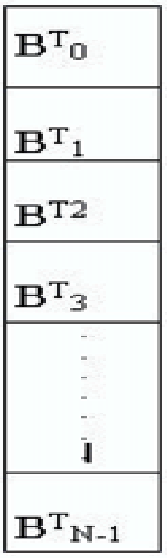


Figura 3.9: Estructura de la Matriz de Coeficientes B^{Global}

del vértice i en la coordenada k en los tiempos $t+1$, t y $t-1$ respectivamente. Todos son representados de la misma manera a través de una matriz de N filas por 3 columnas, donde cada columna corresponde a las posiciones x, y y z del vértice.

3.4.2. Prototipo de las Funciones

A continuación se presentan los prototipos de las principales funciones implementadas para la simulación de tejido blando con sus respectivos algoritmos de alto nivel:

Función Coeficiente

$[B^{ij}_{kw}]$= coeficiente(i,j,k,w, B^{Global},vert-tetra)

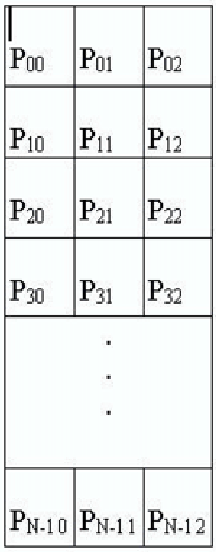


Figura 3.10: Estructura de los vectores P

Esta función devuelve la sumatoria de todos los coeficientes B_{kw}^{ij} de i con respecto a j en la fila k columna w de las matrices B^{ij} . Se sumarán tantos coeficientes como tetraedros comparta i con j. Donde $vert - tetra$ representa la estructura descrita en la sección (3.4.1) y B^{Global} la matriz de todos los coeficientes de todos los tetraedros de la malla, definida también en la sección (3.4.1). Para utilizar esta función deben haber sido calculados previamente la matriz B^{Global} y la estructura $vert - tetra$.

Algoritmo de Alto Nivel

- Entrada: indices i, j, k, w, del coeficiente a calcular. La Matriz global B^{Global} y la estructura $vert - tetra$.
- Salida: la sumatoria de todos los coeficientes B_{kw}^{ij} que comparten los vértices

i y j a través de los tetraedros a los que pertenecen ambos.

```
Coeficiente
 Inicio
  suma=0
  Para z=1 hasta (el numero de tetraedros que posea i)
   Para t=1 hasta (el numero de tetraedros que posea j)
    si (tetraedro de i = tetraedro de j) entonces
     suma = (B[del Tetraedro i] elemento [i][j][k][w]) + suma
    fsi
   fpara
  fpara
 Retornar (suma)
 Fin
```

Función Matriz de Rígidez

[C]= Matriz de Rígidez(i, k, B^{Global}, P^t, P^{t+1}, vert-tetra, vert-enlaces)

Esta función calcula la doble sumatoria definida en la ecuación (3.6) a partir de la estructura $vert - enlaces$ que indica los enlaces que tiene un vértice con sus vecinos. Cuando se se utiliza esta función en realidad estamos realizando el proceso de ensamblaje de las matrices locales de tetraedros.

Siendo C un número flotante, resultado de la suma de todos los coeficientes de todos los vértices vecinos a $P[i][k]$ más el coeficiente de él mismo, donde i denota el

número global del vértice al que se le va a calcular la sumatoria y k la coordenada a calcular (x, y ó z), que son denotadas por (0, 1 ó 2) respectivamente. B^{Global}, vert-tetra, vert-enlaces, representan las estructuras definidas en la sección (3.4.1) y P^t, P^{t+1}, son las matrices que contienen las coordenadas (x,y,z) de los vértices, en un tiempo t y $t+1$.

Algoritmo de Alto Nivel

- Entrada: el número del vértices que se van a calcular en la sumatoria (i), la coordenada a calcular de ese vértice (j), la matriz global de coeficientes B^{Global}, las estructuras vert-enlaces y vert-tetra y las matrices con las posiciones (x,y,z) de los vértices.
- Salida: un número real, que representa el resultado de la sumatoria de todos los coeficientes que están relacionados con el vértice P_{ik}

```
Matriz de Rigidez
Inicio
 suma=0
 Para (cada enlace j que tenga P[i][k]) y
  Para (cada coordenada w)
    suma = calcular coeficiente(i,j,k,w,B[global],vert-tetra)+ suma
  fpara
 fpara
 Retornar(suma)
 Fin
```

Función Principal Resolver Sistema

P^{t+1}=resolver-sistema(Fe,Fr,Nv,P^{t+1},P^{t},P^{t-1},B^{Global},vert-tetra,vert-enlaces,m,ro,$\triangle t$)

Esta función resuelve la ecuación (3.6) a través del método de *Gauss-Seidel*, obteniéndose de esta manera los valores de la matriz P^{t+1}.

Recibe todos los valores descritos en las funciones anteriores y además toma: una función de fuerza externa (Fe), una función para verificar las condiciones de frontera (Fr), la cantidad de vértices que conforman la malla (Nv), el coeficiente de masa (m), el coeficiente de amortiguamiento (ro) y el tamaño de cada intervalo de tiempo ($\triangle t$).

La fuerza externa Fe y la condición de frontera Fr, están definidas como funciones que pueden ser modificadas a gusto del programador, para hacer el algoritmo reusable. Dependiendo de como se definan la fuerza externa y las condiciones de frontera, pueden ser aplicadas al objeto "cargas puntuales" o "cargas sobre toda la malla". Es importante aclarar que cuando se aplica una fuerza externa al objeto, esta fuerza esta siendo ejercida sobre toda la malla, es decir, sobre cada vértice P_{ik}. Para lograr una carga puntual sobre un vértice, debemos igualar la fuerza externa a 0, y establecer en la matriz P^{t+1} el desplazamiento que queramos ejercer sobre el vértice donde deseamos realizar la carga puntual, es decir, que la fuerza vendrá dada como un desplazamiento del vértice.

El criterio de parada del algoritmo esta dado por una tolerancia del error de 10^{-3}, deducido de la ecuación (2.85).

Algoritmo de Alto Nivel

- Entrada: la función de fuerza externa Fe, la función de condiciones de frontera Fr, la cantidad de vértices en la malla, las matrices de posición de los vértices en el tiempo (P^{t+1},P^{t},P^{t-1}),la matriz de coeficientes global B^{Global}, las estructuras vert-tetra y vert-enlaces y las constantes m,ro,dt.
- Salida: la matriz de posición de los vértices P^{t+1} en el tiempo $t+1$.

```
Resolver Sistema
 Inicio
 Mientras (el error sea mayor que 0.001)
  Para (cada P[i][k])
    P[i][k]= calcular ecuacion (3.7)
    P[i][k]= P[i][k] + calcular fuerza externa
    evaluar condiciones de frontera en P[i][k]
  fpara
 fmientras
```

Para mayor detalle acerca de la implementación de las funciones, consultar el apéndice al final de este trabajo.

Capítulo 4

RESULTADOS Y CONCLUSIONES

4.1. Resultados Numéricos

Las pruebas para las simulaciones fueron evaluadas sobre dos factores: (1) el realismo lógico que presentan de acuerdo a la deformación que se le aplique y (2) el tiempo computacional que toma presentar cada cuadro.

Es importante recordar que en este trabajo no se realizó la detección de colisiones y las pruebas fueron realizadas en base a situaciones en las que se adapta el modelo lineal, es decir, deformaciones de carga axial. Recordemos que la gráfica esfuerzo-deformación (figura 2.1) del tejido blando describe un comportamiento lineal sobre una región acotada antes y después por periodos de deformaciones no-lineales. Básicamente lo que se persigue con estos resultados es la aplicación de un modelo sencillo que describa de manera aproximada el comportamiento del tejido blando bajo ciertas deformaciones. Los valores de las constantes que proporcionan

al modelo las características de elasticidad y rigidez, es decir, los coeficientes de Lamé, fueron tomados de [23].

Los tiempos de cómputo que sirvieron de referencia para ser comparados con los resultados del programa fueron tomados de [23], [31] y [32], los cuales realizaron trabajos de simulación pero con modelos no-lineales utilizando diferentes técnicas de optimización. Los tiempos obtenidos en estas investigaciones fueron en promedio de 23 cuadros por segundo en un procesador (Pentium III de 500 a 800 Mhz) con una malla de 2000 tetraedros. Es de hacer notar, que estos tiempos los obtienen trabajando sobre un esquema no-lineal, realizando deformaciones sólo sobre una parte de la malla y no la malla completa.

En la investigación presente, las deformaciones fueron realizadas sobre toda la malla, bajo un procesador AMD Athlon XP2000 de 1,66Ghz, donde se obtuvo en principio un tiempo de 6 cuadros por segundo sobre una malla de 1890 tetraedros, para luego ser mejorado a 17 cuadros por segundo, a través de la optimización del código.

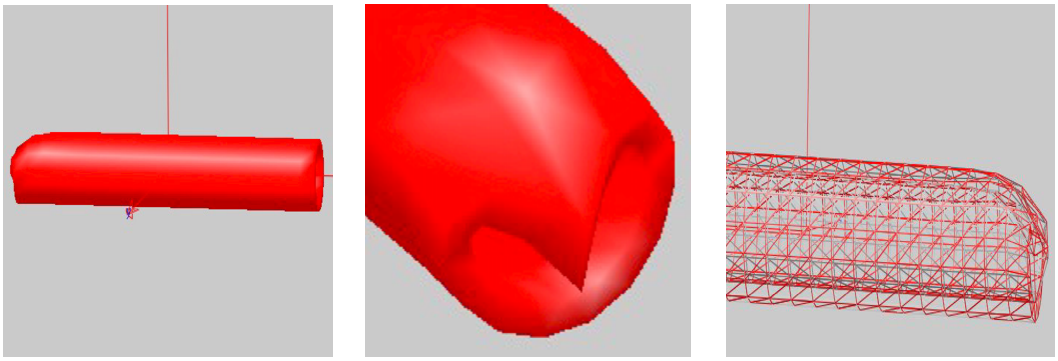
A continuación se presentan pruebas en las que son consideradas cargas axiales simples (sobre un eje de coordenadas) y cargas axiales combinadas (sobre varios ejes coordenados). Las deformaciones en este programa son aplicadas *a través de desplazamientos* que se efectúan sobre algún punto (carga puntual) ó sobre toda la malla, en virtud de que estos son los requerimientos que pide el modelo.

Prueba 1: Esta es una prueba donde se aplica un desplazamiento sobre el eje x al vértice 0. En esta se obtienen tiempos de 17 cuadros por segundo. Los

parámetros de entrada y las figuras resultantes pueden ser vistos en la tabla y figura (4.1), respectivamente.

Parámetro	Valor
Radio Interno	0.3
Radio Externo	0.4
Cortes Verticales	15
Cortes Horizontales	20
Altura	5
Número de Tetraedros	1890
Coeficiente de masa	3
Coeficiente de amortiguamiento	2
Intervalo de tiempo $\triangle t$	0.1
Coeficiente de Lamé λ	200000
Coeficiente de Lamé μ	100000
Vértice a desplazar	0
Posición Presente (x,y,z)	(0.3,0,-2.5)
Posición siguiente (x,y,z)	(-0.3,0,-2.5)

Cuadro 4.1: Valores de los parámetros para la prueba 1



a) Cilindro Completo b) Acercamiento c) La malla

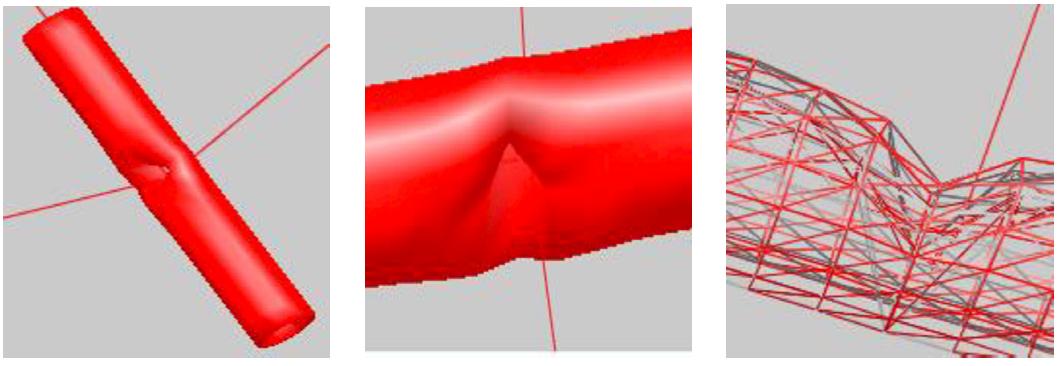
Figura 4.1: Resultados Prueba 1.

Prueba 2: En esta prueba se utilizan los mismos valores que se utilizaron en la tabla (4.1), sólo que ahora cambiamos los valores del coeficiente de masa y el

vértice a desplazar en el eje de coordenadas x. Cuando cambiamos el coeficiente de masa, estamos cambiando los valores de peso de cada partícula dentro del cilindro, por ende lo estamos haciendo más pesado ó más ligero.

Parámetro	Valor
Radio Interno	0.3
Radio Externo	0.4
Cortes Verticales	15
Cortes Horizontales	20
Altura	5
Número de Tetraedros	1890
Coeficiente de masa	**0.3**
Coeficiente de amortiguamiento	2
Intervalo de tiempo $\triangle t$	0.1
Coeficiente de Lamé λ	200000
Coeficiente de Lamé μ	100000
Vértice a desplazar	**330**
Posición Presente (x,y,z)	(0.3, 0, 0.119048)
Posición siguiente (x,y,z)	(-0.4, 0, 0.119048)

Cuadro 4.2: Valores de los parámetros para la prueba 2



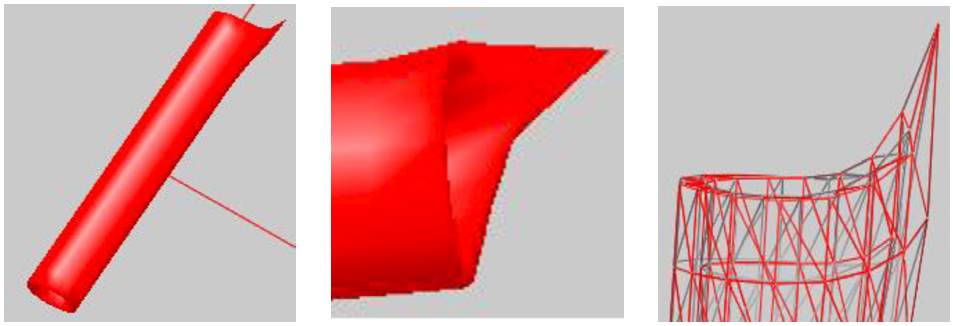
a) Cilindro Completo b) Acercamiento c) La malla

Figura 4.2: Resultados Prueba 2.

Prueba 3: En esta prueba se efectúa un desplazamiento del vértice 659 sobre todos los ejes coordenados. Los parámetros son iguales a la tabla (4.2) , sólo se varia el coeficiente de amortiguamiento y el vértice a desplazar. Cuando cambiamos los valores del coeficiente de amortiguamiento estamos cambiando la resistencia del material a la fuerza externa que se ejerce sobre el sólido.

Parámetro	Valor
Radio Interno	0.3
Radio Externo	0.4
Cortes Verticales	15
Cortes Horizontales	20
Altura	5
Número de Tetraedros	1890
Coeficiente de masa	3
Coeficiente de amortiguamiento	**0.2**
Intervalo de tiempo $\triangle t$	0.1
Coeficiente de Lamé λ	200000
Coeficiente de Lamé μ	100000
Vértice a desplazar	**659**
Posición Presente (x,y,z)	(0.391259 , -0.0831646 , 2.5)
Posición siguiente (x,y,z)	(0.5, 0.1 , 3)

Cuadro 4.3: Valores de los parámetros para la prueba 3



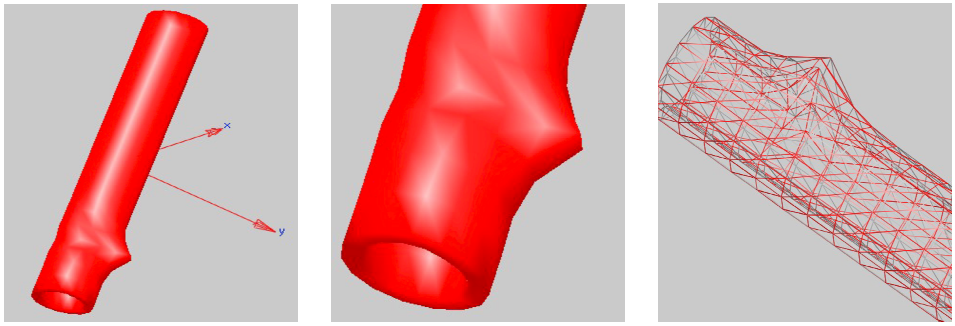
a) Cilindro Completo b) Acercamiento c) La malla

Figura 4.3: Resultados Prueba 3.

Prueba 4: En esta prueba se efectúa un desplazamiento sobre el vértice 120 en el eje y. Los valores de entrada se mantienen igual que en la tabla (4.3), cambiando sólo las constantes de Lamé, λ y μ. Cuando cambiamos las constantes de Lamé, estamos cambiando las características de rigidez y elasticidad del material, lo que comúnmente es llamado módulo de Poisson y módulo de elasticidad.

Parámetro	Valor
Radio Interno	0.3
Radio Externo	0.4
Cortes Verticales	15
Cortes Horizontales	20
Altura	5
Número de Tetraedros	1890
Coeficiente de masa	3
Coeficiente de amortiguamiento	0.2
Intervalo de tiempo $\triangle t$	0.1
Coeficiente de Lamé λ	**2000**
Coeficiente de Lamé μ	**400**
Vértice a desplazar	**120**
Posición Presente (x,y,z)	(0.3 , 0 , -1.54762)
Posición siguiente (x,y,z)	(0.3, 0.5 , -1.54762)

Cuadro 4.4: Valores de los parámetros para la prueba 4



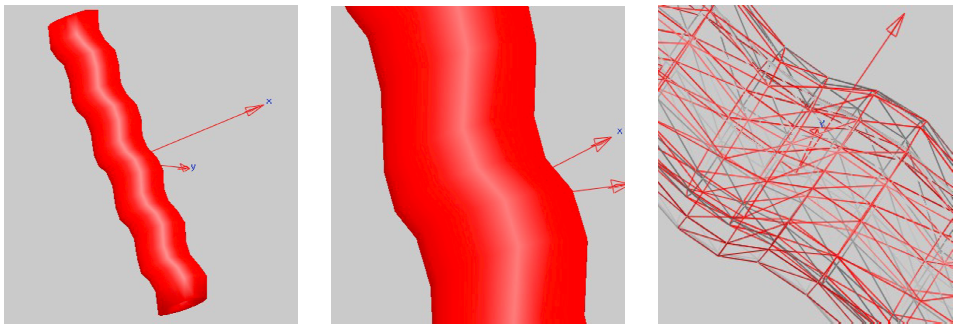
a) Cilindro Completo b) Acercamiento c) La malla

Figura 4.4: Resultados Prueba 4.

Prueba 5: En esta prueba se efectúa un desplazamiento en todos los vértices de la malla, a través de una fuerza externa senoidal igual a $(10000 * sin(10 * 3{,}14159265358979 * P^t_{i2}/4{,}9))$, donde P^t_{i2} es la coordenada z de cada vertice i de la malla. Los parámetros de entrada se mantienen iguales a los de la tabla (4.1).

Parámetro	Valor
Radio Interno	0.3
Radio Externo	0.4
Cortes Verticales	15
Cortes Horizontales	20
Altura	5
Número de Tetraedros	1890
Coeficiente de masa	3
Coeficiente de amortiguamiento	0.2
Intervalo de tiempo $\triangle t$	0.1
Coeficiente de Lamé λ	200000
Coeficiente de Lamé μ	100000
Vértice a desplazar	**TODA LA MALLA**
Posición Presente (x,y,z)	
Posición siguiente (x,y,z)	**Se aplica una fuerza externa senoidal**

Cuadro 4.5: Valores de los parámetros para la prueba 5



a) Cilindro Completo b) Acercamiento c) La malla

Figura 4.5: Resultados Prueba 5.

4.2. Conclusiones y Recomendaciones

- Las deformaciones aplicadas en función de los desplazamientos presentan una respuesta "semejante" a la deformación que podría ocurrir en la realidad. Sin embargo, es importante destacar que el modelo no considera situaciones de ruptura del sólido y la detección de colisiones tanto internas como externas a éste, por lo que en ningún momento podríamos apreciar este tipo de situaciones bajo cualquier prueba de estudio.

- En todos los casos la tolerancia del error en la aproximación a las coordenadas de los vértices por el método iterativo de Gauss-Seidel fue de 10^{-3}, exactitud generalmente alcanzada en la 1era ó segunda iteración. Por lo que no se considera que haga falta cambiar el método a menos que se necesite mayor precisión.

- El tiempo de cómputo obtenido para una malla de 1890 tetraedros fue de 17 cuadros por segundo trabajando sobre toda la malla, el cual puede disminuir ó aumentar dependiendo de la cantidad de tetraedros que se utilicen. Sí se utiliza una menor cantidad de tetraedros los tiempos serán menores y en caso contrario serán mayores. Este tiempo aunque puede ser considerado relativamente bueno-

  según las comparaciones hechas con otros trabajos [32],[31],[23], puede ser aún mejorado en las situaciones donde se aplique sólo una carga puntual al sólido, ya que bajos estas condiciones, se puede trabajar estrictamente sólo con una parte de la malla y no con la malla completa, lo cual disminuiría la cantidad de vértices a calcular por iteracción.

- A través de la formulación del método del elemento finito se obtuvieron diversas subrutinas que permiten simular deformaciones, no sólo de tejido blando, sino también de cualquier otro tipo de material. Esto lo podemos lograr simplemente cambiando las propiedades del material a través de los coeficientes de Lamé y construyendo la malla del sólido a simular bajo la estructura requerida por el módulo de software aquí implementado.

- La observación y validación de las pruebas fue algo fuerte, ya que no se contó con los materiales e instrumentación necesarios para validar experimentalmente los resultados de las pruebas de esta investigación. Por ello, sólo se contó con el visto bueno de algunos profesores del Departamento de Ingeniería Mecánica de la Universidad de Carabobo, entre ellos el Profesor Edwin Peña y con el sentido lógico que nos dan nuestras experiencias adquiridas en la vida diaria.

4.3. Trabajos Futuros

- Implementar una interfaz gráfica de usuario que permita plantear problemas de simulación y deformación de una manera amigable y sencilla, permitiendo la configuración de las funciones de fuerza externa y condiciones de frontera a través de la misma interfaz.

- Implementar funciones que permitan la detección de colisiones en el sólido.

- Implementar un modelo no-lineal para la simulación de deformaciones, que incluyan los casos de ruptura del sólido.

- Reformular las subrutinas implementadas para mejorar los tiempos de cómputo.

Apéndice A

Codigo Fuente de la Implementación

```
% #include "VVM_Classes.h"
% #include "VVM_Tetrahedras.h"
% #include "VVM_UnstructureVolumeData.h"
% #include "VVM_UnstructureVolumeModel.H"
% #include <math.h>
% #include <time.h>
% #include "VVM_Window.h"
% #include "VVM_SurfaceSource.h"
% #include "VVM_GeometrySource.h"

% extern   bool VVM_Key_F4, VVM_Key_F5;
% extern   unsigned char VVM_key;
% extern   VVM_Panel_GLUI *ActivePannel;
```

```
% extern   VVM_Window *asocWindow[100];
% extern int asoWindowID[100];

% char  New_Name[12] = {"max 10"};
% int   nodo=0;  // nodo donde se va a efectuar la deformacion
% int   numiter; // numero de iteraccion actual
% double numiter2;
% float r,R;         // Radio interno y externo del cilindro
% int   N,M;         // Numero de cortes verticales y horizontales
% float H;           // Altura del cilindro
% int   nverts;      // Numero de vertices en el solido
% int   ntetras;     // Numero de tetraedros en el solido
% float *vertics;    // Arreglo de vertices
% int   *tetra;      // Arreglo de tetraedros
% float lambda,miu;  // Coeficientes de Lame
% float m;           // Masa de cada punto en el volumen
% float ro;          // Coeficiente de Amortiguamiento
% float dt;          // Paso en el tiempo
% float (*p)[3];     // Matriz de vertices
% float (*Pprox)[3]; // Puntos proximos a desplazar el solido
% float (*pant)[3];  // Posiciones anteriores del s\'{o}lido
% float (*tetra_coeficiente)[4][4][3][3];  // Matriz de coeficientes B
% float med,roed,mx2,constant,medmroed,dt2;
% int *voxel_verts;
```

```
% VVM_vertArray *Verts = Verts->New();  // Arreglo de Vertices
% VVM_Tetrahedras  *Cells =Cells->New(); // Arreglo de tetraedros
% VVM_UnstructureVolumeData *VOL = VOL->New();
% cell_list **vert_tetra;
% cell_list **vert_enlaces;
% clock_t startt, finish;
% *defo_button;

% float _Red[] ={
                 0.400000f, 0.000000f, 0.000000f, 1.000000f,
                 1.000000f, 0.000000f, 0.000000f, 1.000000f,
                 0.800000f, 0.800000f, 0.800000f, 1.000000f,
                 0.000000f, 0.000000f, 0.000000f, 0.000000f,
                 20.0f
                };

% float _Gris[] ={
                  0.200000f, 0.200000f, 0.200000f, 1.000000f,
                  0.500000f, 0.500000f, 0.500000f, 1.000000f,
                  0.000000f, 0.000000f, 0.000000f, 1.000000f,
                  0.000000f, 0.000000f, 0.000000f, 0.000000f,
                  0.0f
                 };
```

```
//Inicializa los valores del objeto "Int"
//sobre el sistema

void inicializa_triangulo(VVM_InteractorProcess *Int)
{
 static VVM_SuperObject *e1 = (VVM_SuperObject *)Int->
 getDataPointer()->
 getObjectPointer("Cilindro","VVM_UnstructureVolumeData");
 e1->sys->init();
 numiter=0;
}

//Calcula el producto pto de un vector alfa de indice "a"
//con un vector alfa de indice "b" y devuelve una matriz
//resultante "c" de 3x3 cuya diagonal es igual al producto
// pto de los dos vectores.

void producto_pto (int a,int b, float (*alfa)[3], float (*c)[3])
{
  int i,j;
  float producto;
  producto=0;
   for (j=0;j<=2;j++)
   {
```

```
    producto= (alfa[a][j]*alfa[b][j]) + producto;
   }
   for (i=0;i<=2;i++)
   {
    for (j=0;j<=2;j++)
    {
     c[i][j]=0;     // Actua como matriz identidad
    }
   }
   c[0][0]=producto; // Elementos de la diagonal
   c[1][1]=producto;
   c[2][2]=producto;
}



//Devuelve una matriz c de 3x3, la cual es el resultado
// del producto tensorial entre un vector alfa de indice
// "a" con respecto a un vector alfa de indice "b".

void producto_cruz(int a,int b, float (*alfa)[3], float (*c)[3])
{
 int i,j;
 for (i=0;i<=2;i++)
 {
```

```
  for (j=0;j<=2;j++)
  {
   c[i][j]= alfa[a][i]*alfa[b][j];
  }
 }
}

void producto_vect(int i, int j,float (*p)[3],  float *m)
{
  // Calcula el producto vectorial de dos vectores
  // "i" es el 1er vector a multiplicar
  // "j" es el 2do vector a multiplicar
  // "p" es una matriz  donde se encuentran las coordenadas
  // de cada punto del volumen global
  // "p" es el vector resultante del producto vectorial
  m[0]= (p[i][1]*p[j][2]) - (p[i][2]*p[j][1]);
  m[1]= (-1)*((p[i][0]*p[j][2]) - (p[i][2]*p[j][0]));
  m[2]= (p[i][0]*p[j][1]) - (p[i][1]*p[j][0]);
}

//Calcula el volumen de un tetraedro cuyos ptos
//son i,j,k,w. numerados a nivel global, a trav\'{e}s
//del producto mixto de dos vectores
```

```
float calcular_vol_Tetraedro(int i, int j, int k,int w, float
(*p)[3])
{
  int z;
  float v,a,b,c;
  float *a1,*a2,*a3;
   a1= new float[3];
   a2= new float[3];
   a3= new float[3];
   for (z=0;z<=2;z++)
   {
    a1[z]= (p[i][z] - p[j][z]);
    a2[z]= (p[i][z] - p[k][z]);
    a3[z]= (p[i][z] - p[w][z]);
   }
  a= a1[0]*( a2[1]*a3[2] - a2[2]*a3[1] ); //Aplicacion del
  b= a1[1]*( a2[0]*a3[2] - a2[2]*a3[0] ); //producto mixto
  c= a1[2]*( a2[0]*a3[1] - a2[1]*a3[0] ); //3 vectores.
  v= fabs(a - b + c);
  return(v);
}

//Calcula los alfas para un tetraedro cuyos ptos
//globales son i,j,k,w. Son calculados segun la formula
```

```
// utilizada y varian de acuerdo a la posicion del
//solido.

void calcular_alfas (int i, int j, int z, int k, float (*p)[3],
float (*alfa)[3])
{
  int h;
  float v,*a,*b,*c,unoe6xv,munoe6xv;
  a=new float[3];
  b=new float[3];
  c=new float[3];
  v=calcular_vol_Tetraedro(i,j,z,k,p);
  unoe6xv= 1/(6*v);
  munoe6xv= -1/(6*v);
  // calculo alfa 0
  producto_vect(j,z,p,a);
  producto_vect(z,k,p,b);
  producto_vect(k,j,p,c);
  for (h=0;h<=2;h++)
  {
   alfa[0][h]= (unoe6xv)*(a[h]+b[h]+c[h] );
  }
  //calculo alfa1
  producto_vect(z,k,p,a);
```

```
  producto_vect(k,i,p,b);
  producto_vect(i,z,p,c);
  for (h=0;h<=2;h++)
  {
   alfa[1][h]= (munoe6xv)*(a[h]+b[h]+c[h] );
   }
   //calculo alfa2
   producto_vect(k,i,p,a);
   producto_vect(i,j,p,b);
   producto_vect(j,k,p,c);
   for (h=0;h<=2;h++)
   {
    alfa[2][h]= (unoe6xv)*(a[h]+b[h]+c[h] );
   }
   //calculo alfa3
   producto_vect(i,j,p,a);
   producto_vect(j,z,p,b);
   producto_vect(z,i,p,c);
   for (h=0;h<=2;h++)
   {
    alfa[3][h]= (munoe6xv)*(a[h]+b[h]+c[h] );
   }
}
```

```
// Este procedimiento busca todos los coeficientes "B" de
// un pto "i" con respecto a "j". "k" y "w", corresponden a la
// direccion de la fila y la columna respectivamente del
// coeficiente en la matriz. Es necesario recordar que un
// pto puede tener varios coeficientes "B" de "i" con respecto
// a "j" dependiendo del tetraedro en que se encuentren.
// Este procedimiento ademas de buscar estos coeficientes,
// los suma para encontrar el valor del coeficiente global.

float coeficiente2(int i, int j, int k, int w, float
(*tetra_coeficiente)[4][4][3][3],cell_list **vert_tetra)
{
  float suma;
  int   nbits;
  suma=0;
  nbits=4;
  cell_list *_l;
  cell_list *_b;
  _l = vert_tetra[i];
  while(_l)
  {
   _b = vert_tetra[j];
   while(_b)
   {
```

```
    if (((_l->value>>nbits))== ((_b->value>>nbits)))
    {
     suma= tetra_coeficiente[(_l->value>>nbits)]
     [(_l->value & ( (1<<nbits) -1 ) )]
     [(_b->value & ( (1<<nbits) -1 ) )][k][w] + suma;
    }
     _b = _b->next;
    }
     _l = _l->next;
    }
  return(suma);
}

// Calcula la sumatoria de todas las contribuciones de
// los vecinos, con respecto a un pto "i" en la coordenada "k".
// Donde k puede ser 0,1,2. que es equivalente a las
// coordenadas x,y,z de un vector.

float Matriz_de_rigidez(int i,int k, float
(*tetra_coeficiente)[4][4][3][3],float (*p)[3],float
(*Pprox)[3],cell_list **vert_tetra, cell_list **vert_enlaces) {
  float sumarp;    // Calcula la suma de todos los Ptos actuales
  float sumarpprox;// Calcula la suma de todos los Ptos proximos
  float sumatotal;
```

```
int z;
float B;
 sumarp    =0;     //Inicializacion de sumatorias
 sumarpprox=0;
 sumatotal =0;
 cell_list *_l;
 _l = vert_enlaces[i];
 while(_l)
 {
  for (z=0;z<=2;z++)
  {
   B=(coeficiente2(i,(_l->value),k,z,tetra_coeficiente,vert_tetra));
   sumarpprox= (2*B*Pprox[(_l->value)][z]) + sumarpprox;
   sumarp    = (2*B*p[(_l->value)][z])     + sumarp;
  }
  _l = _l->next;
 }
 for (z=0;z<=2;z++)
 {
  B=(coeficiente2(i,i,k,z,tetra_coeficiente,vert_tetra));
  if (z!=k)
  {
   sumarpprox= (2*B*Pprox[i][z]) + sumarpprox;
  }
```

```
     sumarp= (2*B*p[i][z]) + sumarp;
    }
    sumatotal= sumarpprox - sumarp;
 return(sumatotal);
}


//Procedimiento que resuelve un sistema de "n",
//ecuaciones lineales a traves del metodo de Gauss-Seidel.
// n= numero de vertices del solido. Esta  subrutina
// es utilizada cuando se consideran situaciones de carga
// puntual sobre el s\'{o}lido

void resolver_sistema2(int nodo,int ncubos, float (*p)[3],float
(*Pprox)[3],float (*pant)[3],float(*tetra_coeficiente)[4][4][3][3],
 cell_list **vert_tetra,cell_list **vert_enlaces, float m, float ro,
 float dt)
{
  int i,k,it;
  float dt2;
  float med,roed,mx2;
  mx2  = 2*m;
  dt2 = dt*dt;
  med = m/dt2;
```

```
  roed= ro/(2*dt);
  for (it=0;it<=3;it++) //iteracciones para calcular el sistema
  {
   for(i=0;i<ncubos;i++)
   {
    if (i != nodo)
    {
     for(k=0;k<=2;k++)
     {
      Pprox[i][k]= Matriz_de_rigidez(i,k,tetra_coeficiente,p,Pprox,
      vert_tetra,vert_enlaces) + ((mx2/dt2)*p[i][k]) -
      ((med+roed)*pant[i][k]) ;
      Pprox[i][k]= Pprox[i][k] / (med - roed -
      (2*(coeficiente2(i,i,k,k,tetra_coeficiente,vert_tetra))));
     }
    } //calculamos las coordenadas siguientes de cada uno de los
   }  //de los ptos pertenecientes al solido.
  }
}

// Este sistema hace lo mismo que resolver_ sistema2, solo que en
// este caso se consideran fuerzas externas que actuan sobre toda
// la malla del solido.
```

```
void resolver_sistema3(int nodo,int ncubos, float (*p)[3],float
(*Pprox)[3],float(*pant)[3],float(*tetra_coeficiente)[4][4][3][3],
cell_list **vert_tetra, cell_list **vert_enlaces, float m, float
ro, float dt)
{
  int i,k;   // contadores para los ciclos
  for(i=0;i<ncubos;i++)
  {
   for(k=0;k<=2;k++)
   {
    if (Pprox[i][2]>=0)
    {
     Pprox[i][k]= sumatoria2(i,k,tetra_coeficiente,p,Pprox,
     vert_tetra,vert_enlaces)+ 1000*sin(constant*p[i][2]) +
     ((mx2/dt2)*p[i][k]) - ((med+roed)*pant[i][k]) ;
     Pprox[i][k]= Pprox[i][k] /
     (medmroed - (2*(coeficiente2(i,i,k,k,tetra_coeficiente,
     vert_tetra))));
     cout << "Pprox["<< i<<"]["<< k<<"]"<<Pprox[i][k]<<endl;
    }
   }
  }
}
```

```
// Este procedimiento crea la estructura tetra_coeficiente,
// la cual nos indica dado el numero de un tetraedro todos sus
// respectivos coeficientes B. Dado los ptos i,j,k,w pertenecientes
// al tetraedro y el numero del tetraedro, calculamos y guardamos
// sus respectivos coeficientes.

void crear_tetra_coeficiente(int i,int j, int k, int w, int
&numtetra,float (*tetra_coeficiente)[4][4][3][3],float
(*p)[3],float lambda, float miu)
{
  float (*alfa)[3];
  float (*a)[3],(*b)[3],(*Id3)[3];
  int x,t,z,h,cont;
  float miue2,lambdae2;
  alfa = new float[4][3];
  a    = new float[3][3];
  b    = new float[3][3];
  Id3  = new float[3][3];
  miue2=miu/2;
  lambdae2=lambda/2;
  cont=0;
  calcular_alfas(i,j,k,w,p,alfa);
  //"i" es el nodo donde se esta ejerciendo la fuerza
  //"j" es un nodo perteneciente al tetraedro
```

```
  for(x=0;x<=3;x++)
  {
   for (t=0;t<=3;t++)
   {
    producto_cruz(x,t,alfa,a);
    producto_cruz(t,x,alfa,b);
    producto_pto(x,t,alfa,Id3);
    for (z=0;z<=2;z++)
    {
     for (h=0;h<=2;h++)
     {
      tetra_coeficiente[numtetra-1][x][t][z][h] =
       ((lambdae2)*a[z][h])+((miue2)*(b[z][h] +
        Id3[z][h]));
     } // Por definicion de B's coeficientes para calcular
    }  // calcular fuerzas
   }
  }
}

// Esta funcion unida a la anterior crean la matriz de
// coeficientes B del sistema

void evaluate_tetra_coeficiente(int   ntetras, int   *tetra, float
```

```
lambda, float miu,float (*tetra_coeficiente)[4][4][3][3], float
(*p)[3])
{
 int i,numtetra;
 numtetra=1;
 for(i=0;i<ntetras*4;i+=4)
 {
  crear_tetra_coeficiente(tetra[i],tetra[i+1],tetra[i+2],tetra[i+3],
  numtetra,tetra_coeficiente,p,lambda,miu);
  numtetra=numtetra+1;
 }
}

// Esta funcion es la que permite la interacci\'{o}n del usuario con el
// programa. Esta permite visualizar la deformacion del objeto a
// traves del tiempo.


int Mover2(VVM_Object *Interactor)
{
 static VVM_InteractorProcess *Int =
 (VVM_InteractorProcess *)Interactor;
 static VVM_SuperObject *e1= (VVM_SuperObject *)Int->
 getDataPointer()->
```

```
getObjectPointer("Cilindro","VVM_UnstructureVolumeData");
static bool start = false;
if(VVM_Key_F4)
{
 start = !start;
 VVM_Key_F4=false;
}
if(start)
{
 startt = clock();
 numiter=numiter+1;
 numiter2=numiter2+1;
 resolver_sistema3(nodo,nverts,p,Pprox,pant,tetra_coeficiente,
 vert_tetra,vert_enlaces,m,ro,dt);
 copiar_vector(nverts,p,pant);
 copiar_vector(nverts,Pprox,p);
 matriz_a_vector(p,vertics,nverts);
 Verts->copyVert(vertics);
 VOL->Modified();
 if (numiter=4)
 {
  numiter=0;
  evaluate_tetra_coeficiente(ntetras,voxel_verts,lambda,miu,
  tetra_coeficiente,p);
```

```
  }
  finish = clock();
  elapsed_time = finish -  startt;
  cout<<"tiempo que toma:"<<elapsed_time <<endl;
 }
return 1;
}

//Esta funcion es la que realiza la malla del cilindro hueco a
//traves de cunas y tetraedros. "N" representa el numero de cortes
//verticales del cilindro, "M" el numero de cortes horizontales, "R" el radio
//externo, "r" el radio interno, "H" la altura, "nverts" y "ntetras" el numero
//de vertices y el numero de tetraedros por los que esta formada la
//malla respectivamente. "vertex" el arreglo de vertices, "tetra"
//el arreglo de tetraedros.

void realizar_malla(int N, int M, float R, float r, float H,int
nverts,int ntetras,float *vertex, int *tetra)
{
    //N es el numero de cu\~{n}as internas y extrenas
    //M es el numero de capas
    float *_vertex;            //auxiliar de nvertex
    int   *_tetra;             //auxiliar de tetra
    int   n, m, _n;
```

```
float z, alfa, d_z, d_alfa;
d_z = H/(M+1);              // dz de la altura
d_alfa = VVM_PI/N;          // d_alfa de cada cuna
// Inicializacion de vertices
for(m = 0, z = -H/2.0f, _vertex = vertex; m <= M+1; m++,
 z += d_z, _vertex += 3*N)
{
 for(n = 0, alfa = 0; n < N; n++, alfa += 2*d_alfa,
  _vertex += 3)
 {
  _vertex[0] = r*cos(alfa);
  _vertex[1] = r*sin(alfa);
  _vertex[2] = z;
  _vertex[3*N]   = R*cos(alfa + d_alfa);
  _vertex[3*N+1] = R*sin(alfa + d_alfa);
  _vertex[3*N+2] = z;
 }
}
// Inicializacion de tetraedros
for(m = 0, _tetra = tetra; m <= M; m++)
{
 for(n = 1, _n = 1; n <= N; n++, _n++, _tetra += 24)
 {
  if(_n == N) _n = 0;
```

```
// Tetraedros de la cu\~{n}a interna
_tetra[0] = _n + m*2*N;
_tetra[1] = _n + (m+1)*2*N;
_tetra[2] =  n - 1 + m*2*N;
_tetra[3] =  n - 1 + N + m*2*N;
_tetra[4] = _n + (m+1)*2*N;
_tetra[5] =  n - 1 + (m+1)*2*N;
_tetra[6] =  n - 1 + m*2*N;
_tetra[7] =  n - 1 + N + m*2*N;
_tetra[8]  = _n + (m+1)*2*N;
_tetra[9]  =  n - 1 + (m+1)*2*N;
_tetra[10] =  n - 1 + N + m*2*N;
_tetra[11] =  n - 1 + N + (m+1)*2*N;
// Tetraedros de la cu\~{n}a externa
_tetra[12] = _n + N + m*2*N;
_tetra[13] =  n - 1 + N + m*2*N;
_tetra[14] = _n + (m+1)*2*N;
_tetra[15] = _n + m*2*N;
_tetra[16] = _n + N + m*2*N;
_tetra[17] =  n - 1 + N + m*2*N;
_tetra[18] =  n - 1 + N + (m+1)*2*N;
_tetra[19] = _n + (m+1)*2*N;
_tetra[20] = _n + N + m*2*N;
_tetra[21] =  n - 1 + N + (m+1)*2*N;
```

```
      _tetra[22] = _n + N + (m+1)*2*N;
      _tetra[23] = _n + (m+1)*2*N;
     }
    }
}

// Esta subrutina permite armar la malla del cilindro hueco y
// efectuar deformaciones a trav\'{e}s de cargas puntuales.

void Crear_cilindro_hueco_dinamico2 () {
  lambda=2000;
  miu   =400;
  m     =3;
  ro    =0.2;
  dt    =0.1;
  cout<<"Introduzca el radio interno del cilindro: ";
  cin >> r;
  cout<<"Introduzca el radio externo del cilindro: ";
  cin >> R;
  cout<<"Introduzca el numero de cortes verticales: ";
  cin >> N;
  cout<<"Introduzca el numero de cortes horizontales: ";
  cin >> M;
  cout<<"Introduzca la altura del cilindro: ";
```

```
cin >> H;
nverts  = 2*N*(M+2);
ntetras = 6*N*(M+1);
p       = new float[nverts][3];
Pprox   = new float[nverts][3];
pant    = new float[nverts][3];
vertics = new float[3*nverts];
tetra   = new int  [4*ntetras];
char opcion;
realizar_malla(N,M,R,r,H,nverts,ntetras,vertics,tetra);
vector_a_matriz(vertics,p,nverts);
tetra_coeficiente= new float[ntetras][4][4][3][3];
Verts->setVertType(VVM_N3F_V3F_PNF);
Verts->setVertQtty(nverts);
Verts->initArray();
Verts->copyVert(vertics);
Cells->setArraySize(ntetras*4);
Cells->setVertQtty(nverts);
Cells->initArray();
Cells->copy(tetra);
Cells->evaluateOutFaceVerts();
VOL->setObjectName("tetraedro");
int vert_voxel_mode = VVM_VERT_VOXEL_AND_POSITION;
Cells->setVertVoxelMode(vert_voxel_mode);
```

```
cell_list **vert_tetra   = Cells->evaluateVertVoxels();
cell_list **vert_enlaces = Cells->evaluateVertVerts();
int       *voxel_verts   = Cells->evaluateVoxelVerts();
evaluate_tetra_coeficiente(ntetras,voxel_verts,lambda,miu,
tetra_coeficiente,p);
copiar_vector(nverts,p,pant);

copiar_vector(nverts,p,Pprox);
do
{
 cout <<"Introduzca el pto que desea desplazar
 (0 a "<<nverts-1<<"): ";
 cin  >>  nodo;
 cout <<"Esta es la posicion actual (x,y,z) del pto# "<<nodo<<":
 ("<<p[nodo][0]<<","<<p[nodo][1]<<","<<p[nodo][2]<<")"<<endl;
 cout <<"Indique la posicion proxima:"<<endl;
 cout <<endl<<"Posicion proxima en el eje x:";
 cin  >> Pprox[nodo][0];
 cout <<endl<<"Posicion proxima en el eje y:";
 cin  >> Pprox[nodo][1];
 cout <<endl<<"Posicion proxima en el eje z:";
 cin  >> Pprox[nodo][2];
 cout << "Desea introducir algun otro punto (s/n):"<<endl;
 cin  >> opcion;
```

```
}
while (opcion=='s');
resolver_sistema2(nodo,nverts,p,Pprox,pant,tetra_coeficiente,
vert_tetra,vert_enlaces,m,ro,dt);
copiar_vector(nverts,Pprox,p);
matriz_a_vector(p,vertics,nverts);
Verts->copyVert(vertics);
VOL->get(Verts);
VOL->get(Cells);
VOL->evaluateNormals();
VOL->setFrontMaterial(_Red);
VOL->setBackMaterial(_Gris);
VOL->setRenderMode(VVM_LINE);
VOL->setFaceMode(VVM_FRONT_AND_BACK);
VVM_UnstructureVolumeModel *MODEl = MODEl->New();
MODEl->get(VOL);
VVM_Scene *SCENE = SCENE->New();
SCENE->add(MODEl);
VVM_Window *WIN = WIN->New();
WIN->setPosition(30,30);
WIN->setSize(512,512);
WIN->init("Cilindro Hueco No Estructurado");
WIN->setWindowScene(SCENE);
WIN->Wide(3.0);
```

```
  WIN->start();
}


// Esta funcion permite deformar cualquier clase de objeto,
// siempre y cuando se introduzcan como parametros el numero de
// vertices y tetraedros que posee la malla y sus respectivos
// arreglos con la informaci\'{o}n sobre los vertices y los
// tetraedros. Adem\'{a}s de los coeficientes de Lame  adecuados al
// tipo de material a utilizar (lambda, miu). Esta funcion no
// encierra al programador a un tipo de malla en especial,
// permitiendole de esta forma dise\~{n}ar el solido como lo desee.
//

void deformacion(int nverts, int ntetras, float *vertics, int
*tetra,float lambda, float miu)
{
 vector_a_matriz(vertics,p,nverts);
 tetra_coeficiente= new float[ntetras][4][4][3][3];
  Verts->setVertType(VVM_N3F_V3F_PNF);
  Verts->setVertQtty(nverts);
  Verts->initArray();
  Verts->copyVert(vertics);
  Cells->setArraySize(ntetras*4);
```

```
Cells->setVertQtty(nverts);
Cells->initArray();
Cells->copy(tetra);
Cells->evaluateOutFaceVerts();
VOL->setObjectName("Cilindro");
int vert_voxel_mode = VVM_VERT_VOXEL_AND_POSITION;
Cells->setVertVoxelMode(vert_voxel_mode);
vert_tetra   = Cells->evaluateVertVoxels();
vert_enlaces = Cells->evaluateVertVerts();
voxel_verts  = Cells->evaluateVoxelVerts();
evaluate_tetra_coeficiente(ntetras,voxel_verts,lambda,
miu,tetra_coeficiente,p);
copiar_vector(nverts,p,pant); //La posicion anterior del solido
                              //es igual a la presente,empezando
copiar_vector(nverts,p,Pprox);
Verts->copyVert(vertics);
VOL->get(Verts);
VOL->get(Cells);
VOL->evaluateNormals();
VVM_InteractorProcess *INTERACTOR = INTERACTOR->New();
INTERACTOR->setFunction(Mover2);
INTERACTOR->SelectInput(0);
INTERACTOR->get(VOL);
inicializa_triangulo(INTERACTOR);
```

```
  VOL->setFrontMaterial(_Red);
  VOL->setBackMaterial(_Gris);
  VOL->setRenderMode(VVM_SMOOTH);
  VOL->setFaceMode(VVM_FRONT_AND_BACK);
  VVM_UnstructureVolumeModel *MODEl = MODEl->New();
  MODEl->get(VOL);
  VVM_Scene *SCENE = SCENE->New();
  SCENE->add(MODEl);
  SCENE->setProcess(INTERACTOR);
  VVM_Window *WIN = WIN->New();
  WIN->setPosition(30,30);
  WIN->setSize(512,512);
  WIN->init("Cilindro Hueco No Estructurado");
  WIN->setWindowScene(SCENE);
  WIN->Wide(3.0);
  WIN->start();
}

// Esta subrutina permite armar la malla del cilindro hueco y
// aplicar una fuerza externa al solido sobre toda la malla.

void cilindro_dinamico3()
{
 lambda=200000;
```

```
miu   =100000;
m     =3;
ro    =2;
dt    =0.1;
mx2  = 2*m;
dt2 = dt*dt;
med = m/dt2;
roed= ro/(2*dt);
constant= (10*3.14159265358979)/4.9;
medmroed= med - roed;
numiter=0;
numiter2=0;
cout<<"Introduzca el radio interno del cilindro: ";
cin >> r;
cout<<"Introduzca el radio externo del cilindro: ";
cin >> R;
cout<<"Introduzca el numero de cortes verticales: ";
cin >> N;
cout<<"Introduzca el numero de cortes horizontales: ";
cin >> M;
cout<<"Introduzca la altura del cilindro: ";
cin >> H;
nverts       = 2*N*(M+2);
ntetras      = 6*N*(M+1);
```

```
  p            = new float[nverts][3];
  Pprox        = new float[nverts][3];
  pant         = new float[nverts][3];
  vertics      = new float[3*nverts];
  tetra        = new int  [4*ntetras];
  realizar_malla(N,M,R,r,H,nverts,ntetras,vertics,tetra);
  deformacion(nverts,ntetras,vertics,tetra,lambda,miu);
}


// Programa principal donde se llaman a todas la subrutinas

void main()
{
  //Crear_cilindro_hueco_dinamico2();
  //Crear_cilindro_hueco_dinamico3();
  cilindro_dinamico3();
}
```

Apéndice B

Demostración de la Derivada de la Energía

A continuación se presenta un código realizado en Maple donde se demuestra que la derivada de la ecuación (2.79) (W_{lineal}) con respecto al desplazamiento es igual a la fuerza, ecuación (2.82).

```
> with(linalg);

  [BlockDiagonal, GramSchmidt, JordanBlock, LUdecomp, QRdecomp,
   Wronskian, addcol, addrow, adj, adjoint, angle, augment,
   backsub, band, basis, bezout, blockmatrix, charmat, charpoly,
   cholesky, col, coldim, colspace, colspan, companion, concat,
   cond, copyinto, crossprod, curl, definite, delcols, delrows,
```

```
   det, diag, diverge, dotprod, eigenvals, eigenvalues,
   eigenvectors, eigenvects, entermatrix, equal, exponential,
   extend, ffgausselim, fibonacci, forwardsub, frobenius,
   gausselim, gaussjord, geneqns, genmatrix, grad, hadamard,
   hermite, hessian, hilbert, htranspose, ihermite, indexfunc,
   innerprod, intbasis, inverse, ismith, issimilar, iszero,
   jacobian, jordan, kernel, laplacian, leastsqrs, linsolve,
    matadd, matrix, minor, minpoly, mulcol, mulrow, multiply,
    norm, normalize, nullspace, orthog, permanent, pivot,
    potential, randmatrix, randvector, rank, ratform, row, rowdim,
    rowspace, rowspan, rref, scalarmul, singularvals, smith,
    stackmatrix, submatrix, subvector, sumbasis, swapcol, swaprow,
    sylvester, toeplitz, trace, transpose, vandermonde, vecpotent,
    vectdim, vector, wronskian]
```

Sea Bij definida matricialmente de la siguiente manera:

```
> B:= linalg[matrix] (3,3,[lambda/2*bj*bk+mu/2*(2*bj*bk+cj*ck+dj*dk),
> lambda/2*bj*ck+mu/2*cj*bk,lambda/2*bj*dk+mu/2*dj*bk,
> lambda/2*cj*bk+mu/2*bj*ck,lambda/2*cj*ck+mu/2*(bj*bk+2*cj*ck+dj*dk),
> lambda/2*cj*dk+mu/2*dj*ck,lambda/2*dj*bk+mu/2*bj*dk,
> lambda/2*dj*ck+mu/2*cj*dk,lambda/2*dj*dk+mu/2*(bj*bk+cj*ck+2*dj*dk)]);
>
> B :=
```

```
>
>         [1/2 lambda bj bk + 1/2 mu (2 bj bk + cj ck + dj dk) ,
>
>         1/2 lambda bj ck + 1/2 mu cj bk ,
>
>         1/2 lambda bj dk + 1/2 mu dj bk]
>
>         [1/2 lambda cj bk + 1/2 mu bj ck ,
>
>         1/2 lambda cj ck + 1/2 mu (bj bk + 2 cj ck + dj dk) ,
>
>         1/2 lambda cj dk + 1/2 mu dj ck]
>
>         [1/2 lambda dj bk + 1/2 mu bj dk ,
>
>         1/2 lambda dj ck + 1/2 mu cj dk ,
>
>         1/2 lambda dj dk + 1/2 mu (bj bk + cj ck + 2 dj dk)]
>
```

Sea U por la izquierda definido

```
> Uiz:= linalg[matrix](1,3,[ujx,ujy,ujz]);
```

```
                    Uiz := [ujx    ujy    ujz]
```

Sea U por la derecha definido

```
> Uder:= linalg[matrix](3,1,[ukx,uky,ukz]);
```

$$\mathrm{Uder} := \begin{bmatrix} ukx \\ uky \\ ukz \end{bmatrix}$$

Sea Wlineal definido como Uiz*B*Uder

```
> resultado:= evala(evalm(Uiz&*B&*Uder));

  resultado :=

        [1/2 ukx ujx lambda bj bk + ukx ujx mu bj bk

         + 1/2 ukx ujx mu cj ck + 1/2 ukx ujx mu dj dk

         + 1/2 ukx ujy lambda cj bk + 1/2 ukx ujy mu bj ck
```

```
          + 1/2 ukx ujz lambda dj bk + 1/2 ukx ujz mu bj dk

          + 1/2 uky ujx lambda bj ck + 1/2 uky ujx mu cj bk

          + 1/2 uky ujy lambda cj ck + 1/2 uky ujy mu bj bk

          + uky ujy mu cj ck + 1/2 uky ujy mu dj dk

          + 1/2 uky ujz lambda dj ck + 1/2 uky ujz mu cj dk

          + 1/2 ukz ujx lambda bj dk + 1/2 ukz ujx mu dj bk

          + 1/2 ukz ujy lambda cj dk + 1/2 ukz ujy mu dj ck

          + 1/2 ukz ujz lambda dj dk + 1/2 ukz ujz mu bj bk

          + 1/2 ukz ujz mu cj ck + ukz ujz mu dj dk]


Sea la derivada de Wlineal con respecto a x
> dfx:=diff(1/2*ukx*ujx*lambda*bj*bk+ukx*ujx*mu*bj*bk
> +1/2*ukx*ujx*mu*cj*ck +1/2*ukx*ujx*mu*dj*dk
> +1/2*ukx*ujy*lambda*cj*bk+1/2*ukx*ujy*mu*bj*ck
```

```
> +1/2*ukx*ujz*lambda*dj*bk+1/2*ukx*ujz*mu*bj*dk
> +1/2*uky*ujx*lambda*bj*ck +1/2*uky*ujx*mu*cj*bk+
> 1/2*uky*ujy*lambda*cj*ck+1/2*uky*ujy*mu*bj*bk+uky*ujy*mu*cj*ck
> +1/2*uky*ujy*mu*dj*dk+1/2*uky*ujz*lambda*dj*ck
> +1/2*uky*ujz*mu*cj*dk+1/2*ukz*ujx*lambda*bj*dk
> +1/2*ukz*ujx*mu*dj*bk+1/2*ukz*ujy*lambda*cj*dk
> +1/2*ukz*ujy*mu*dj*ck+1/2*ukz*ujz*lambda*dj*dk
> +1/2*ukz*ujz*mu*bj*bk+1/2*ukz*ujz*mu*cj*ck+ukz*ujz*mu*dj*dk,ujx);

  dfx := 1/2 ukx lambda bj bk + ukx mu bj bk + 1/2 ukx mu cj ck

         + 1/2 ukx mu dj dk + 1/2 uky lambda bj ck + 1/2 uky mu cj bk

         + 1/2 ukz lambda bj dk + 1/2 ukz mu dj bk
```

Sea la derivada de Wlineal con respecto a y

```
> dfy:=diff(1/2*ukx*ujx*lambda*bj*bk+ukx*ujx*mu*bj*bk
> +1/2*ukx*ujx*mu*cj*ck +1/2*ukx*ujx*mu*dj*dk
> +1/2*ukx*ujy*lambda*cj*bk+1/2*ukx*ujy*mu*bj*ck
> +1/2*ukx*ujz*lambda*dj*bk+1/2*ukx*ujz*mu*bj*dk
> +1/2*uky*ujx*lambda*bj*ck +1/2*uky*ujx*mu*cj*bk
> +1/2*uky*ujy*lambda*cj*ck+1/2*uky*ujy*mu*bj*bk+uky*ujy*mu*cj*ck
> +1/2*uky*ujy*mu*dj*dk+1/2*uky*ujz*lambda*dj*ck
```

```
> +1/2*uky*ujz*mu*cj*dk +1/2*ukz*ujx*lambda*bj*dk
> +1/2*ukz*ujx*mu*dj*bk+1/2*ukz*ujy*lambda*cj*dk
> +1/2*ukz*ujy*mu*dj*ck+1/2*ukz*ujz*lambda*dj*dk
> +1/2*ukz*ujz*mu*bj*bk +1/2*ukz*ujz*mu*cj*ck+ukz*ujz*mu*dj*dk,ujy);

  dfy := 1/2 ukx lambda cj bk + 1/2 ukx mu bj ck

         + 1/2 uky lambda cj ck + 1/2 uky mu bj bk + uky mu cj ck

         + 1/2 uky mu dj dk + 1/2 ukz lambda cj dk + 1/2 ukz mu dj ck
```

Sea la derivada de Wlineal con respecto a z

```
> dfz=diff(1/2*ukx*ujx*lambda*bj*bk+ukx*ujx*mu*bj*bk
> +1/2*ukx*ujx*mu*cj*ck+1/2*ukx*ujx*mu*dj*dk
> +1/2*ukx*ujy*lambda*cj*bk+1/2*ukx*ujy*mu*bj*ck
> +1/2*ukx*ujz*lambda*dj*bk+1/2*ukx*ujz*mu*bj*dk
> +1/2*uky*ujx*lambda*bj*ck +1/2*uky*ujx*mu*cj*bk
> +1/2*uky*ujy*lambda*cj*ck+1/2*uky*ujy*mu*bj*bk+uky*ujy*mu*cj*ck
> +1/2*uky*ujy*mu*dj*dk+1/2*uky*ujz*lambda*dj*ck
> +1/2*uky*ujz*mu*cj*dk+1/2*ukz*ujx*lambda*bj*dk
> +1/2*ukz*ujx*mu*dj*bk+1/2*ukz*ujy*lambda*cj*dk
> +1/2*ukz*ujy*mu*dj*ck+1/2*ukz*ujz*lambda*dj*dk
> +1/2*ukz*ujz*mu*bj*bk+1/2*ukz*ujz*mu*cj*ck+ukz*ujz*mu*dj*dk,ujz);
```

```
  dfz = 1/2 ukx lambda dj bk + 1/2 ukx mu bj dk

          + 1/2 uky lambda dj ck + 1/2 uky mu cj dk

          + 1/2 ukz lambda dj dk + 1/2 ukz mu bj bk + 1/2 ukz mu cj ck

          + ukz mu dj dk
```

Obtenemos,

```
> resultado2:= linalg[matrix](3,1,[dfx,dfy, dfz]);

  resultado2 :=

         [1/2 ukx lambda bj bk + ukx mu bj bk + 1/2 ukx mu cj ck

          + 1/2 ukx mu dj dk + 1/2 uky lambda bj ck + 1/2 uky mu cj bk

          + 1/2 ukz lambda bj dk + 1/2 ukz mu dj bk]

         [1/2 ukx lambda cj bk + 1/2 ukx mu bj ck
```

```
          + 1/2 uky lambda cj ck + 1/2 uky mu bj bk + uky mu cj ck

          + 1/2 uky mu dj dk + 1/2 ukz lambda cj dk + 1/2 ukz mu dj ck

         ]

         [dfz]
```

Sea la fuerza definida como

```
> Fuerza:=evalm(2*B&*Uder);

  Fuerza :=

        [2 (1/2 lambda bj bk + 1/2 mu (2 bj bk + cj ck + dj dk)) ukx

         + 2 (1/2 lambda bj ck + 1/2 mu cj bk) uky

         + 2 (1/2 lambda bj dk + 1/2 mu dj bk) ukz]

        [2 (1/2 lambda cj bk + 1/2 mu bj ck) ukx +

        2 (1/2 lambda cj ck + 1/2 mu (bj bk + 2 cj ck + dj dk)) uky

         + 2 (1/2 lambda cj dk + 1/2 mu dj ck) ukz]
```

```
[2 (1/2 lambda dj bk + 1/2 mu bj dk) ukx

 + 2 (1/2 lambda dj ck + 1/2 mu cj dk) uky +

2 (1/2 lambda dj dk + 1/2 mu (bj bk + cj ck + 2 dj dk)) ukz]

>

>
```

Obteniendo finalmente que $resultado2 = Fuerza$. Demostrando de esta manera que la derivada de la energía lineal (W_{lineal}) con respecto al desplazamiento es igual a la fuerza.

Bibliografía

[1] P. BEYLOT, P. GINGINS, P. KALRA, M. THALMANN, W. MAUREL, D. THALMANN, J. FASEL. Modeling Using the Visible Human Dataset. Proc. Medical Informatics Europe, IOS Press, Genova. pp. 739-743. 1996.

[2] B. JIANG. *The Least-Squares Finite Element Method. Springer.*1998.

[3] R. BURDEN, D. FAIRES. *Análisis Numérico.* International Thomson Editores. Sexta Edición. 1997.

[4] T. CHANDRUPATLA, A. BELEGUNDU. *Introducción al Estudio del Elemento Finito en Ingeniería.* Pearson. Segunda Edición. 1999.

[5] P. CIARLET. Mathematical Elasticity. Vol.1: Three dimensional elasticity. Elsevier Science Publishers B.V.. 1988.

[6] D. DE CECHIS. *Una Librería para resolver ecuaciones diferenciales parciales elípticas en 2d por el método de elemento finito mínimo cuadrado.* Tesis de Grado. Universidad de Carabobo. 2001.

[7] F. DI SIMONE. *Solución de Problemas de elasticidad plana por el método de elementos de frontera mediante el uso de un microcomputador*. Trabajo de Ascenso. Universidad Central de Venezuela. Caracas. 1985.

[8] L. ELSGOLTZ. Ecuaciones Diferenciales y Cálculo Variacional. MIR-Moscú. Segunda Edición. 1977.

[9] G. FINN. *Histología*. Médica Panamericana, Tercera Edición. 1999.

[10] P. FISHBANE, S. GASIOROWICZ. *Física para Ciencias e Ingeniería*. Vol. 1. Prentice-Hall. pp.92-103,377-394. 1993.

[11] Y.C. FUNG. *Foundations of Solid Mechanics*. Prentice-Hall.pp.93-191 1965.

[12] S. GIBSON. *A Survey of Deformable Modeling in Computer Graphics*. Mitsubishi Electric Research Laboratory. TR-97-19. 1997.

[13] P. GINGINS, P. KALRA, P. BEYLOT, M. THALMANN, J. FASEL. *Using VHD to Build a Comprehensive Human Model*.The Visible Human Project Conference. Bethesda, Maryland, USA, Oct. 7-8, 1996.

[14] T. HARPER, K. ROW. *Resistencia de Materiales*. Singer. 1992.

[15] P. KALRA, P. BEYLOT, P. GINGINS, M. THALMAN, P. VOLINO, P. HOFFMEYER, J. FASEL, F. TERRIER. *Topological Modeling of Human Anatomy Using Medical Data*. Proc. Computer Animation '95. IEEE Computer Society, 172-80, Genova, 1995.

[16] W. MAUREL, D. THALMANN. *Human Upper Limb Modeling including Scapulothoracic Constraint and Joint Sinus Cones*. Computers Graphics. Vol 24. Num.2, 2002.

[17] W. MAUREL, D. THALMANN, P. HOFFMEYER, P. BEYLOT, P. GINGINS, P. KALRA, M. THALMANN. *A Biomechanical Musculoskeletal Model of Human Upper Limb for Dinamic Simulation*. Proc. 7th Eurographics International Workshop on Computer Animation and Simulation. 121-136. 31 Aug - 1 Sept. Poitiers, France. 1996.

[18] W. MAUREL, Y. WU, M. THALMANN, D. THALMANN. *Biomechanical Models for Soft Tissues Simulation*. 173pp., Springer-Verlag Berlin/Heidelberg. 1998.

[19] G. MONTILLA, A. BOSNJAK, H. VILLEGAS. *Visualización de Mundos Virtuales en la Medicina*. Reporte Interno. Centro de Procesamiento de Imágenes, Universidad de Carabobo. 2002.

[20] S. NAKAMURA. *Métodos Numéricos Aplicados con Software*. Pearson. pp.155-156. 1992.

[21] E. OÑATE, F. ZARATE. *Introducción al Método de los Elementos Finitos*. XI Curso de Máster en Métodos Numéricos para Cálculo y Diseño en Ingeniería. Universidad Politécnica de Catalunya. 2000.

[22] F. PARKE. *Parameterized Models for Facial Animation*. IEEE Computer Graphics and Aplications. Num. 9 Vol. 2 . 1982.

[23] G. PICINBONO, H. DELINGETTE, N. AYACHE. *Non-Linear Anisotropic Elasticity for Real-Time Surgery Simulation*. INRIA Report. *N°*4028. Octubre 2000.

[24] R. PRESSMAN. *Ingeniería del software: Un enfoque práctico*. McGraw Hill. 1997.

[25] E. POPOV. *Introduction to Mechanics of Solids*. Prentice-Hall. 1968.

[26] J. REDDY. *An Introduction to the Finite Element Method*. McGraw-Hill. Segunda Edición. 1994.

[27] R. SPIEGEL. *Ecuaciones Diferenciales Aplicadas*. Prentice-Hall. Tercera Edición. 1983.

[28] J. STEWART. *Cálculo - Conceptos y Contextos* . International Thomson Editores. pp.171-172. 1999.

[29] S.P. TIMOSHENKO, J. GEVE. *Mecánica de Materiales*. Centro Regional de Ayuda Técnica. 1972.

[30] Z. WESOLOWSKI. *NonLinear Dinamics of Elastic Bodies*. Springer Wien. 1978.

[31] X. WU, M. DOWNES, T. GOKTEKIN, F. TENDICK. *Adaptative Nonlinear Finite Elements for Deformable Body Simulation Using Dinamic Progressive Meshes*. EUROGRAPHICS. Vol. 20. Num. 3. 2001.

[32] Y. ZHUANG. *Real-Time Simulation of Physically Realistic Deformations*. Tesis Doctoral. Universidad de California en Berkley. 2000.