DESARROLLO DE UN INTERFAZ PARA SIMULAR EN TIEMPO REAL LA INTERACCIÓN ÓRGANO-INSTRUMENTO EN UN PROCEDIMIENTO QUIRÚRGICO DENTRO DE UN AMBIENTE DE TELEOPERACIÓN

JAIME ANDRÉS CASTILLO LEÓN

Tesis presentada como requisito parcial para obtener el título de

MAESTRÍA EN INGENIERÍA

AUTOMATIZACIÓN INDUSTRIAL

Director:
CARLOS HUMBERTO GALEANO
Profesor Titular

UNIVERSIDAD NACIONAL DE COLOMBIA
FACULTAD DE INGENIERÍA
BOGOTÁ
2011

Aprobada por la Facultad de Ingeniería en cumplimiento de los requisitos exigidos para otorgar el título de: **Maestría en Ingeniería — Automatización Industrial**

Carlos Humberto Galeano
Director de la Tesis

Ricardo Emiro Ramirez Heredia
Jurado

Jorge Iván Sofrony Esmeral
Jurado

Universidad Nacional de Colombia
Bogotá, Junio de 2011

RESUMEN

DESARROLLO DE UN INTERFAZ PARA SIMULAR EN TIEMPO REAL LA INTERACCIÓN ÓRGANO-INSTRUMENTO EN UN PROCEDIMIENTO QUIRÚRGICO DENTRO DE UN AMBIENTE DE TELEOPERACIÓN

por

JAIME ANDRÉS CASTILLO LEÓN
Maestría en Ingeniería en Automatización Industrial

UNIVERSIDAD NACIONAL DE COLOMBIA

Director: Carlos Humberto Galeano

El documento a continuación describe el camino llevado a cabo para el modelamiento y la simulación de tejidos en tiempo-real utilizando los métodos númericos de BEM y FEM para modelos tanto lineales como no-lineales en geometría y en constitución del material, además se trata el tema de la colisión y la generación de condiciones de contorno. Una aplicación multiplataforma y sus detalles de implementación en C++, bajo el paradigma de la programación genérica, los resultados obtenidos y detalles técnicos, en adición con una comparación de implementación en MATLAB.

El primer capítulo coloca la tesis como continuación de otras dos tesis llevadas en la Universidad Nacional de Colombia en el área de la teleoperación y la robótica asistencial quirúrgica, aunque no se prentende integrar con los trabajos anteriores, si se pretende como profundización de sus concluciones y trabajos futuros, en cuanto al modelamiento adecuado de tejidos. Finalmente se definen alcances, limitaciones y requerimientos de la interfaz gráfica a diseñar en el documento.

El segundo capítulo describe brevemente el estado de arte de la simulación de tejidos en tiempo-real, la detección de colisiones, el uso de tarjetas gráficas para mejorar el rendimiento de los algoritmos y otros métodos y técnicas utlizadas para lograr los requerimientos de tiempo real.

El tercer capítulo presenta la geometría computacional como piedra angular de la robustez de los algorírtmos que resuelven los problemas que se presentan en la manipulación de mallas, como lo son las colisiones y la generación de las condiciones de contorno de las PDEs al ser resueltas en tiempo real. A su vez se presenta la libreria CGAL, sus potentes capacidades y la necesidad de la programación generica como herramienta de optimización y generalización del código escrito.

El cuarto capítulo presenta la formulación de la ecuación de elasticidad teniendo en cuenta las no-linealidades geométricas y no-linealidades de material, las restriciones del modelo y todo lo que se tiene en cuenta partiendo desde la mecánica de medio continuo hasta los modelos de hiperelasticidad. En este capitulo se encuentras las ecuaciones a resolver por los métodos numéricos de BEM y FEM.

El capítulo cinco realiza una comparación de los métodos de BEM y FEM, incluyendo la formulación de los elementos BEM con su respectiva implementación en MATLAB bajo modelamiento lineal y finalmente obeservando la limitación presente en el método de los BEM para la aplicación no-lineal a causa de su compleja formulación y dejando como única opción a el método de los FEM.

El capítulo seis muestra la formulación y algunos detalles de implementación y solución en MATLAB de los elementos finitos para la solución de la elasticidad con no-linealidades.

El séptimo capítulo muestra screenshots de la aplicación desarrollada en C++, multiplataforma probada para GNU/Linux y Windows y acoplada a dos joystics de video juegos con realimentaci'on de fuerzas y los resultados obtenidos en tiempo de ejecución. Se describe el algortimo principal de la aplicación.

Finalmente se enumeran las conclusiones, logros y trabajos futuros, se anexan los detalles de implementación y menejo de programación de gráficos que se consideran importantes para futuros trabajos.

ABSTRACT

DEVELOPMENT OF AN INTERFACE TO SIMULATE IN REAL-TIME THE TISSUE-TOOL INTERACTION UNDER A QUIRURGICAL PROCESS INSIDE A TELEOPERATION ENVIROMENT

por

JAIME ANDRÉS CASTILLO LEÓN
Maestría en Ingeniería en Automatización Industrial

UNIVERSIDAD NACIONAL DE COLOMBIA

Advisor: Carlos Humberto Galeano

The following document describes a way for modelling and simuleting tissues under real-time using the FEM and BEM numerical methods for linear and non-linear behaviors on geometry and constitutive relations of material, additionally collision detection and boundary condition generation subjects are treated. A multi-plataform application and its implementation details with a benchmark with a code in MATLAB.

First chapter, puts the thesis as a continuation of other works made at Universidad Nacional de Colombia in teleoperation and assitant quirurgical robotic, but it is not expected to merge works, the objective is to deepen conclusions and future works of these on right tessue models. Finally it defines scopes, limitations and requirements on the design of the graphical interface.

The Second chapter describes the state of art on, tissue simulations under real-time, collision detection, and using graphical cards in orden to improve performances of algorithms and other methods and techniques to get the real-time requirements.

Third chapter presents the computational geometry as an angular stone for geometric robustness to solve appeared preblems in the mesh handling, such as collision detection and boundary conditions is, to solve in real-time. CGAL library is shown with its capabilities and generic programming paradigm for optimization and generalization of the written code.

Fourth chapter contains the formulation of the elasticity with nonlinear geometric and nonlinear material equations, restrictions on model and all topics, starting from continumm mechanics to hyperelasticity models. This chapter has the equations to be solved for FEM and BEM methods.

Fifth chapter carries out a comparison between BEM and FEM methods, includes formulation of bounedary elements with the implementation on MATLAB under linear models, and finally observed limitations in BEM method for nonlinear applications due to the complex formulations of BEM, left FEM method as the only one choice.

Finite elements in nonlinear elasticity solution in MATLAB, implementation details and formulations are shown in Chapter six.

Seventh chapter shows some screenshots tested in the application development in C++, proved under GNU/Linux and Windows, moreover with to joysticks with force-feedback. Some results in real-time execu-

tion. Describe the principal algorithm of the application.

Finally some conclusions, achievements and future works are listed and described. Anexed details of implementations and graphics programming considered important for future works.

RECONOCIMIENTOS

El autor desea expresar su reconocimiento a:

- A mi director por permanecer hasta el final, aun con la intermitencia de mi trabajo.
- A mi profesor Ricardo por darme la oportunidad de dictar Robótica, experiencia enriquecedora a lo largo de la maestría.
- A mis compañieros de maestría: Mauricio, Carol y Daniel quienes ayudaron de una u otra forma en este trabajo.

DEDICATORIA

A Luchis

A mis viejos

Contents

1 Introducción **1**
- 1.1 Motivación de la tesis . . . 1
- 1.2 Temas afines . . . 3
- 1.3 Alcance y requerimientos del simulador . . . 4
  - 1.3.1 Visualización de la escena . . . 4
  - 1.3.2 Interacción de la Herramienta . . . 4
  - 1.3.3 Generación de fuerzas de contacto . . . 5

2 Estado del arte **7**
- 2.1 Antecedentes de modelamiento de tejidos . . . 7
- 2.2 Modelos Computacionales . . . 7
  - 2.2.1 Modelo Masa-Resorte MSN . . . 8
  - 2.2.2 Metéodo Teoría de Elasticidad **ETM** . . . 8
  - 2.2.3 Método de los Elementos Finitos **FEM** . . . 9
  - 2.2.4 Modelo de Masa-Tesor **TMM** . . . 10
  - 2.2.5 Modelos de Elasticidad Hibridos **HEM** . . . 10
  - 2.2.6 Método de las Esferas Finitas . . . 11
  - 2.2.7 Método de los elementos de contorno **BEM** . . . 11
  - 2.2.8 Método de los elementos largos **LEM** . . . 12
  - 2.2.9 Método de la distribución de volúmenes **VDM** . . . 12
- 2.3 Métodos de resolución . . . 12
  - 2.3.1 Modelos Estáticos . . . 12
  - 2.3.2 Modelos Dinámicos . . . 13
- 2.4 Detección de Colisiones . . . 13
  - 2.4.1 Modelos de Colisiones . . . 14
- 2.5 GPUs en la solución de elasticidad en tiempo real . . . 14

3 Geometría Computacional: Interacción con mallas **15**
- 3.1 Algoritmo Geométrico . . . 15
  - 3.1.1 Predicados geométricos . . . 16
  - 3.1.2 Constructores geométricos . . . 17
- 3.2 Aritmética de punto flotante . . . 18
- 3.3 Computación Geométrica Exacta (ECG) . . . 20
- 3.4 Estructuras de información topológica . . . 21
- 3.5 CGAL . . . 23
  - 3.5.1 Estructura de la librería . . . 24
  - 3.5.2 Algoritmos utilizados en el simulador de tejido . . . 24

4 Modelamiento de tejidos **27**
4.1 Mecanica del medio continuo . . . 27
4.1.1 Relaciones Cinemáticas . . . 27
4.1.2 Ecuaciones de movimiento y relaciones de equilibrio . . . 30
4.1.3 Relación Constitutiva . . . 32
4.1.4 Material hiperelástico . . . 34

5 Implementación FEM y BEM de la elasticidad lineal **37**
5.1 Introducción . . . 37
5.2 Ecuación de Elasticidad . . . 37
5.2.1 Ecuación de equilibrio . . . 38
5.2.2 Ecuación de Constitución . . . 38
5.2.3 Ecuación Cinemática . . . 39
5.2.4 Ecuación Diferencial de la Elasticidad Lineal . . . 39
5.3 Solución de PDEs sujetas a BVCs . . . 39
5.4 Elementos Finitos . . . 39
5.4.1 Elemento Lineal Triangular . . . 40
5.4.2 Solución y Posproceso . . . 41
5.4.3 Implementación en MATLAB . . . 41
5.4.4 Solución del problema . . . 43
5.5 Boundary Elements . . . 43
5.5.1 Solución Fundamental . . . 46
5.5.2 Kernels de desplazamiento y tracción . . . 47
5.5.3 Formulación del elemento . . . 48
5.5.4 Elemento Constante de Contorno . . . 48
5.5.5 Implementación en MATLAB . . . 49
5.6 Comparación entre BEM y FEM . . . 51

6 Implementación FEM de la elasticidad no-lineal **53**
6.1 Formulación del elemento finito . . . 53
6.2 Comparación entre deformación lineal y deformación no-lineal . . . 57

7 Descripción de la aplicación **59**
7.1 Modelo tridimensional del órgano o tejido . . . 59
7.2 Enmallado del modelo del órgano . . . 60
7.3 Posición Herramienta . . . 61
7.4 Detección de colisión . . . 61
7.5 Generación de las condiciones de contorno . . . 62
7.6 Resolución de la elasticidad por elementos finitos . . . 62
7.7 Generación de fuerzas en la herramienta . . . 63

8 Conclusiones, logros y trabajos futuros **65**
8.1 Conclusiones . . . 65
8.2 Logros . . . 66
8.3 Trabajos Futuros . . . 66

A VISUALIZACION DE LA ESCENA **69**
A.1 Creación de una escena . . . 69
A.1.1 Viewers . . . 69
A.1.2 Scenes . . . 71
A.1.3 Items . . . 72

List of Tables

List of Figures

1.1 Interfaces Háticas [32] . . . 2
1.2 Plataforma Robótica [36] . . . 2

2.1 Modelado de las deformaciones de una pierna con MSN . . . 8
2.2 Aplicación de deformaciones en tiempo real con ETM . . . 9
2.3 Simulación en tiempo real de un hígado utilzando FEM. . . . 10
2.4 Simulación de un órgano con modelos lineal y no lineal utilizando TMM . . . 10
2.5 Simulación del corte de un hígado utilizando HEM . . . 11
2.6 Características principales del FSM, el mallado y sus funciones de forma . . . 11
2.7 Aplicación de deformaciones lineales . . . 12

3.1 Objetos geométricos: a) primitivas geométricas de un kernel 2-D. b) primitivas geométricas de un kernel 3-D. . . . 16
3.2 Predicado de orientación de tres puntos ordenados . . . 17
3.3 Predicado incircle . . . 17
3.4 Intersección de dos líneas . . . 18
3.5 Orientación con float y extended double[24] . . . 19
3.6 Falla en la intersección a causa de la aritmética . . . 19
3.7 Predicado de orientación 2D sobre tres líneas[40] . . . 20
3.8 estructura Half-edge . . . 22
3.9 Estructura para triangulaciones . . . 22
3.10 k-nearest . . . 23
3.11 kd-tree . . . 23
3.12 gerarquia volumenes de contorno . . . 24
3.13 Arquitectura de CGAL . . . 25

4.1 Configuracion Indeformada y Deformada . . . 28
4.2 Efecto del gradiente de deformación . . . 29

5.1 Tensor de esfuerzo . . . 38
5.2 Idealización de un esfuerzo plano . . . 40
5.3 Malla del sólido con elementos triangulares . . . 43
5.4 Matriz de Riguidez del Sistema K . . . 44
5.5 Sólido deformado a causa de la distribución de carga . . . 45
5.6 Postproceso cálculo de esferzos . . . 45
5.7 Sólido discretizado solo en la frontera para resolver por BEM . . . 49
5.8 Representación de la integral de línea . . . 49
5.9 Comparación de resultados entre BEM y FEM . . . 51

6.1 Espacio de aproximación . . . 54

6.2 Espacio de aproximación 57

7.1 Descripción de la aplicación 60
7.2 Enmallado de un hígado, con 146 tetrahedros 61
7.3 Posibles intersecciones 62
7.4 Sólido deformado después del paso de resolución de la elasticidad 63

A.1 Scene tree viewer 69
A.2 Scene openGL viewer 70

Chapter 1

Introducción

San Fancisco de Azís: *Empieza por hacer lo necesario, luego lo que es posible, y de pronto te encontrarás haciendo lo imposible*
Jaime Andrés: *Empecé por lo imposible, luego corrí todo lo posible y no logrando lo necesario, terminé con esto. :(*

1.1 Motivación de la tesis

El propósito de este documento es investigar la posibilidad de implementar un simulador de la ecuación de elasticidad en tiempo real, para ser aplicado en ambientes de realidad aumentada con la sensación del tacto. Se conoce como *háptica* a la característica que se incluye en un entorno de realidad aumentada para darle realismo al contacto o interacción de objetos con distintas propiedades mecánicas y así poder involucrar el tacto en el ambiente simulado.

El esfuerzo de esta investigación va encaminado al desarrollo de simuladores de cirugía que facilita en muchos aspectos, el aprendizaje, la práctica, la planeación e inclusive el transcurso de una cirugía. Este estudio forma parte del complemento de otros trabajos previos desarrollados en la Universidad Nacional [32, 36]. El estado de arte de estos documentos colocan en un contexto más amplio la finalidad de este trabajo, sin embargo acá se irá directamente a la implementación del simulador y lo que se requirío para ello.

En el trabajo desarrollado por [32], se hizo una investigación exahustiva del estado del arte, de los distintos dispositivos hápticos presentes en el mercado y la Universidad adquirió por parte del departamento de Ingeniería Mecanica y Mecatrónica dos de ellos (un Phanthom OMNI™ y un Novit Falcom™). Se hicieron aplicaciones sobre Windows, utilizando librerias de programación de dispositivos hápticos como Open Haptics y otras librerias suministradas por los fabricantes con lo cual se logró el manejo del dispositivo. La parte gráfica se trabajó usando OpenGL y el manejador de ventanas y contexto de OpenGL, GLUT. Finalmente se hizo una realimentación de fuerzas del entorno virtual hacia la mano del operador del Joystic hático (ver figura 1.1).

En el trabajo de [36], se diseñó y construyó una plataforma paralela robótica, que busca replicar los movimientos de traslación y rotación de una herramienta quirúrgica en procedimientos de reconstrucción

maxilo-facial principalmente. La plataforma debe ser capaz generar fuerzas y momentos necesarios en un procedimiento, generados préviamente en una etapa de planeación en un ambiente de realidad virtual a causa de la simulación háptica (ver figura 1.2).

Revisando el estado del proyecto, los modelos de generación de fuerzas utilizados en [32], la simulación de sólidos se simplifican a lo siguiente: $\vec{F} = k\vec{x} + b\dot{\vec{x}} + m\ddot{\vec{x}}$, donde $\vec{F}$ es la fuerza que la herramienta hace sobre el sólido, $\vec{x}$ es la posición o punto de aplicación de la fuerza $\vec{F}$, k es una constante de rigidez (su valor se da de forma arbitraria), b es el coeficiente de fricción viscosa (que en definitiva no se usa) y finalmente, m es la masa (que al incluirla en el modelo hace que se pierda el sentido de la ecuación y por lo tanto tampoco se usa). El hecho de que el efecto viscoso b y la inercia m no se usen es por que el sólido que se analiza es el de un cuerpo deformable empotrado en alguna parte de su contorno, por lo tanto la inclusión de estos dos efectos requiere de un modelo más adecuado.

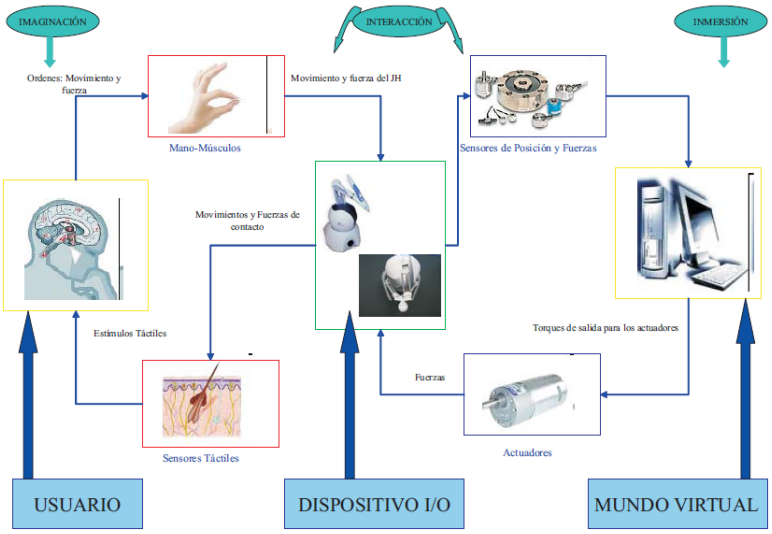


Figure 1.1: Interfaces Háticas [32]

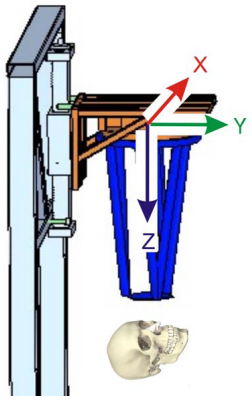


Figure 1.2: Plataforma Robótica [36]

En conclusión es necesario un modelo que se acerque más a la realidad para poder obtener una realimentación de fuerzas dentro del sistema de realidad aumentada, en la siguiente referencia [43] se mencionan los distintos tipos de modelos que se han realizado y documentado. De momento el estudio de modelos bajo la *mecanica de medio continuo* [44], se usa como el más recomendado para lograr una aplicación adecuada de realidad aumentada. El trabajo de esta tesis busca únicamente la implementación de modelos elásticos en tiempo real mas no la integración de éste con los trabajos de [32, 36], el cual podrá ser un trabajo futuro.

1.2 Temas afines

La investigación de esta tesis descansa sobre los marcos y los avances de las siguientes ramas de estudio: la *Programación genérica*[50], el *Análisis numérico*, la *Computación gráfica*[42], la *Geometría computacional*, los *Sistemas dinámicos* y la *Mecánica Computacional*. El soporte matemático de las materias anteriormente mencionadas consta de: *teoría de números*, *algebra lineal*, *análisis funcional*, *cálculo tensorial*, *cálculo variacional* y algunas nociones básicas de *topología*. Cada una de ellas encuentra una parte importante en la investigación del presente documento.

La programación genérica en especial de C++ será clave en la implementación de los algoritmos utilizados. Este paradigima optimiza los códigos escritos en tiempo de compilación y generaliza las ideas de programación permitiendo la adaptabilidad, extensibilidad y eficiencia de librerias externas al código [18], la programación orientada a objetos y estructurada esta incluida en este paradigma. Actuálmente, los desarrolladores de software con requerimientos de eficiencia en tiempo-real tienden a utilizar C++ usando la programación genérica (STL, BOOST, MATLAB, SCILAB, BLENDER, ANSYS, OpenSCAD y otros), ha sido la principal justificacion de elegir ésta como lenguaje para la implementación de la aplicación. Otra tendencia de más es que la revista *Comutational Geometry: Theory and Applications* exige sus documentos en C++ usando los beneficios de las *templates*.

El análisis númerico será importante tanto en la validación y convergencia de los métodos de los elementos finitos **FEM** y los elementos de contorno **BEM**, así como también en la solución de estos métodos donde se destaca la utilización de métodos iterativos sobre matrices densas y esparcidas, y los método de Newton-Rhapson, Runge-Kutta y Euler.

El lenguaje de representacion visual será expresado por la computación gráfica en dos nivel: técnico[42], como usuario de las tarjetas de video para renderizar e investigativo, para realizar cálculos en paralelo sobre la GPU[39]. Este tema será relevante en la representación de los tejidos manipulados en pantalla asi como en el posible aceleramiento de los cálculos usando la GPU[30].

La geometría computacional[5] es de vital importancia para la manipulación de las estructuras que discretizan y representan los sólidos a modelar, como tejidos y herramientas. Por ello será tambien importante en el tema de las colisiones para el contacto y el establecimiento de las condiciones iniciales de contorno para resolver las ecuaciones diferenciales parciales. Además dentro de este tema se encuentran pre-procesos requeridos tales como, el mallado volumétrico y el mallado superficial. Las librerias diseñadas para estos fines son tema actual de investigación en áreas como: la robustez en la mecánica computacional, las ciencias de la computación, la implementación sobre la programación génerica y la metaprogramación. Temas que hasta ahora solo el lenguaje C++ tiene como ventaja al momento del desempeño en tiempo de ejecución.

El modelo del tejido como sistema dinámico[4] es puesto en consideración para analizar sus respuestas en frecuencia y bajo cargas quasi-estaticas para acotar el modelo. Luego se supondrá que los procedimientos quirúrgicos se harán sobre un comportamiento quasi-estatico de las cargas debido a su naturaleza.

La mecánica computacional[8, 44, 4] aplica la teoría mecánica de medio continuo, que forma la mayor parte del documento en donde se encuentran los fundamentos para el modelamiento y la simulación correcta de tejidos. A partir de estas teorías y mediante ensayos empíricos es posible determinar una serie de constantes que representan las propiedades mecánicas de los materiales y de esta forma poder modelar adecuadamente el tejido a trabajar. Los métodos utilizados acá son **FEM** y **BEM** y no se realizan ensayos empíricos.

1.3 Alcance y requerimientos del simulador

Los principales objetivos o funcionesque en el simulador puede realizar son:

1. Visualizar los objetos
2. Permitir manejo interactivo de usuario-herramienta
3. Generar las fuerzas de contacto debidas a la interacción.

1.3.1 Visualización de la escena

El proceso de visualización se debe realizar con una tasa de refresco superior a los 25Hz para que el ojo humano sienta la continuidad en los movimientos, sin embargo para que no capte una sensación de parpadeo [10, 28, 32] la tasa de renderización gráfica debe ser superior a 60Hz. En esta parte de la investigación fue necesario el uso de las librerias de OpenGL para renderizar las escenas necesarias de forma adecuada.

OpenGL[19] es una libreria multiplataforma escrita en C, que deja a disposición del programador las funcionalidades de la tarjeta de video mediante el manejo de buffers, variables de estado y enumeraciones. Algunos de los buffers más importantes son el buffer de renderizado, el buffer de selección y el buffer de feedback. Los dos primeros buffers serán fundamentales para la interacción con el usuario.

Como el manejo de la tarjeta gráfica es independiente del sistema de ventanas que se utilice en el sistema operativo, tambien se escogió las librerias de Qt[29] para el manejo de interfaces grafícas de usuario GUIs, ya que esta libreria Qt es a su vez multiplataforma (es decir independientes del sistema operativo). La integración de un manejador de ventanas y OpenGL requiere de la creación de un contexto para el uso de OpenGL. El contexto para OpenGL ya está creado para el widget QGLWidget, sin embargo se utiliza una extención de esta llamada libQGLViewer [14] que simplifica notoriamente este proceso bastante oscuro para principiantes en programación de gráficos.

1.3.2 Interacción de la Herramienta

El usuario podrá mover una herramienta en la escena mediante el manejo de los eventos implementados en Qt, estos eventos permiten adicionar dispositivos HID, tales como joystics hápticos, aunque estos estarán usando su propio hilo a través de una instancia de QThread.

Para este estudio se utilizó: un joystic de videojuegos para simuladores de avión con force-feedback Logitech 3D-FORCE, el joystick de la consola de videojuegos PS2, el teclado y el mouse del computador, es posible adaptar cualquier tipo de joystick simplemente implementado una clase que hereda una interfaz.

Además de estos detalles técnicos, hablando de la interacción de dos sólidos, cabe mencionar que estos sólidos son representados por poliedros mediante el uso de mallas de polgonos (triángulos generalmente). Las

mallas por lo general son triangulaciones y el manejo de estas requiere de estructuras de datos adecuadas para expresarlas y algoritmos para su manipulación. Las estructuras deben facilitar las búsquedas sobre la malla y adicional a esto ser livianas en su consumo de memoria. Debido a estas caracteristica y otras mencionadas más adelante, se utiliza la libreria de C++, CGAL: *Computational Geometric Algorithms Library* [48], la cual tiene como paradigma la programación generica, paradigma altamente utilizado por desarrolladores de librerias altamente reutilizables y eficientes (como las STL de C++ o las librerias Boost).

1.3.3 Generación de fuerzas de contacto

Para la generación de fuerzas los modelos que se tendrán en cuenta serán los propuestos en la teoría de la elasticidad. En esta teoría se plantean ecuaciones diferenciales parciales no lineales, las cuales por lo general se resulven empleando algun método numérico como los elementos de contorno *BEM* o los elementos finitos *FEM* [3, 26, 25, 52]. Aquí la generación de mallado volumétrico y mallado superficial aparece como una importante decisión en el diseño de la interfaz, la colocación de las condiciones de contorno de las ecuaciones es de vital importancia por esto *CGAL* se ha utlizado, ya que el diseño de las estructuras que manejan las mallas optimiza este proceso, que facilitan en tiempo real la colocación de condiciones de contorno probocadas por el moviemiento de la herramienta.

El uso de un solucionador de métodos iterativos es necesario para la solución de sistemas de la forma $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$. Existen una gran variedad de *solvers*, para este trabajo se probó con las librerias para matrices esparcidas SMLIB escrita para FORTRAN 90 y para matrices densas con *Linear Algebra PACKage* LAPACK escrita inicialemente en FORTRAN 77. LAPACK es fácilmente integrable con C++ y de hecho hoy en día los programas de elementos finitos y lenguajes como MATLAB y SCILAB las utilizan para realizar operaciones de matrices y vectores densos por sus altos rendimientos, las GPUs sacan versiones nuevas de LAPACK como es el caso de CUDA. El problema con las librerias de métodos iterativos y las matrices esparcidas ya no se deja a disposición de las librerias hechas en FORTRAN si no que se utlizan structuras en C/C++ para resolver estos sistemas. Los solvers de C++ utilizados fueron ITL++, SparseLib++, GMM del paquete de elementos finitos GetFEM, estas tres últimas bajo el paradigma de la programación genérica. Además de estos solvers se utilizaron los métodos iterativos implementados en MATLAB, como se describirá más adelante.

Los requisitos de los tiempo para calcular las fuerzas de contacto dependen de que sensación se debe identificar en el simulador, es diferente percibir rugosidades a sentir que se estrella una herramienta con un objeto elástico o rígido. Las rugosidades requieren de una tasa igual o mayor a 1000Hz [10, 51], pero sentir si un objeto es duro o blando no requiere de tan alta realimentación. Esta frecuencia es algo a investigar en este documento.

El uso de la GPU para incrementar el proceso de solución de estos sistemas resultantes en los métodos númericos son un punto de investigación, en donde solo se utilizó la manipulación de matrices esparcidas[30]. Se utiliza el SDK de nVidia, CUDA, en donde se utlizan dos tarjetas una de 12 nucleos y otra de 96 nucleos.

Chapter 2

Estado del arte

Neo: *This isn't real?*
Morpheus: *What is real? How do you define real? If you are talking about your senses, what you feel, taste, smell, or see, then all you're talking about are electrical signals interpreted by your brain.*

2.1 Antecedentes de modelamiento de tejidos

La simulación de tejidos ha llegado a ser el siguiente paso para la realidad aumentada **AR**, el avance en el procesamiento de imagenes, el diseño asistido por computador, el control y su aplicación a entornos con características hápticas ponen como necesidad la simulación de los tejidos para alcanzar un entorno más veráz y con aplicaciones en la teleoperación y otras áreas nuevas más.

A continuación se hace una breve descripción no exhaustiva de los método más tradicionales y recientemente utilizados para alcanzar la simulación de tejidos sobre tiempo real. Los puntos y objetivos a destacar de los métodos son los siguientes,

1. Realidad en los modelos físicos, es decir basados en principios físicos que describan correctamente el comportamiento de los tejidos.
2. Velocidades de $1kHz$ en la solución del modelo mecánico de los tejidos, para engañar el sentido del tacto del operador del sistema.
3. Velocidad de $25Hz$ en el renderizado para engañar el sentido de la vista del operador de sistema.
4. Detección de colisiones que establezca de forma fácil las condiciones de contorno o borde.

2.2 Modelos Computacionales

Algunos de los modelos computacionales actualmente estudiados en la simulación de tejidos en tiempo real son resumidos destacando algunas de sus ventajas y desventajas. Algunos de ellos se encuentran desarrollados dentro de marcos físicos adecuados, otros tienen un soporte matemático suficientemente robusto como para generar modelos mas no un soporte físico como tal.

2.2.1 Modelo Masa-Resorte MSN

En este modelo las deformaciones son llevadas sobre una malla de resortes [47], en donde cada una de las esquinas o terminales del resorte se encuentra sujeta a un nodo con masa puntual (ver Fig 2.2.1).

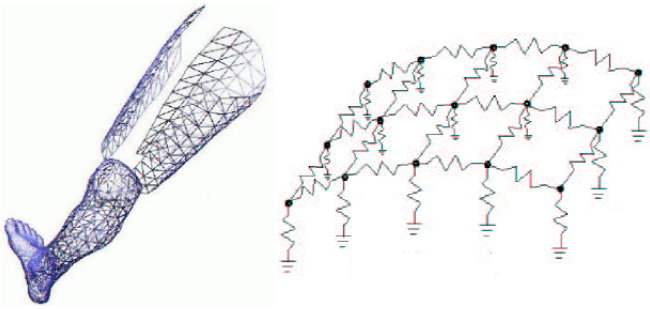


Figure 2.1: Modelado de las deformaciones de una pierna con MSN

Por lo general la misma malla que representa el gráfico es la misma que coincide con los nodos de la **MSN**, de esta forma queda discretizado el movimiento del sólido a una serie de nodos y resortes. Este método es bastante utilizado en la industria del cine par crear animaciones faciales.

Como ventajas este método presenta una dinámica muy bien conocida, como lo es una malla de resortes, con una sólida matemática para su modelamiento, además de esto se puede considerar algunos de los resortes en comportamiento no lineal. Los modelos constitutivos pueden contener relaciones tanto de rigidez como de viscoelasticidad.

El modelo dinámico es representado de la forma

$$m_i \ddot{u}_i + c_i \dot{u}_i + g_i = f_i \tag{2.1}$$

donde u_i es el despalzamiento de cada nodo, c_i es el coeficiente de amortiguamiento, m_i es la masa puntual asociada al nodo i, g_i son las fuerzas intríncecas que se generan en el material y f_i es la fuerza externa que actua sobre ese nodo. El modelo se acopla con f_i de la siguiente forma,

$$f_i = \left(\lambda d - \mu \dot{d}\right) \hat{k} \tag{2.2}$$

donde λ es la rigidez del resorte y ν es el coeficiente de viscoelasticidad.

El costo computacional de este método es bajo por lo que se utiliza en aplicaciones con interacciones de tiempo real. Las principales desventajas de este metódo son que son poco realistas y que a la hora de especificar las constantes del modelo no hay un forma válida de proponer los valores, este método es más bien una imposición al modelo físico.

2.2.2 Metéodo Teoría de Elasticidad ETM

Esta basado en la ecuación dinámica de la elasticidad [31],

$$\rho \ddot{u}_i = \mu \nabla^2 u_i + (\lambda + \mu) \nabla (\nabla \cdot u_i) \tag{2.3}$$

donde u_i es el deplazamiento del nodo i, μ y λ son las constantes de Lammé.

La aproximacién es basada calculando cada operador de eq.2.3, con los nodos vecinos j así,

$$\nabla^2 u_i = \frac{2}{\sum_j |L_{ij}|} \sum_j \frac{u_j - u_i}{|L_{ij}|} \tag{2.4}$$

$$\nabla (\nabla \cdot u_i) = \frac{2}{\sum_j |L_{ij}|} \sum_j \frac{(u_j - u_i) \cdot n_{ij}}{|L_{ij}|} n_{ij} \tag{2.5}$$

en donde $|L_{ij}|$ es la distancia entre los nodos j e i, n_{ij} es el vector unitario en la dirección del vector L_{ij}.

Al igual que el método MSN, ofrece estabilidad pero a diferencia de MSN el ETM ofrece más realismo. Se reportan pruebas con este método sobre tiempo real[31]. Algunas ventajas es que no requiere generación de una malla estructurada y su precisión puede mejorarse en tiempo de ejecución con refinamientos de malla locales.

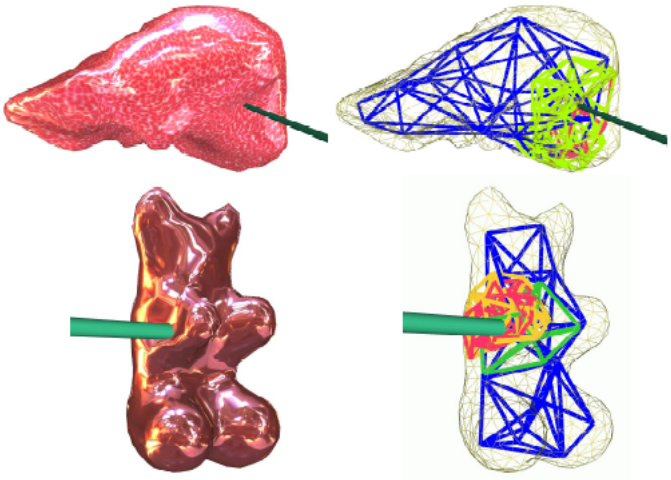


Figure 2.2: Aplicación de deformaciones en tiempo real con ETM

2.2.3 Método de los Elementos Finitos FEM

Este método es el más utilizado[4], con este método es posible implementar problemas no lineales. Para grandes deformaciones se usa el tensor de deformaciones de *Green-Lagrange* en combinación con una ya conocida relación de constitución generada a partir del segundo tensor de esfuerzo de *Piola-Kirchhoff* que no viola los principios termodinámicos del material[44].

Las principales ventajas es que es un método con gran realismo ya que este resuelve la ecuación de elasticidad de forma númerica. Como desventajas se encuentra su alto costo computacional lo que hace poco atractivo el uso de este método para ser utilizado en soluciones de tiempo real[43].

El preproceso de este método es costoso ya que se requiere de procesos de mallado en 3D, sin embargo cuando se trata de pequeñas deformaciones se puede tener precomputado el problema pero este no es el caso para la simulación de tejidos.

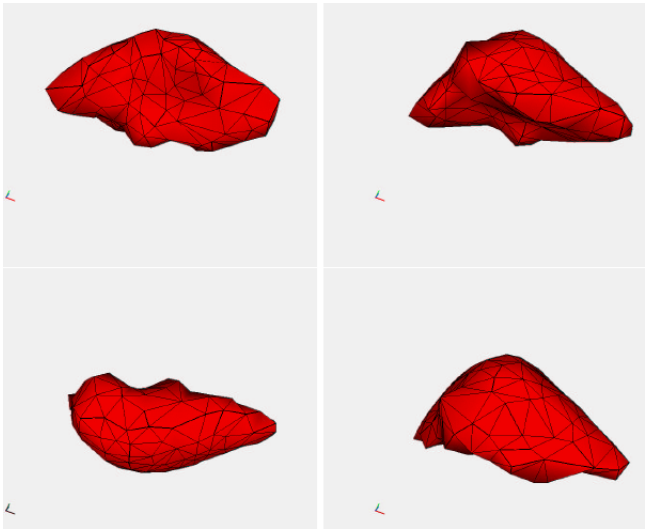


Figure 2.3: Simulación en tiempo real de un hígado utilzando FEM.

2.2.4 Modelo de Masa-Tesor TMM

Aquí la formulación es similar al método de MSN a diferencia de que, g_i que representa las fuerza internas en el módelo es tomada de la energía almacenada en el tetrahedro como si fuera un elemento finito.

Se han encontrado aplicaciones para tejidos en tiempo real con modelos no lineales anisotropicos[43] y aplicando viscoelasticidad[27].

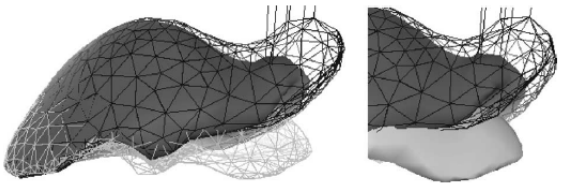


Figure 2.4: Simulación de un órgano con modelos lineal y no lineal utilizando TMM

2.2.5 Modelos de Elasticidad Hibridos HEM

Existe tambien una combinación de métodos en los cuales se parte de una solución precomputada de FEM y apartir de esta se procede a resolver el problema no lineal o con largas deformaciones utilizando el TMM [10]. Una forma interesante de abordar este reto es que se subdivide el dominio en dos, (1) en el que se asume que las deformaciones son pequeñas y no existirá un cambio topológico como tal, en este caso se utiliza una solución precomputadada a cargo de los FEM y (2) en una región (que es la de interés) en la que el comportamiento se espera sea no lineal a cargo de los TMM.

A la hora de implementar un HEM es importante que los fundamentos teóricos sean similares (por ejemplo [11]). Una combinación muy frecuente es FEM-BEM.

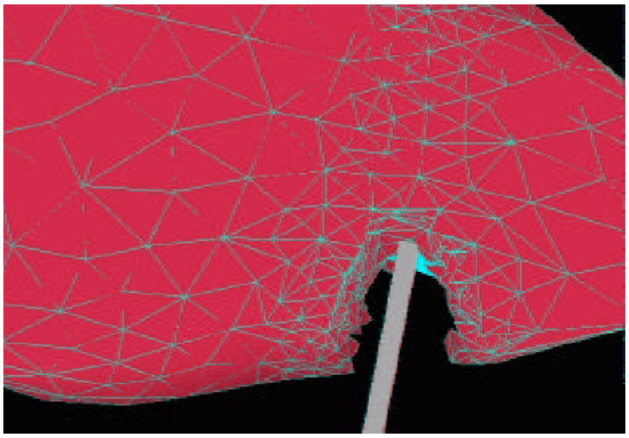


Figure 2.5: Simulación del corte de un hígado utilizando HEM

2.2.6 Método de las Esferas Finitas

Su principal ventaja es que es un método que no requiere malla, en lugar de la malla utiliza un conjunto de punto al interior o sobre el borde del volumen para resolver las ecuaciones de equilibrio [13]. A diferencia de los FEM, las funciones de forma son funciones racionales un poco más complicadas que los polinomios que generalemente se utilizan en FEM las cuales se escogen de forma que el costo computacional sea bajo. De todas formas al interior de este método surgen integrales que no tiene solución analítica y requieren de reglas numéricas.

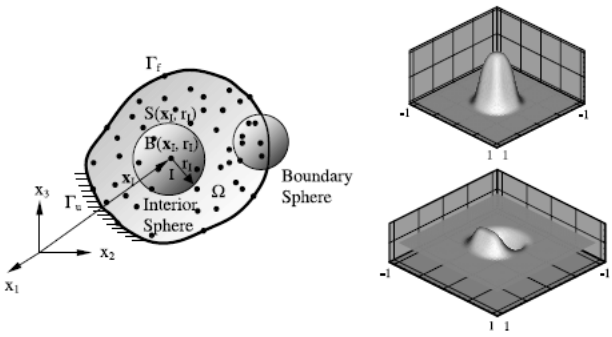


Figure 2.6: Características principales del FSM, el mallado y sus funciones de forma

Realmente el costo de las funciones de forma hace que este método sea costoso, existen técnicas que llevan este método a la implementación en tiempo real. Otra desventaja presente en estos métodos es que las condiciones de contorno son ajenas a los nodos de las funciones de forma y vuelven este proceso de condiciones iniciales no tan transparente y más bien enredado.

2.2.7 Método de los elementos de contorno BEM

En este método el problema de generar la malla se reduce a solo tener el modelado del sólido por elementos triangulares sobre su superficie[6]. Aunque se puede pensar que las matrices de rigidez del sistema serán mucho más pequeñas que las generadas en los FEM, el problema es que estas matrices son densas a diferencia

de las FEM que son esparcidas[17].

Este método se comporta bien para problemas lineales[51, 45], sin embargo para las no linealidades hay que proponer esquemas o generar kernels para la solución no lineal lo cual puede requerir de un gran manejo matemático[17].

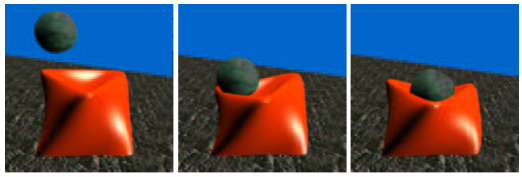


Figure 2.7: Aplicación de deformaciones lineales

Al igual que el FSM se presentan integrales númericas, pero estas ya no son tan costosas, y por el hecho de que se puede tener soluciones precomputada se puede generar aplicaciones sobre tiempo real (pero solo en modelos lineales). Gracias al planteamiento de los BEM sa ha descubierto tcnicas alternativas a los FEM que presentan ventajas en cuanto a la convergencia, los jacobianos negativos y teorías de fractura.

2.2.8 Método de los elementos largos LEM

En este método los elementos son pensados como objetos distribuidos con un fluido incompresible en su interior. Sin embargo obedecen la ley de Hooke en sentido axial. Las condiciones de frontera son una mezcla de los principios de pascal y conservación de energía. Este método permanece aun sin explorar pero existen aplicaciones para tiempo real.

2.2.9 Método de la distribución de volúmenes VDM

Este método es inspirado en las ideas del LEM, ejemplos con un hígado en el que se tiene en cuenta su composición mecánica como: (1) una superficie de piel eléstica, (2) el interior lleno de líquido (sangre) y (3) un sistema de irrigación[43].

Al igual que los BEM solo se discretiza el contorno del dominio, las fuerzas de campo producidas en el modelo son simplificadas por la acción de un fluido incompresible al interior del modelo. Para plantear las ecuaciones de equilibrio tiene en cuenta los mismos principios de los LEM.

2.3 Métodos de resolución

En esta parte se tiene encuenta dos tipos de modelos, estáticos y dinámicos.

2.3.1 Modelos Estáticos

Como principales concideraciones se desprecian los efectos inerciales y viscoelásticos. Esto se considera así cuando las cargas que actúan sobre el sistema con frecuencias menores a un tercio de la frecuencia natural

del sistema[43].

En estos casos el modelo físico es representado por una sistema de ecuaciones lineales con incógnitas,

$$K\,u\,=\,f \tag{2.6}$$

donde K es la matriz de rigidez que relaciona los desplazamientos u del sistema con las cargas f aplicadas en él.

Se puede tener dos modelos estáticos, lineales y no lineales. En los modelos lineales la solución se reduce a encontrar la inversa de K, para esto existen dos tipos de técnicas, directas e iterativos.

En las técnicas directas se tiene algoritmos que ofrecen precisión pero pueden ser costos en espacio y tiempo debido a que crean otras matrices semejantes durante el proceso y la eficiencia de estos algoritmos depende de si la matriz K es esparcida y de su tamaño en general. Algunas técnicas son descomposición LU, factorización de Cholesky, etc [4].

A costo de precisión se utilizan herramientas iterativas como Gauss-Seidel y el Gradiente Conjugado los cuales depende del punto inicial u_0.

Cuando las deformaciones y rotaciones del continuo presentan cambios mayores al cinco por ciento se tiene problemas no lineales en los cuales se tiene en cuenta los siguientes posibles cambios,

- **Material No-Lineal.** Encontrar relaciones constitutivas del material.
- **Geometría.** Cuando los cambios son grandes las relaciones desplazamiento-deformacin cambian y por lo tanto A.

El cambio más notorio con la ecuación 2.6 es que la matriz de rigidez será función del estado de desplazamientos $K = K(u)$ y por esto la ecuación 2.6 se resuelve utilizando algoritmos para encontrar raices como Newton-Rhapson[4].

2.3.2 Modelos Dinámicos

A diferencia del modelo estático en este se llega a un sistema de ecuaciónes difernciales de segundo orden de la forma,

$$M\,\ddot{u} + C\,\dot{u} + K\,u\,=\,f \tag{2.7}$$

en donde M y C son las matrices de inercias y amortiguaciones.

En el caso que $M(u)$, $C(u)$ y $K(u)$, es decir que presenta no lineadlidades, no existe una solución analítica de la ecuación 2.7 y entonces se recurre a métodos recursivos como Newton-Euler o Runge-Kutta.

2.4 Detección de Colisiones

Otra parte importante en un simulador de tejidos, es sin duda los algoritmos de detección de colisiones que se han de implementar, este tema es más de *geometría computacional* que de la mecánica computacional. Estos algoritmos deben ser robustos, rápidos y eficientes[43]. Una clasificación de estos algoritmos es según: el volumen de espacio-tiempo, barrido de volumen, chequeos multiples y parametrización de trayectorias[20]. Dependiendo de cual es la principal necesidad, robustez, rapidez o eficiencia, se escoge cual es la mejor opcin.

Otra clasificación esta dada por en el ya clasico libro de video juegos de [16] en donde hacen un detallado planteamiento y solución de problemas de colisión, donde se expone el algoritmo GJK para la solución de colisión entre politopos convexos, aunque el planteamiento de [16] para la implementación de los predicados ha sido recientemente mejorada por el uso de aritmética modular y filtros adaptativos usando una reciente rama de investigación llamada *exact computation geometry* ECG.

Los tres primeros tipos de algoritmos según la clasificación anterior son conocidos por *Pruebas de Interferencia Estticas* **SIT**.

2.4.1 Modelos de Colisiones

Generalmente la detección de colisiones esta dividida en tres fases de optimización:

1. **Fase extensa:** Se compara por pares de objetos si existe una posible colisión. Por lo general estos métodos utilizan la *descomposicin espacial*.
2. **Fase de reducción:** De esta no se obtiene la colisión exacta pero se reduce el volumen a una región mínima en la cual se puede encontrar la colisión. Los algoritmos de esta fase utilizan la *descomposicin de objetos*.
3. **Fase exacta** Encuntran la intersección exacta entre los politopos.

Existen librerias como las V-COLLIDE, H-COLLIDE y RAPID que aplican estos métodos, pero tiene el inconveniente de no estructurar la información y solo trabajar sobre sopas de triángulos, ademas los algoritmos externos deben adaptarse a las librerias. CGAL trabaja con árboles kd-tree para los cuales se organiza en un preproceso una estructura que representa un poliedro o una sopa de triángulos.

2.5 GPUs en la solución de elasticidad en tiempo real

Los recientes avances y la exponencialmente creciente capacidad de computo de las tarjetas gráficas respecto a las CPUs ha llevado a que muchos algoritmos que se diseñaban sobre arquitecturas seriales sean ahora acondicionadas a la programación en paralelo con un bajo costo. Las tarjetas nVIDIA mantiene una SDK gratuita llamada CUDA para usar sus tarjetas en aplicaciones GP-GPU *General Purpose Graphical Processor Unit*.

La programación en paralelo enfrenta nuevos retos, tanto técnicos como de análisis en el rediseño. Los retos técnicos implican la constante actualizacion de la SDK al comienzo de este documento la version 2.0 estaba reciente y en linux era un poco inestable y dificil de poner a funcionar, al final la ultima versión de CUDA con que se trabajo fue la versión 3.2 donde su instalación ya era sencilla y transparente pero su mantenimiento era un poco fastidioso. Al 2011 la versón de CUDA 4 salio con notorias mejoras e incorporaciones en C++.

En el rediseño aplicado a las ecuaciones de elasticidad muestran unos avances y aplicaciones hacia el reto de tiempo real, como lo es [46] en donde se propone una eficiente solución de la visco-elasticidad sobre materiales anisotrópicos usando CUDA, en [15] se propone un esquema basado en la conectividad de la malla tambien usando la GPU bajo CUDA y la referencia [12] propone un esquema de corte de tejido en tiempo-real usando la GPU.

Chapter 3

Geometría Computacional: Interacción con mallas

El error no es lógico, es aritmético!!!

Los números reales no existen.
— El Computador

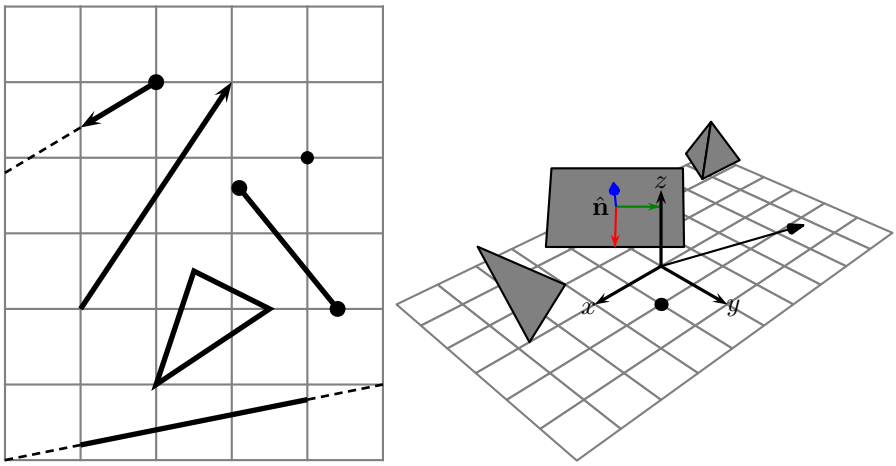
Es extremadamente tentador generar algoritmos geométricos desde cero, sin embargo poco después de empezar ese camino saldrán a flote problemas con los algoritmos y pese a que su lógica sea correcta su comportamiento es inesperado o simplemente nunca termina. La razón fundamental del problema es que la lógica del algoritmo esta escrita pensando en el espacio Euclidiano $\mathcal{E}^2$ o $\mathcal{E}^3$ que está estructurada sobre el cuerpo de los reales $\mathbb{R}$, pero la implementación esta hecha con un subconjuto de $\mathbb{R}$, llamado *números de punto flotante*.

Este capítulo trata de la importancia que tienen los algoritmos geométicos y su manejo adecuado para la robustez de los algoritmos en general, las aplicaciones y programas finales. El mallado y la partición de espacios para búsquedas efectivas recaen en su totalidad sobre la robustez de estos algoritmos geométricos. Sus aplicaciones son muy variadas pero para mencionar algunas se encuentran: los preprocesos de mallado en la mécanica computacional, las particiones de espacio y árboles de detección de colisión, los mapas de Voronoi para aplicaciones de planeación de trayectorias en robótica.

3.1 Algoritmo Geométrico

Un algoritmo geométrico define una serie de operaciones y procesos llevados acabo sobre objetos geométricos. El tipo de geometría que interesa por ahora es la geometría euclidiana en donde los principales objetos que la conforman son: puntos, líneas, vectores, rayos, segmentos, planos, triángulos, polígonos, poliedros, etc,. (ver figura fig.3.1).

Así como un algoritmo puramente aritmético sobre $\mathbb{R}$, solo descansa sobre su estructura algebraica $(\mathbb{R}, +, \cdot, \leq)$, es decir usando los siguientes conjutos de operaciones: $\{+, -, /, \cdot\}$ y $\{<, \leq, =, \geq, >\}$, en donde el primer conjunto, *operadores de cuerpo o campo* son relaciones binarias de la forma $\mathcal{O}_1 : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ y el segundo conjuto, *operadores de orden*, tiene la forma $\mathcal{O}_2 : \mathbb{R} \times \mathbb{R} \mapsto \{\text{falso,verdadero}\}$. Lo mismo ocurrirá para un algoritmo geométrico en donde existen dos tipos de operadores: *predicados* y *constructores*.



a) b)

Figure 3.1: Objetos geométricos: a) primitivas geométricas de un kernel 2-D. b) primitivas geométricas de un kernel 3-D.

3.1.1 Predicados geométricos

En cuanto a los predicados, estos operadores revelan información acerca del orden, relación y combinatoria de los objetos en cuestión, al igual que los operadores de orden el resultado de esta operación es un elemento de una enumeración. Un ejemplo, son los malladores basados en las triangulaciones de Delaunay [41], para el cual los principales predicados buscan: 1) una misma orientación en cada triángulo y además 2) cada triángulo de la triangulación no puede tener ningún vértice de la triangulación en su interior.

El predicado de *orientación de un triángulo* se basa en el orden de los vértices que conforman el triángulo, puede dar las posibilidades de la figura3.2, pero dicho orden se puede dar de diferentes formas y tener la misma orientación, por esto surge en los predicados una componente de combinatoria de los vértices, esta componente combinatoria se puede complicar en objetos compuestos más complejos, y abordar las enumeraciones de los predicados de forma exhaustiva de todos los casos no es fácil, esto es lo que ocurre en algoritmos como *la distancia con signo a un poliedro no-convexo.*

El segundo predicado y el más importante en una triangulación de Delaunay es el predicado *incircle* como lo muestra la figura3.3, el predicado *(*incircle) es implementado sobre el signo del siguiente determinante (ecuación 3.1), siempre y cuando p, q y r conserven siempre la misma orientacion (por ejemplo giro a la izquierda). El punto s en color rojo en la figura 3.3, dará positivo si se encuentra en el interior, cero si esta sobre el borde y negativo si está por fuera.

$$\begin{vmatrix} p_x & p_y & p_x^2+p_y^2 & 1 \\ q_x & q_y & q_x^2+q_y^2 & 1 \\ r_x & r_y & r_x^2+r_y^2 & 1 \\ s_x & s_y & s_x^2+s_y^2 & 1 \end{vmatrix} \tag{3.1}$$

Aunque parezca extraño, el determinante de 4 por 4 de la equación 3.1, que se puede reducir a un determinante de 3 por 3, no es tan fácil de implementar de forma robusta. De hecho en códigos de malladores, en los

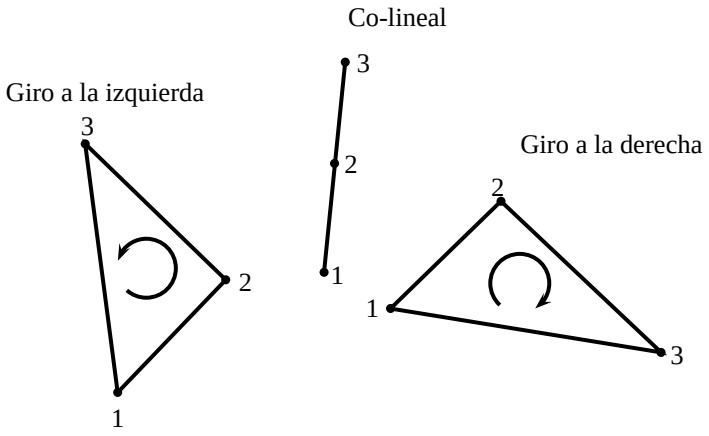


Figure 3.2: Predicado de orientación de tres puntos ordenados

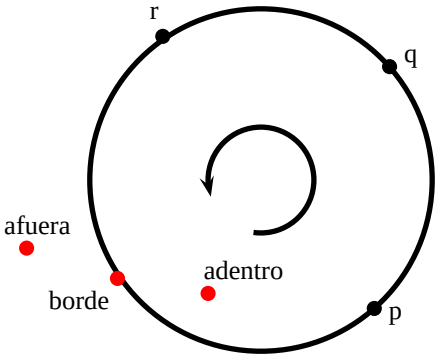


Figure 3.3: Predicado incircle

cuales este predicado solo representa una línea en el algoritmo principal detrás de ellos se esconden el ochenta por ciento de la totalidad de las líneas de todo el mallador, como se puede ver en le mallador Triangle-2D[41].

3.1.2 Constructores geométricos

Como su nombre lo indica estas operaciones tiene como resultado objetos geométricos o un conjuto de estos. El caso más común son las intersecciones de objetos, por ejemplo, la intersección de dos líneas (ver figura 3.4), puede regresar un punto, una línea o nada, por esto el operador puede construir un objeto. Sin embargo este contructor geométrico puede tener una versión tipo predicado en la cual solo se pregunta el tipo de intersección, pero no se requiere el objeto construido.

Por lo general los predicados dependen del signo de un determinante o del comportamiento de un discriminante, sin embargo los constructores requieren además de evaluar predicados, operaciones adicionales que permiten encontrar los objetos geométricos que resuelven la construcción. Estos constructores tiene un mayor reto para la implementación y las últimas tendencias es clasificar los algoritmos en unos que dependen de los predicados y/o de los constructores.

Otra inquietud surge al representar de forma exacta los objetos construidos, ya que este problema requiere de aritmética exacta o una forma de representación exacta.

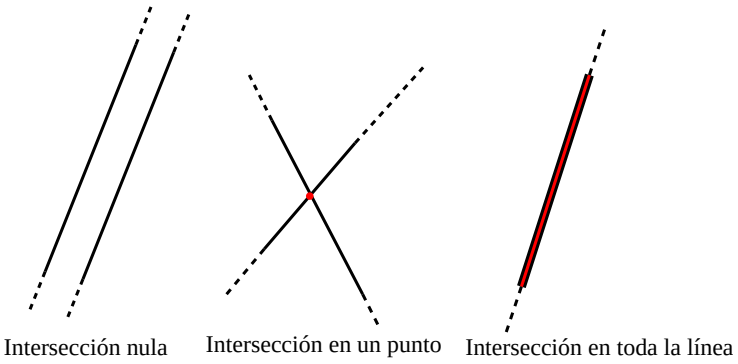


Figure 3.4: Intersección de dos líneas

3.2 Aritmética de punto flotante

Las computadoras están basadas en aritmética de punto flotante y debido a que esta puede ser distinta entre cada máquina, los constructores del hardware se han puesto de acuerdo para definir un estándar con la IEEE-std754-1984 [1]. Acá se define un conjunto de números que pueden ser representados por una máquina de 32 ó 64 bits, estos números están lejos de ser el conjunto de los reales $\mathbb{R}$ y por el contrario se comportan de forma extraña, burlando las reglas de la aritmética de los reales y definiendo sus propias reglas, un ejemplo de esto es que tan solo la ley asociativa de la suma, que dice para el cuerpo de los reales $\mathbb{R}$ que, $x+y+z = (x+y)+z = x+(y+z)$ y siempre se cumple. Para los conjutos de números `double` y/o `float` esto ya no siempre será válido. Las reglas de truncamiento y redondeo pueden hacer que un algoritmo se comporte de forma distinta y operaciones que toman deciciones lógicas sobre operaciones cercanas a cero o a uno, generan agujeros sobre la recta de los reales cambiando su estructura algebráica.

Ejemplo 1 *Predicado de Orientación de tres puntos ordenados.* `orientacion(p,q,r)` *donde* $p(p_x, p_y)$, $q(q_x, q_y)$ *y* $r(r_x, r_y)$ *representan los puntos. Los posibles resultados del predicado son cualquiera de la siguiente enumeración:* {`Gira a la derecha = -1`, `Co-lineal = 0`, `Gira a la izquierda = 1`}. *La implementación de este predicado sería:*

$$orientacion(p,q,r) = sign\left(det\begin{bmatrix}1 & p_x & p_y\\ 1 & q_x & q_y\\ 1 & r_x & r_y\end{bmatrix}\right) \tag{3.2}$$

al implementar el predicado con números tipo `float`*, el redondeo provocaría: 1) clasificar los puntos como* ± 1 *cuando en realidad eran co-lineales, 2) clasificar como co-lineal cuando en realidad no lo era y 3) clasificar en* $+1$ *cuando en realidad era* -1 *o viceversa.*
Se podría pensar que este tipo de problemas se soluciona con una mejor precisíon pero la fig:3.5, muestra lo que pasa con el extended `double` *del estándar IEEE754, usando con* gfortran*,*

Estos resultados erróneos de este predicado sencillo llevan a algoritmos lógicamente correctos a comportamientos impredecibles.

Un error sobre un predicado o una construcción, como por ejemplo: *la intersección de una línea y dos triángulos unidos por una arista o lado, para cual la línea atraviesa los triángulos cerca de la arista compartida*, es que el error aritmético mágicamente provoca una falla de clasificación en los predicados y al

[1] la última versión del estandar salió en 2008

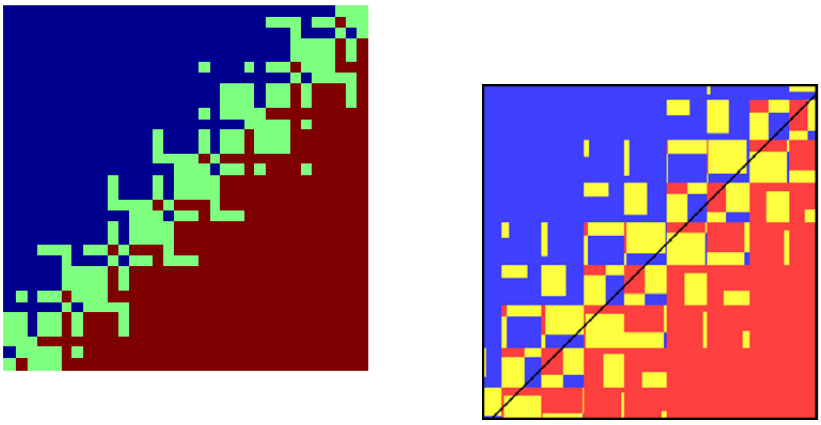


Figure 3.5: Orientación con float y extended double[24]

preguntar si existe intersección que en ese caso es obvia, resulta que no encuetra intersección por más de que el algoritmo sea correcto en su forma lógica.

En la figura 3.6 se tiene una parte de un polígono, en donde su topología puede ser especificada por una lista ordenada de vértices. Dos vértices consecutivos de en la lista y el último con el primero forman la totalidad de los segmentos que definen la frontera del polígono. Al evaluar un predicado implementado con la IEEE-754, la intersección de una línea con ese polígono se puede obtener los casos de la figura 3.6, donde la mayoría de los casos serán *correctos*, pero cuando falla la aritmética en su redondeo puede darse que: 1) *intersección incorrecta*: el caso en que una línea intersecte dos segmentos consecutivos ó 2) *intersección nula*: en donde la linea pasa por arte de magia a través de la frontera del polígono sin generar intersección y este es el error más peligroso ya que puede ser silencioso y arrojar resultados incoherentes. Es posible superar este impase usando aritmetíca exacta, disponible en librerías como *GNU MultiPrecision arithmetic* (**GMP**), la librería de grandes enteros **CORE** ó la librería de código cerrado **LEDA**, sin embargo la aritmética exacta no es una solución para el tiempo real.

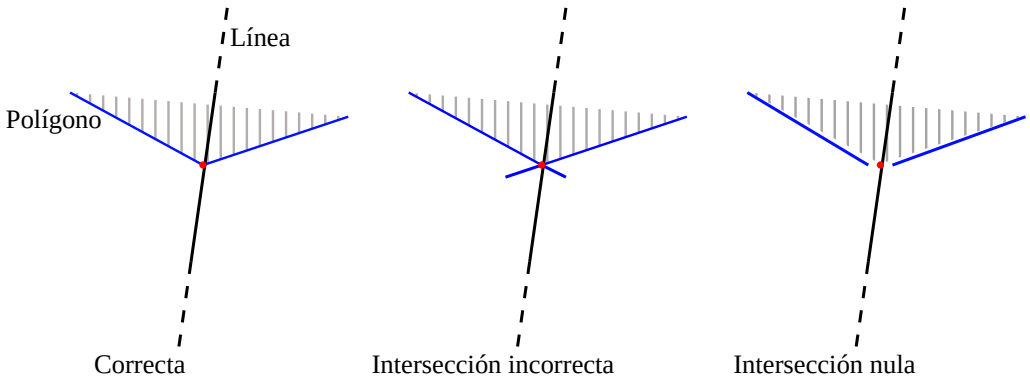


Figure 3.6: Falla en la intersección a causa de la aritmética

Esquema de una implementación de uno de los predicado usados en un mallador triangular 2D, con relaciones adecuadas de jacobiano y aspecto aptas para **FEM**, para números del estandar IEEE, robusto y

adaptativo para la solución del mallador basado en Delaunay [40].

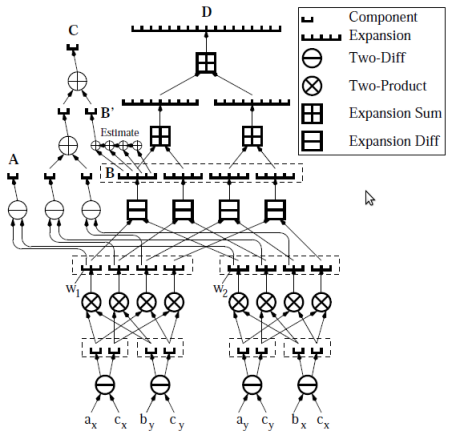


Figure 3.7: Predicado de orientación 2D sobre tres líneas[40]

3.3 Computación Geométrica Exacta (ECG)

La *robustez* en un algoritmo se puede definir de muchas formas para un algoritmo geométrico, sin embargo el término acá hace referencia a que los resultados sean correctos y consistentes. En un algoritmo geométrico los cálculos aritméticos son utilizados para definir las cuestiones topológicas de los objetos geométricos y sus características los cuales intervienen en el algoritmo, como son: que un ángulo sea obtuso o agudo, que un punto pertenezca a una línea o a un plano, o, que este por delante de un plano orientado o por detrás. Éstas características que son *predicados geométricos* requieren que sus resultados sean topológicamente correctos, esta es la piedra angular de la robustez.

En las dos últimas décadas la comunidad de la geometría computacional a abordado de muchas formas este problema, con soluciones como la *geometría epsilon* que consiste en ensanchar toda identidad geométrica como: puntos, líneas y/o planos para filtrar los predicados y producir construcciones válidas, pero esta solución no es válida topológicamente dando los mismos problemas de la geometría soportada sobre los numeros del estandar IEEE-754 e incluso complicandola más.

Otra tendencia fue analizar la herramienta en la que se implentan los algoritmos, es decir el computador, para mirar bajo los ojos de la *teoría de números* la topólogia de los números `float` y los `double` en donde claramente la carencia de *completitud* de su espacio y las caracteristicas en las cercanias al *cero* y al *uno* forman comportamientos que en muchas ocaciones son la clave de la robustez geométrica. Como es bien sabido los elementos *cero* y *uno* son las identidades de las operaciones estructurales de suma y multiplicación respectivamente, y los predicados que al final están sobre estas dos operaciones filtran el primer problema como el *acotamiento a cero*, este acotamiento genera una nueva estructura algebraica sobre un nuevo conjunto de números que contiene tanto a los `float` como a los `double` y consiste en jugar con los números cercanos *cero* que son el producto de una suma o una multiplicación para luego mirar si existe incertidumbre en su resultado. En caso de que exista incertidumbre se utiliza entonces una nueva estructura aritmética que obviamente es más precisa que la anterior y por lo tanto requiere de una mayor memomoria del sistema para sus operaciones. Estas aritméticas ya no estan implementadas en hardware y por lo tanto son mas lentas que las del IEEE-754. Sin embargo la ocurrencia de estos casos en un algoritmo es bajo, pero con que

ocurra una sola vez el algoritmo se rompe, por eso es frecuentemente visto en los códigos de malladores y programas de CAD que el mayor esfuerzo esta encaminado a la robustez de los predicados.

La importancia de CGAL en esta parte es que la artimética se puede manejar de forma modular y esto da la posibilidad de optimizar mejor un algoritmo en casos particulares. Algunas de las aritméticas que se usan además de las tradicionales son *la aritmetica de filtros* y *aritmetica exacta*, que están implementadas como anillos o cuerpos según la necesidad de las operaciones.

3.4 Estructuras de información topológica

Además de los predicados y los constructores de primitivas básicas, existen geometrías compuestas por primitivas básicas que conservan ciertas relaciones entre ellas. Principalmente la geometría es asociada a una lista de vértices que conservan una posición en el espacio y por lo tanto definen sus medidas (por esto *geometría: medida de la tierra*). Pero esos puntos asociados a un vértice y una posición están vinculados a primitivas básicas como, puntos, líneas, triángulos, planos, etc., que comienza a tener ideas topológicas. Estas ideas de carácter de completitud, continuidad y/o combinatorio que permiten equivalencia entre los espacios. Las primitivas básicas a su vez se pueden agrupar creando estructuras que complican las ideas topológicas y la forma en que se lleva la implementación es por medio de estructuras combinatorias que finalmente definen la topología de la lista de vértices.

Las estructuras topológicas son las que finalmente le dan la forma a una lista de vértices, estas estructuras suelen ser ordenadas para definir orientaciones, el caso más sencillo es un triángulo, en donde su geometría se define por tres vértices $\{p, q, r\}$. Entonces toda permutación par conservaria la orientación, es decir, en este caso toda rotación seria equivalente topológica y geométricamente hablando, $\{q, r, p\}$ y $\{r, p, q\}$. Por ejemplo $\{p, r, q\}$ no conserva la orientación. En el caso de triangulaciones y poliedros las estructuas son complicadas y dependiendo de los requerimientos unas son utilizadas y otras no.

Las mallas triangulares superficiales sobre poliedros y mallas de tetrahedros son los datos que se deben procesar dentro de la colisión e interacción de la herramenta con el tejido. Las condiciones de contorno de los problemas FEM-BEM son el resultado de seleccionar de forma rápida los vértices que entran en contacto. La colisión es una búsqueda que puede ser optimizada por medio de árboles.

Por ejemplo la estructura Half-Edge (figura 3.8) permite representar manifold-2d orientados y sirven para: recorrer las caras que conforman el polihedro, recorrer la adyacencia de vértices y mantener la orientación de la superficie. Esta estructura se utiliza para optimizar la colisión entre polihedros, caso bastante común para la simulación de una herramienta con tejidos [21].

Algunos inconvenientes es que asegurar la orientación de los manifolds no es sencillo por lo tanto la construción de la estructura debe ser un preproceso de todo poliedro, que además debe coincidir con la malla volumétrica generada sobre el dominio a discretizar por los elementos finitos. Cuando este problema no se puede solucionar el detector de colisiones debe utilizar una sopa de triángulos, que en definitiva no es un manifold orientado.

Exiten otras estructuras sobre las mallas de triángulos en 2D y tetrahedros en 3D (fig.3.9)que mantienen las relaciones de adyacencia y permiten mediante una triangulación de Delaunay [7] y un kd-tree con complejidad en $\mathcal{O}(\log n)$ para la búsqueda de puntos vecinos, regiones vecinas y puntos más cercanos fig.3.10, estas estructuras ahorran la evaluación de todo el algoritmo de colisión.

Otra ventaja de estas estructuras es que mantiene las relaciones combinatorias de los vértices separadas de la geomtría. El kd-tree es un árbol binario que se utiliza como preproceso para un ordenamiento espacial,

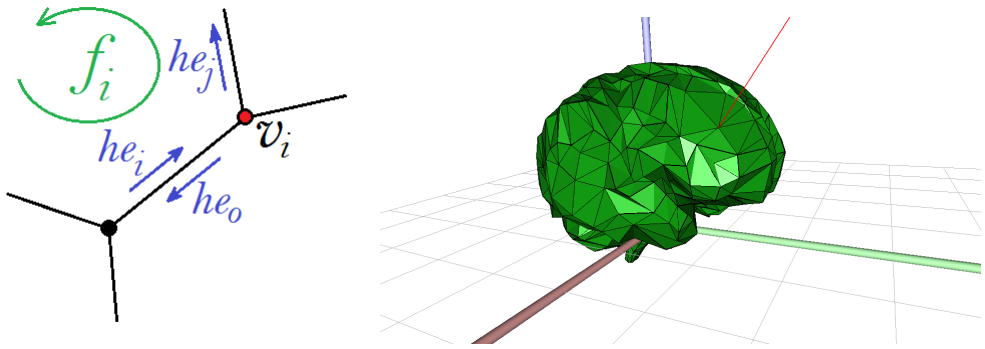


Figure 3.8: estructura Half-edge

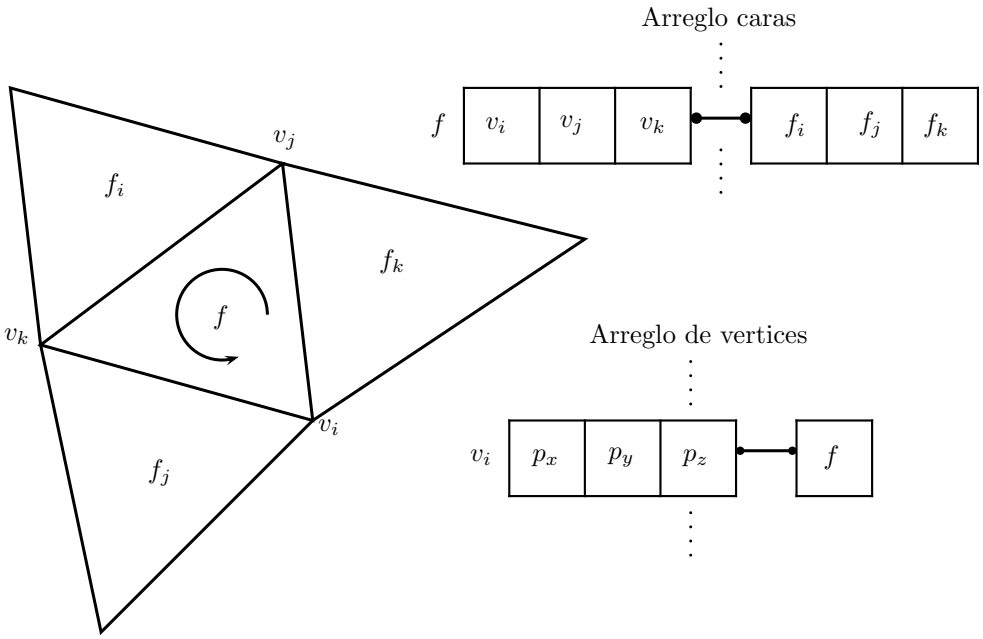


Figure 3.9: Estructura para triangulaciones

utiliza hiperplanos para partir en dos cada región del espacio (ver figura 3.11).

En cuanto a las colisiones, se utilizan árboles que mantiene la información de forma jerárquica de los volúmenes de contorno que envuelven el objeto, esto se hace con el fin de utilizar predicados menos costosos para descartar casos de no colisión. El manejador de escena para aplicación de tejidos utiliza un árbol dinámico para encontrar pares de AABB (axis aligned bounded boxes) para evaluar un predicado inicial, y luego se utiliza otro árbol AABB para la intesección de dos poliedros con un árbol estático kd-tree.

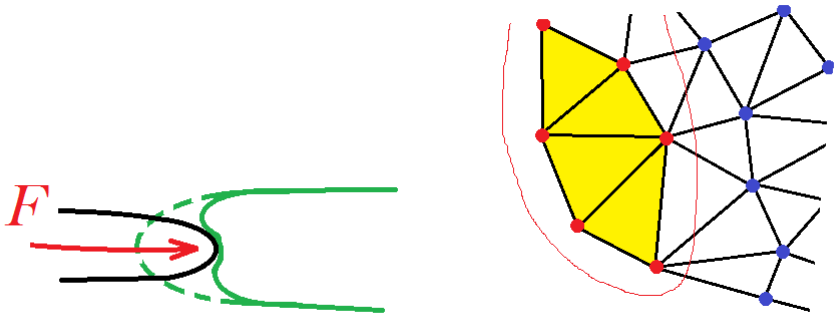


Figure 3.10: k-nearest

3.5 CGAL

La librería CGAL, es una libreria de algoritmos geométricos que lleva más de 15 años trabajando en su diseño, bajo el paradigma de la programación genérica, varias universidades europeas y norteamericas han unido fuerzas para mantener, extender y optimizar esta librería de C++. La filosofia GNU de este preyecto permanece en su cabeza, y durante años sus paquetes han sido y continuan siendo estudios investigativos de tesis doctorales y de maestría, en el área de la geometría computacional y topología aplidada con aplicaciones en diferentes áreas como la medicina y el análisis de imagenes, los programas tipo CAD-CAM-CAE, la computación grafica, los video juegos, la robótica y la mecánica computacional entre otros. Muchas industrias y universidad más investigan en sus proyectos específicos usando esta librería como soporte ahorrando mucho esfuerzo y dedicación a problemas que este proyecto comparte de antemano para el mundo.

La mayoria de estos algoritmos son utilizados como cajas negras por debajo de los programas como Solid Edge, Inventor u Ansys, en forma de malladores, detectores de colisión, generación de geometrías por medio de operaciones booleanas como los son las *Constructive Solid Geometry* **CSG** y muchas otras más. Estas cajas negras que al final probocan que el usuario de estos programas se vea limitado a ser un simple chofer de sus herramientas. CGAL abre sus códigos, la magnitud de estas librerias ascienden a más medio millon de líneas de código, CGAL mantiene una constante revisión del código y cada seis meses saca una versión nueva con sus nuevos avances y corrigiendo errores anteriores. Las nuevas tendencias en lenguajes como MATLAB, que ahora se interesan fuertemente en algoritmos geometricos, han involucrado a CGAL como solución.

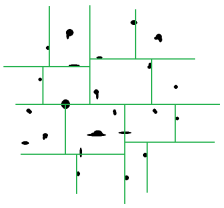


Figure 3.11: kd-tree

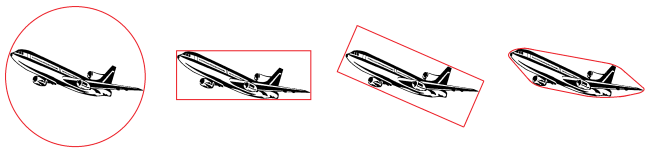


Figure 3.12: gerarquia volumenes de contorno

3.5.1 Estructura de la librería

La documentación comienza con la aritmética modular y el álgebra de cada conjuto de números que se puede trabajar, se abordan desde la aritmética de la IEEE, pasando por la de Big Integers como lo son GMP, CORE y LEDA. Una aritemética importante a mencionar es la de filtros, que será fundamental ya que es una aritmética adaptativa que permite alcanzar la robuztes sin perder desempeño en tiempo de ejecución.

Una vez se entiende la capa más baja de la libreria que es la de aritmética modular, esta es utilizada en un conjuto de predicados y constructores, este conjuto es llamdo *kernel geométrico* y es la piedra angular de la libreria CGAL [18], su diseño ha venido evolucionando desde el comienzo de la librería. Este kernel es a un algoritmo geométrico, lo que una *Unidad Aritmética Lógica* ALU es para para un algoritmo aritmético.

Sobre ese kernel geomeétrico, la libreria desarrolla paquetes como voronoi, Delaunay, convex hulls, trianguladores, sumas Minkowski, diesmados sobre polihedros, malladores superficiales, volumetricos en n-D, descomposición compuesta de polígonos y polihedros, snap rounding, muchos más. Cabe mencionar que todos estos algoritmos se pueden utilizar con cualquiera de las aritmeticas previamente mencionadas y que los kernels pueden variar dependiendo de su necesidad en la aplicación. Además de todos estos algoritmos es posible diseñar nuevos predicados como es el caso de los recientes temas investigativos de CGAL donde la calidad de los simplex en los malladores debe de ser apta para no tener problemas en los retos de la mecánica computacional.

La librería CGAL separa todas las ideas de las estructuras combinatorias de los objetos geometricos y la aritmetica robusta, los algoritmos son escritos a parte del tipo de estructura y los tipos de datos son independientes de los algoritmos y su aritmetica utilizada, esta libreria es modular y esta diseñada con el paradigma de programación genérica. Como se puede ver (fig3.13, la capa más elemental es la aritmética, utilizada para declarar predicados y constructores con diferentes grados de precisión, este conjunto define el kernel geométrico y con este y las estructuras topológicas se definien los algoritmos de la librería.

3.5.2 Algoritmos utilizados en el simulador de tejido

Los algoritmos a utlizar en el documento diseñados por CGAL son: 1) Trianguladores (Delaunay, Constrained Delaunay y Regular triangulation) y sus repectivas estructuras, 2) Poliedros y sus respetivas estructuras, 3) Malladores (2D, 3D y superficiales 3D), y 4) AABB tree (sobre poliedros cerrados y sopas de triángulos).

Los trianguladores (en 2D[53] y 3D[34]) son usados sobre geometrías sencillas como cuadrados y cubos, para hacer pruebas rápidas en el código, estas triangulaciones son prefectas para probar la veracidad de los algoritmos, ya que las matrices son bien formadas y no tiene problemas de reventarse, ademas cualquier politopo convexo es facilmente mallado. Las estructuras de estas triangulaciones[35, 33] son usadas para encontrar problemas de vértices y caras cercanos[9] y para recorrer las mallas de forma rápida en el contacto

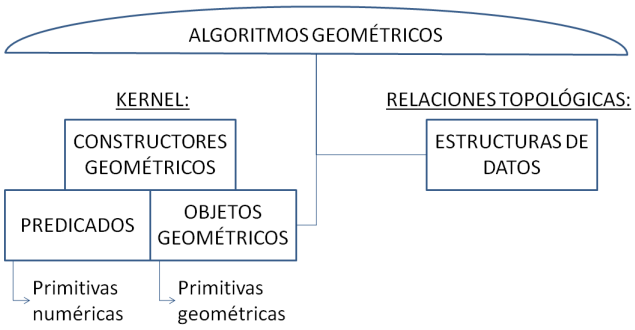


Figure 3.13: Arquitectura de CGAL

de la herramienta con el tejido.

Los poliedros[22] y las estructuras Half-Edge[23], optimizan de la mejor manera la colisión, tanto sobre el árbol de búsqueda como recorriendo caras adyacentes. La dificultad que se tiene es que no toda sopa de triángulos es convertible a Half-Edge y para estos casos toca utlizar las triangulaciones que son mas lentas que estas estructura.

El mallador 2D[37] están badados en Triangle-2D[41], este mallador devuelve una estructura con una triangulación restringida en su contorno y con propiedades de Delaunay. El mallador es robusto y sirve para todo tipo de geometría en 2D. Es posible hacer refinamiento de malla.

El mallador 3D[1] es un paquete reciente que están en constante desarrollo, es robusto pero su kernel debe de ser exacto en los predicados y contrucciones. Este mallador devuelve una estructura que permite trabajar con la frontera y con el interior. La frontera es la que se debe contruir posteriormente en una Half-Edge, para preprocesar el árbol de colisión. Este mallador sirve como mallador superficial y trabajar con BEM.

El mallador superficial 3D[38], es robusto devuelve un polyhedro y su Half-Edge, es ideal para trabajar con BEM pues su la calidad de la malla es adecuada y con jacobianos homógeneos en toda los triángulos.

El árbol AABB[2], es un prepoceso que se hace sobre la sopa de triángulos en el peor caso o sobre el Half-Edge del poliedro en el mejor de los casos, la ventaja de este árbol es que la interacción de la herramienta con el tejido se hace lo más rápido posible. Luego se usa las otras estructuras para moverse sobre las vecindades.

Chapter 4

Modelamiento de tejidos

...veras que la cuchara no es la que se deforma si no tú...

4.1 Mecanica del medio continuo

Cualquier sólido en la mecánica de medio continuo se puede analizar de forma dinámica o estática, mediante el uso de tres ecuaciones: una relación cinemática, una ecuación de equilibrio o movimiento y una relación constitutiva.

4.1.1 Relaciones Cinemáticas

Sea $\Omega_0 \subset \mathbb{R}^3$, el dominio de todos los puntos espaciales $\vec{\mathbf{R}}$ en $\mathbb{R}^3$, que marcan la posición de un punto material $\mathcal{P}$ que forma parte del sólido $\mathcal{B}$ en una *configuración inicial o de referencia* en $t = 0$ (Ver la figura 4.1 hasta la definición de gradiente de deformación),

$$\Omega_0 = \{\forall\, \vec{\mathbf{R}} \in \mathbb{R}^3 : \ \vec{\mathbf{R}} \text{ es la posición de } \mathcal{P} \in \mathcal{B} \text{ para } t = 0\}$$

de la misma manera se llamará a la *configuración deformada* en $t = \tau$ por,

$$\Omega_\tau = \{\forall\, \vec{\mathbf{r}}_\tau \in \mathbb{R}^3 : \ \vec{\mathbf{r}}_\tau \text{ es la posición de } \mathcal{P} \in \mathcal{B} \text{ para } t = \tau\}.$$

Posicion, Desplazamiento y Sistemas Coordenados

Tanto $\vec{\mathbf{r}}$ como $\vec{\mathbf{r}}_\tau$ son vectores representando las posiciones de $\mathcal{P}$ sujetas a dos sistemas de coordenadas diferentes, $(\mathcal{O}, \{X_I\})$ y $(\mathcal{O}_\tau, \{x_i\})$. El primer sistema (de la configuración indeformada) consta de un punto arbitrario denominado *origen*, en este caso $\mathcal{O}$ y un conjunto de tres ejes $\{X_I\} = \{X_1, X_2, X_3\}$, que no necesariamente son: rectos, ortogonales y cuya interseccion coincida con el origen $\mathcal{O}$. De la misma forma se cumple para $(\mathcal{O}_\tau, \{x_i\})$, lo anterior.
Se define el *vector entre origenes* $\vec{\mathbf{b}}$ como el vector que representa el segemento dirigido de $\overline{\mathcal{O}\mathcal{O}_\tau}$.
Se define el *desplazameinto* $\vec{\mathbf{u}}_\tau$ de $\mathcal{P} \in \mathcal{B}$ entre la configuración deformada y la configuración de referencia, como $\vec{\mathbf{u}}_\tau = \vec{\mathbf{r}}_\tau + \vec{\mathbf{b}} - \vec{\mathbf{R}}$.

Encontrar una mapeo que tome $\mathcal{T} : \Omega_0 \to \Omega_\tau$, no es una tarea fácil, sin embargo aquí hay dos ejemplos.

Ejemplo 2 *Supongase que el cuerpo $\mathcal{B}$ solo se translada por un vector $\vec{\mathbf{t}}$, encontrar el mapeo $\mathcal{T}$:*

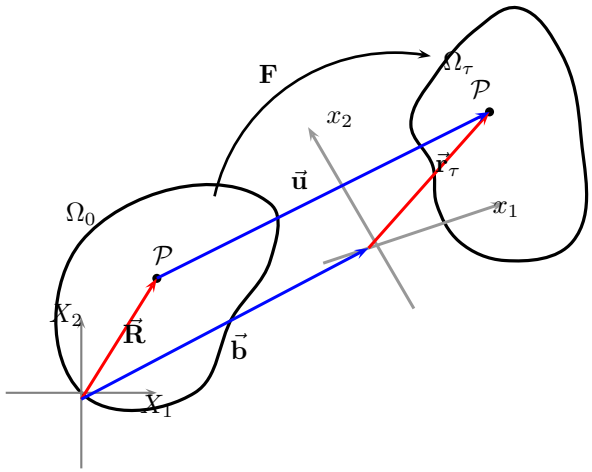


Figure 4.1: Configuracion Indeformada y Deformada

Solución: *El desplazamiento para todo* $\mathcal{P} \in \mathcal{B}$ *es* $\vec{\mathbf{u}}_\tau = \vec{\mathbf{t}}$ *y* $\mathcal{T} : \Omega_0 \to \Omega_\tau$*, que sobre cada elemento es* $\mathcal{T} : \vec{\mathbf{R}} \mapsto \vec{\mathbf{r}} = \mathcal{T}(\vec{\mathbf{R}})$*, por lo tanto el mapeo queda como:*

$$\mathcal{T}(\vec{\mathbf{R}}) = \vec{\mathbf{R}} + \vec{\mathbf{t}} - \vec{\mathbf{b}} \tag{4.1}$$

Observación: El vector $\vec{\mathbf{r}}$*, que representa una posición de* $\mathcal{P}$ *en* $t = \tau$*, se puede describir como una combinación lineal de la base* $\{\vec{\mathbf{g}}_i\}$ *del sistema coordenado* $(\mathcal{O}_\tau, \{x_i\})$ *como:*

$$\vec{\mathbf{r}} = r^i \vec{\mathbf{g}}_i \tag{4.2}$$

y a su vez se puede representar con la base de $\{\vec{\mathbf{G}}_\mathbf{I}\}$*, como* $\vec{\mathbf{r}} = r^I \vec{\mathbf{G}}_\mathbf{I}$*, pero este último no tiene la información de la ubicación de* $\mathcal{P}$*. El hecho de que este vector pierda su significado, lleva a que la posición de un punto no es invariante y depende del observador, la propiedad de invarianza es fundamental para la formulación del modelo de tejidos.*
El vector de desplazamiento sí conserva su invarianza y la descripcion de $\vec{\mathbf{u}}_\tau = u^i \vec{\mathbf{g}}_\mathbf{i}$ *o* $\vec{\mathbf{u}}_\tau = u^I \vec{\mathbf{G}}_\mathbf{I}$ *son equivalentes físicamente, lo único que cambia es su descripción.*

Ejemplo 3 *Ahora supongase que el cuerpo* $\mathcal{B}$ *del ejemplo anterior, además de transladarse por un vector* $\vec{\mathbf{t}}$*, se le aplica una rotatición* $\mathbf{Q}$ *antes de transladarse, encontrar el mapeo* $\mathcal{T}$*:*

Solución: *Acá el desplazamiento ya no es el mismo para todo* $\mathcal{P} \in \mathcal{B}$ *y el mapeo no es tan sencillo como el anterior, sin embargo, si se conoce la matriz de rotación* $\mathbf{Q}$ *se sabe que:*

$$\mathcal{T}(\vec{\mathbf{R}}) = \mathbf{Q} \cdot \vec{\mathbf{R}} + \vec{\mathbf{t}} - \vec{\mathbf{b}} \tag{4.3}$$

el cual no es un mapeo lineal.

Gradiente de deformación

Se define como el tensor bipunto a $\mathbf{F}$ el *gradiente de deformación*, que asume que el material se deforma localmente de forma suave, es decir de la variación de, $\vec{\mathbf{r}} = \vec{\mathbf{R}} + \vec{\mathbf{u}}_\tau - \vec{\mathbf{b}}$, o sea $d\vec{\mathbf{r}} = d\vec{\mathbf{R}} + d\vec{\mathbf{u}}_\tau$ se puede aproximar por un mapeo lineal $\mathbf{F}$ tal que $d\vec{\mathbf{r}} = \mathbf{F} \cdot d\vec{\mathbf{R}}$, de la forma $\mathbf{F} : \Omega_0 \ni d\vec{\mathbf{R}} \mapsto d\vec{\mathbf{r}} \in \Omega_\tau$, como mapea

de un dominio a otro, cada uno de ellos tiene su propia base y queda (ver figura 4.2),

$$d\vec{\mathbf{r}} = \mathbf{F} \cdot d\vec{\mathbf{R}} \tag{4.4}$$

$$dr^i \vec{\mathbf{g}}_i = \mathbf{F} \cdot dR^I \vec{\mathbf{G}}_I \tag{4.5}$$

$$dr^i \vec{\mathbf{g}}_i = F^i_J \, \vec{\mathbf{g}}_i \vec{\mathbf{G}}^J \cdot dR^I \vec{\mathbf{G}}_I \tag{4.6}$$

$$dr^i \vec{\mathbf{g}}_i = F^i_I \, dR^I \, \vec{\mathbf{g}}_i \tag{4.7}$$

El hecho de que $\det \mathbf{F} \neq 0$ asegura la existencia de el mapeo $\mathcal{T}$.

Ejemplo 4 *Continuando con el ejemplo anterior, encontrar el gradiende de deformacion:*

Solución:

$$d\vec{\mathbf{r}} = d(\mathbf{Q} \cdot \vec{\mathbf{R}} + \vec{\mathbf{t}} - \vec{\mathbf{b}}) \tag{4.8}$$

$$= d(\mathbf{Q} \cdot \vec{\mathbf{R}}) = d\mathbf{Q} \cdot \vec{\mathbf{R}} + \mathbf{Q} \cdot d\vec{\mathbf{R}} \tag{4.9}$$

$$= \mathbf{Q} \cdot d\vec{\mathbf{R}} \tag{4.10}$$

Observación: $\mathbf{Q}$ *es un gradiente de deformación donde* $\det \mathbf{Q} = \pm 1$.

Como se puede ver el tensor bipunto requiere del conocimiento de las base deformada y de la indeformada, sin embargo la información se puede compactar a una sola base si se toma ds, la longitud del vector $d\vec{\mathbf{r}}$, como $ds^2 = d\vec{\mathbf{r}} \cdot d\vec{\mathbf{r}} = d\vec{\mathbf{R}} \cdot \mathbf{F}^T \cdot \mathbf{F} \cdot d\vec{\mathbf{R}}$, de donde se define $\mathbf{C} = \mathbf{F}^T \cdot \mathbf{F}$, éste último se conoce como el tensor de *Cauchy-Green por derecha*, que solo requiere de una base para su descripción y además es simétrico.

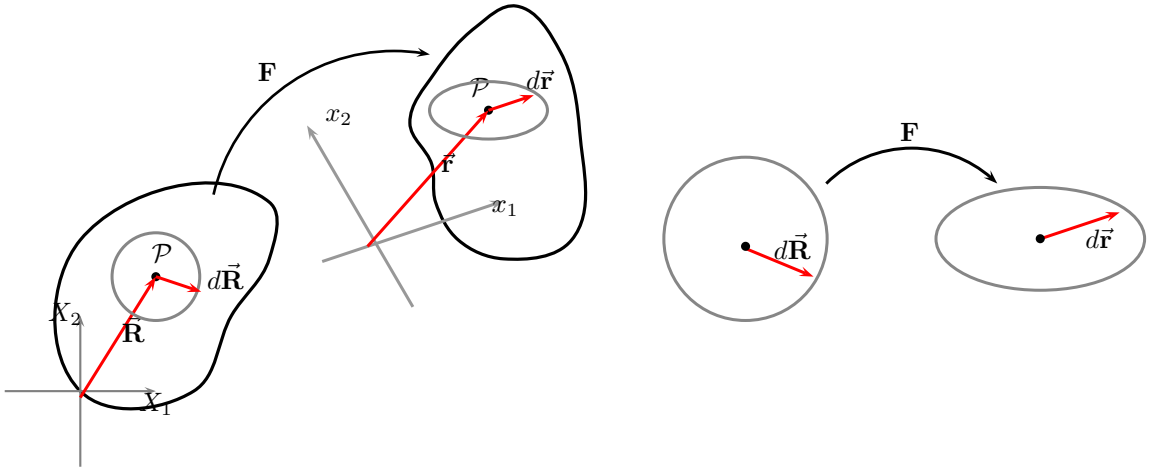


Figure 4.2: Efecto del gradiente de deformación

Tensor de deformación unitaria

El tensor de Cauchy-Green por derecha $\mathbf{C}$ contiene informaciones de alargamiento y orientaciones en general, pero a veces se suele trabajar no con esas magnitudes sino con los cambios de longitud y las deflexiones de las orientaciones respecto a la configuración inicial, para eso, $ds^2 - dS^2 = d\vec{\mathbf{r}} \cdot d\vec{\mathbf{r}} - d\vec{\mathbf{R}} \cdot d\vec{\mathbf{R}} = d\vec{\mathbf{R}} \cdot (\mathbf{C} - \mathbf{I}) \cdot d\vec{\mathbf{R}}$ y por definición se tiene que el tensor de *deformación unitaria* es $2\mathbf{E} = \mathbf{C} - \mathbf{I}$. Como es de esperarse este tensor tambien simetrico y es descrito con una sola base, la indeformada.

Ejemplo 5 *Encontrar los tensores* $\mathbf{C}$ *y* $\mathbf{E}$ *de la rotación* $\mathbf{Q}$*:*

Solución: *El tensor de Cauchy-Green* $\mathbf{C} = \mathbf{Q}^T \cdot \mathbf{Q} = \mathbf{I}$ *y el tensor de deformación unitaria* $2\mathbf{E} = \mathbf{I} - \mathbf{I} = \mathbf{O}$.
Observacion: Una rotación no tiene deformaciones.

Ejemplo 6 *Encontrar en un mapeo de: un escalado* $\mathbf{S}$ *y una translación* $\vec{\mathbf{t}}$ *consecutivas, los siguientes tensores,* $\mathbf{F}$*,* $\mathbf{C}$*,* $\mathbf{E}$*:*

Solución: *Se tiene que* $d\vec{\mathbf{r}} = d(\mathbf{S} \cdot \vec{\mathbf{R}} + \vec{\mathbf{t}} - \vec{\mathbf{b}}) = \mathbf{S} \cdot d\vec{\mathbf{R}}$*. Por lo que se concluye que* $\mathbf{S}$ *es un gradiente de deformación. El tensor de deformación de Cauchy es* $\mathbf{C} = \mathbf{S}^2$*, debido a la simetría de* $\mathbf{S}$*, el tensor de deformación unitaria Lagrangiana es* $2\mathbf{E} = \mathbf{S}^2 - \mathbf{I}$.

Los tensores $\mathbf{F}$, $\mathbf{C}$ y $\mathbf{E}$, son usados para la formulación de las relaciones constitutivas. Estas relaciones están en base a las invarianzas tensoriales de los tres tensores mencionados.

4.1.2 Ecuaciones de movimiento y relaciones de equilibrio

Las fuerzas en los sólidos son de dos naturalezas de volumen (o campo) y superficiales (o de contacto). Las fuerzas de volumen o campo que generalemente son gravitacionales y/o electromagnéticas. Las fuerzas de contacto definen los vectores de tracción.

Fuerzas volumétricas

Las fuerzas volumétricas dependen de las propiedades intrínsecas del material y de la intensidad del campo que actúa sobre el dominio Ω_τ que ocupa el sólido $\mathcal{B}$ en cuestión, por lo general las propiedaes intrínsecas son representadas por un campo escalar definido como $\phi : \Omega_\tau \mapsto \mathbb{R}$ que escala al campo vectorial $\vec{\mathbf{f}}_\rho : \Omega_\tau \mapsto \mathbb{R}^3$ de alguna naturaleza física como las fuerzas gravitacionales. Por lo general la fuerzas volumetrias se toman como la superposición de todos los campos $\vec{\mathbf{f}} : \Omega_\tau \mapsto \mathbb{R}^3$ y actúan por unidad de volumen, volumen que se toma en la configuración deformada.

Fuerzas de contatcto

Sea la fuerza $d\vec{\mathbf{p}}$ distribuida sobre una superficie orientada $d\vec{\mathbf{a}}$ en la configuración deformada, es la que define un vector de fuerza por unidad de área llamado tracción $\vec{\mathbf{t}} = \boldsymbol{\sigma} \cdot \hat{\mathbf{n}}$, donde $\hat{\mathbf{n}}$ es el vector unitario que dirige la orientación de la superficie en el punto de aplicación de la tracción $\vec{\mathbf{t}}$. El *tensor de esfuerzo de Cauchy* se define como,

$$\boldsymbol{\sigma} = \frac{d\vec{\mathbf{p}}}{d\vec{\mathbf{a}}} \tag{4.11}$$

El problema de este tensor es que se describe en la base deformada, que por lo general no se conoce, por esto se utilizan tensores de seudo-esfuerzo para la formulación del equilibrio.

Segundo tensor de Piola-Kirchhoff

El *segundo tensor de Piola-Kirchhoff* permite trabajar desde la configuración indeformada dejando las posibles deformaciones de área y transformaciones de fuerza en función del gradiende de defomraciones $\mathbf{F}$. Así como $d\vec{\mathbf{r}}$ un segmento de línea en la deformada, se puede relacionar por $d\vec{\mathbf{r}} = \mathbf{F} \cdot d\vec{\mathbf{R}}$, con $d\vec{\mathbf{R}}$ el mismo

segmento antes de deformase. Un vector de área orientada $d\vec{\mathbf{a}} = d\vec{\mathbf{r}}_{(1)} \times d\vec{\mathbf{r}}_{(2)}$, se puede escribir como,

$$d\vec{\mathbf{a}} = \mathbf{F} \cdot d\vec{\mathbf{R}}_{(1)} \times \mathbf{F} \cdot d\vec{\mathbf{R}}_{(2)} \tag{4.12}$$

$$= F_I^i \, dR_{(1)}^I \vec{\mathbf{g}}_i \times F_J^j \, dR_{(2)}^J \vec{\mathbf{g}}_j \tag{4.13}$$

$$= F_I^i \, F_J^j \, dR_{(1)}^I \, dR_{(2)}^J \vec{\mathbf{g}}_i \times \vec{\mathbf{g}}_j \tag{4.14}$$

$$= F_I^i \, F_J^j \, dR_{(1)}^I \, dR_{(2)}^J \epsilon_{ijk} \vec{\mathbf{g}}^k \tag{4.15}$$

en este punto es importante notar que $\{\vec{\mathbf{g}}^k\}$ representa la base contravariante de la configuración deformada que en definitiva es definida de forma arbitraria y no tiene por que tener nada que ver con la base indeformada $\{\vec{\mathbf{G}}^K\}$. Del hecho de que el material en un punto este sometido a un gradiente de deformación $\mathbf{F}$, permite pensar en una base $\{\vec{\mathbf{g}}^K\}$ que representa como se deforma la base indeformada $\{\vec{\mathbf{G}}^K\}$ y se puede decir que $\vec{\mathbf{g}}_K = \mathbf{F} \cdot \vec{\mathbf{G}}_K$ por que,

$$\mathbf{F} = \vec{\mathbf{g}}_I \vec{\mathbf{G}}^I = F_I^i \vec{\mathbf{g}}_i \vec{\mathbf{G}}^I \tag{4.16}$$

$$\mathbf{F} \cdot \vec{\mathbf{G}}_K = \vec{\mathbf{g}}_I \vec{\mathbf{G}}^I \cdot \vec{\mathbf{G}}_K = F_I^i \vec{\mathbf{g}}_i \vec{\mathbf{G}}^I \cdot \vec{\mathbf{G}}_K \tag{4.17}$$

$$\Rightarrow \vec{\mathbf{g}}_K = F_K^k \, \vec{\mathbf{g}}_k \tag{4.18}$$

de la misma forma $\vec{\mathbf{G}}^K = \mathbf{F}^{\mathbf{T}} \cdot \vec{\mathbf{g}}^K$,

$$\mathbf{F}^{\mathbf{T}} = \vec{\mathbf{G}}^I \vec{\mathbf{g}}_I = F_I^i \vec{\mathbf{G}}^I \vec{\mathbf{g}}_i \tag{4.19}$$

$$\mathbf{F}^{\mathbf{T}} \cdot \vec{\mathbf{g}}^K = \vec{\mathbf{G}}^I \vec{\mathbf{g}}_I \cdot \vec{\mathbf{g}}^K = F_I^i \vec{\mathbf{G}}^I \vec{\mathbf{g}}_i \cdot \vec{\mathbf{g}}^K \tag{4.20}$$

$$\Rightarrow \vec{\mathbf{G}}^K = F_I^i \vec{\mathbf{G}}^I \, F_i^J \vec{\mathbf{g}}_J \cdot \vec{\mathbf{g}}^K = \vec{\mathbf{G}}^K \tag{4.21}$$

volviendo a las relaciones de áreas,

$$d\vec{\mathbf{a}} = F_I^i \, F_J^j \, dR_{(1)}^I \, dR_{(2)}^J \epsilon_{ijk} \vec{\mathbf{g}}^k \tag{4.22}$$

$$= F_I^i \, F_J^j \, dR_{(1)}^I \, dR_{(2)}^J \sqrt{g} \, e_{ijk} \, F_K^k \vec{\mathbf{g}}^K \tag{4.23}$$

$$= j\sqrt{g} \, e_{IJK} \, dR_{(1)}^I \, dR_{(2)}^J \, \mathbf{F}^{-\mathbf{T}} \cdot \vec{\mathbf{G}}^K \tag{4.24}$$

$$= j\sqrt{\frac{g}{G}} \, \epsilon_{IJK} \, dR_{(1)}^I \, dR_{(2)}^J \, \vec{\mathbf{G}}^K \cdot \mathbf{F}^{-1} \tag{4.25}$$

$$= \det \mathbf{F} \, d\vec{\mathbf{A}} \cdot \mathbf{F}^{-1} \tag{4.26}$$

donde j es el determinante de la representacion de $\mathbf{F}$, $\det(F_I^i)$, $\sqrt{g}$ y $\sqrt{G}$ son los determinantes de los tensores métricos de las bases $\{\vec{\mathbf{g}}^i\}$ y $\{\vec{\mathbf{G}}^I\}$ respectivamente.
Suponiendo que $d\vec{\mathbf{p}}$ sea independiente de las deformaciones se plantea una seudo fuerza $d\vec{\mathbf{P}}^* = \mathbf{F}^{-1} \cdot d\vec{\mathbf{p}}$, se define el *segundo tensor de Piola-Kirchhoff* como $\mathbf{s} = \frac{d\vec{\mathbf{P}}^*}{d\vec{\mathbf{A}}}$, igualando las fuerzas,

$$d\vec{\mathbf{p}} = \boldsymbol{\sigma} \cdot d\vec{\mathbf{a}} = \mathbf{F} \cdot d\vec{\mathbf{P}}^* = \mathbf{F} \cdot \mathbf{s} \cdot d\vec{\mathbf{A}} \tag{4.27}$$

y despejando $\det \mathbf{F} \, \boldsymbol{\sigma} \cdot d\vec{\mathbf{A}} \cdot \mathbf{F}^{-1} = \mathbf{F} \cdot \mathbf{s} \cdot d\vec{\mathbf{A}}$ se obtiene,

$$\boldsymbol{\sigma} = \mathbf{F} \cdot s \cdot \mathbf{F}^T \det \mathbf{F}^{-1} \tag{4.28}$$

Ecuaciones de movimiento

Definidas las fuerzas en los cuerpos es posible definir las equaciones de movimiento integrando todas las fuerzas e igualandolas al cambio del momento lineal en uso del principio del *momentum lineal*.

$$\int_{d\Omega_\tau} \vec{\mathbf{t}} \, da + \int_{\Omega_\tau} \vec{\mathbf{f}} \, dv = \frac{d}{dt} \int_{\Omega_\tau} \rho \vec{\mathbf{v}} \, dv \tag{4.29}$$

la primera ecuación del lado izquierdo se puede escribir como $\int_{d\Omega_\tau} \boldsymbol{\sigma} \cdot \hat{\mathbf{n}}\, da \;=\; \int_{\Omega_\tau} \overline{\nabla} \cdot \boldsymbol{\sigma}\, dv$ donde $\overline{\nabla}$ es el gradiente espacial en la deformada, es decir $\frac{\partial}{\partial \vec{\mathbf{r}}}$ y el lado derecho se puede escribir en téerminos de la aceleración como $\int_{\Omega_\tau} \vec{\mathbf{a}} \rho\, dv$,

$$\int_{\Omega_\tau} (\overline{\nabla} \cdot \boldsymbol{\sigma} + \vec{\mathbf{f}} - \rho \vec{\mathbf{a}}) \;=\; 0 \tag{4.30}$$

la ecuación anterior se puede escribir de forma local como,

$$\overline{\nabla} \cdot \boldsymbol{\sigma} + \vec{\mathbf{f}} - \rho \vec{\mathbf{a}} \;=\; 0 \tag{4.31}$$

de aplicar el principio de *momentum agular*, se llega a la necesidad de que tanto $\boldsymbol{\sigma}$ como $\mathbf{s}$, son tensores simétricos. Es posible describir la ecuación anterior en términos de la indeformada como,

$$\nabla \cdot (\mathbf{s} \cdot \mathbf{F}^{\mathbf{T}}) + \vec{\mathbf{f}}_0 \;=\; \rho_0 \vec{\mathbf{a}} \tag{4.32}$$

donde ∇ es el gradiente espacial en la indeformada $\frac{\partial}{\partial \vec{\mathbf{R}}}$.

4.1.3 Relación Constitutiva

Esta relación se encuentra en el intento de satisfacer los siguientes principios definidos brevemente:

- **Invariaza del sistema coordenado**: Uso de ecuaciones tensoriales.
- **Determinismo**: El sistema tiene memoria, es decir tiene variables de estado.
- **Acción local**: Las variables varian de forma suave localemente.
- **Equipresencia**: Las variables independientes aparecen en cada una de las ecuaciones independientes.
- **Objetividad**: Independencia del observador.
- **Admisibilidad fisica**: Cumplimiento de las leyes físicas.
- **Simetría del material**: Los tensores de esfuerzo son simétricos.

La invariaza del sistema coordenado ya se ha asegurado por el uso tensorial. Del determinismo se puede hacer la distinción de dos materiales ideales; el primero *thermo-elástico* en el cual no ocurre discipación de energía durate su deformación y por lo tanto se puede decir que no tiene memoria y no depende de como fueron sus estados anteriores, y el segundo el material *elástico* donde se desprecian los efectos de la temperatura y tambien se toma como un material sin memoria.
La acción local se satisface de hacer la siguiente aproximación,

$$\begin{aligned} \vec{\mathbf{r}}(\vec{\mathbf{R}}^*, t) \;-\; \vec{\mathbf{r}}(\vec{\mathbf{R}}, t) \;&=\; (\vec{\mathbf{R}}^* - \vec{\mathbf{R}}) \cdot \frac{\partial \vec{\mathbf{r}}}{\partial \vec{\mathbf{R}}} \\ d\vec{\mathbf{r}} \;&=\; d\vec{\mathbf{R}} \cdot \mathbf{F}^{\mathbf{T}} \end{aligned} \tag{4.33}$$

que aproxima la acción localmente a el punto $\mathcal{P}$ en $\vec{\mathbf{R}}$, como lo dice la serie de Taylor. De la misma forma con otra propiedad alrededor del punto $\mathcal{P}$ como la temperatura T sería,

$$\begin{aligned} T(\vec{\mathbf{R}}^*, t) \;-\; T(\vec{\mathbf{R}}, t) \;&=\; (\vec{\mathbf{R}}^* - \vec{\mathbf{R}}) \cdot \frac{\partial T}{\partial \vec{\mathbf{R}}} \\ dT \;=\; d\vec{\mathbf{R}} \cdot \frac{\partial T}{\partial \vec{\mathbf{R}}} \;&=\; d\vec{\mathbf{R}} \cdot \nabla T \end{aligned} \tag{4.34}$$

a los materiales modelados con acción local se les llama *materiales simples* y las ecuaciones de un material thermo-elástico quedan como,

$$\begin{aligned}
\boldsymbol{\sigma} &= \boldsymbol{\sigma}(\vec{\mathbf{R}}, \mathbf{F}, T, \nabla T) && \text{Esfuerzo} \\
\vec{\mathbf{q}} &= \vec{\mathbf{q}}(\vec{\mathbf{R}}, \mathbf{F}, T, \nabla T) && \text{Flujo de Calor} \\
u &= u(\vec{\mathbf{R}}, \mathbf{F}, T, \nabla T) && \text{Energía interna} \\
\eta &= \eta(\vec{\mathbf{R}}, \mathbf{F}, T, \nabla T) && \text{Entropía}
\end{aligned} \tag{4.35}$$

La objtividad se alcaza con la consistencia de convertir cantidades como la posición, temperaturas, velocidades, esfuerzos y gradientes de deformación, al ser descritos desde otro sistema coordenado, ese principio es importante para el balance momentum y energía.
Por último y la más importante es satisfacer las ecuaciones de conservación de masa, momentum y energia, y la segunda ley de la termodinámica para asegurar una admisibidad física. Las ecuaciones de masa y momentum quedaron resueltas en las relaciones de movimiento, el balance de energia sigue de,

$$\dot{K} + \dot{U} = P + Q$$

$$\underbrace{\frac{1}{2}\int_{\Omega_\tau} \rho \vec{\mathbf{v}} \cdot \vec{\mathbf{v}}\, dv}_{\text{Energía cinética}} + \underbrace{\int_{\Omega_\tau} \rho u\, dv}_{\text{Energía interna}} = \underbrace{\overbrace{\int_{d\Omega_\tau} \vec{\mathbf{t}} \cdot \vec{\mathbf{v}}\, da}^{\text{en frontera}} + \overbrace{\int_{\Omega_\tau} \vec{\mathbf{f}} \cdot \vec{\mathbf{v}}\, dv}^{\text{de cuerpo}}}_{\text{Trabajo}} + \underbrace{\overbrace{\int_{\Omega_\tau} \rho r\, dv}^{\text{de cuerpo}} - \overbrace{\int_{d\Omega_\tau} \vec{\mathbf{q}} \cdot \hat{\mathbf{v}}\, da}^{\text{en frontera}}}_{\text{Calor}} \tag{4.36}$$

esta ecuación después de unas manipulaciones se puede expresar de forma local como

$$\rho \dot{u} - \boldsymbol{\sigma} : \mathbf{D} - \rho r + \overline{\nabla} \cdot \vec{\mathbf{q}} = 0 \tag{4.37}$$

la segunda ley de la termodinámica establece que para un proceso irreversible la entropía η aumenta y se cumple a nivel local la desigualdad de Clausius-Duhem como,

$$\dot{\eta} - \frac{r}{T} + \frac{1}{\rho}\overline{\nabla} \cdot (\frac{\vec{\mathbf{q}}}{T}) \geq 0 \tag{4.38}$$

para cumplir la admisibilidad física se toma el proceso como reversible y la ecuación de la primera ley y la desigualdad de la segunda ley que ahora es una igualdad dejan tres variables independientes : la temperatura T, la entropía η y el seudo esfuerzo $\mathbf{s}$, que expresada en la indeformada queda como,

$$\rho_0(T\dot{\eta} - \dot{u}) + \mathbf{s} : \dot{\mathbf{E}} = 0 \tag{4.39}$$

la ecuación anterior se debe cumplir localmente para todo punto en el dominio. Para satisfacer esta ecuación se puede hacer más simplificaciones, para que dependa solo de la temperatura T, en base a la energía libre $\psi = u - T\eta$, eliminando la energía interna u y teniendo en mente que $\psi = \psi(\vec{\mathbf{R}}, \mathbf{E}, T, \nabla T)$,

$$\rho_0(\dot{\psi} + \eta\dot{T}) + \mathbf{s} : \dot{\mathbf{E}} = 0 \tag{4.40}$$

usando los principios variacionales sobre la ecuación anterior, una variación en las variables independientes, $\mathbf{E}$, T y ∇T,

$$(\mathbf{s} - \rho_0 \frac{\partial \psi}{\partial \mathbf{E}}) : \delta\mathbf{E} - \rho_0(\eta + \frac{\psi}{\partial T})\delta T - \rho_0 \frac{\partial \psi}{\partial \nabla T} \cdot \delta \nabla T = 0 \tag{4.41}$$

conduce a la solución no trivial de variaciones:

$$\mathbf{s} = \rho_0 f \frac{\partial \psi}{\partial \mathbf{E}} \tag{4.42}$$

$$\eta = -\frac{\partial \psi}{\partial T} \tag{4.43}$$

$$\frac{\partial \psi}{\partial \nabla T} = 0 \tag{4.44}$$

si se toma el proceso isotérmico $\dot{T} = 0$ en la deformación, se define entonces *la función de densidad de energía de deformación* $W(\vec{\mathbf{R}}, \mathbf{E}) = \rho_0 \psi$, definiendo con esto ecuación constitutiva:

$$\mathbf{s} = \frac{\partial W}{\partial \mathbf{E}} \tag{4.45}$$

De forma similiar dejando la energía interna como variable independiente, simplificando a un proceso isoentrópico $\dot{\eta} = 0$, se llega a la expresión de $W(\vec{\mathbf{R}}, \mathbf{E}) = \rho_0 u$.

4.1.4 Material hiperelástico

El material hiperelástico puede ser caracterizado bajo un proceso isotérmico, asociando W con la energía libre $W = \rho_0 \psi$ o bien durante un proceso isoentŕopico asociando W con la energía interna $W = \rho_0 u$. La función W suele darse en función de los invariantes del tensor de deformación de Cauchy por derecha $\mathbf{C}$, el gradiente de deformaciones $\mathbf{F}$ o el tensor de deformación lagrangiana $\mathbf{E}$.
Los invariantes I_1, I_2 y I_3 del tensor $\mathbf{C}$ son:

$$I_1 = \mathrm{tr}\mathbf{C} \tag{4.46}$$
$$I_2 = \mathrm{tr}^2\mathbf{C} - \mathrm{tr}\mathbf{C}^2 \tag{4.47}$$
$$I_3 = \det \mathbf{C} \tag{4.48}$$

el tercer invariante cuando se considera constante $\det \mathbf{C} = 1$ no se tiene en cuenta y se tiene que el material es a su vez *incompresible*, $W = W(I_1, I_2)$.
Es común encontrar el segundo tensor de Piola-Kirchhoff $\mathbf{s}$ discriminado por los invariantes del tensor de Cauchy por derecha $\mathbf{C}$ por medio de la regla de la cadena como,

$$\mathbf{s} = \frac{\partial W}{\partial \mathbf{E}} = 2\left(\frac{\partial W}{\partial I_1}\mathbf{I} + \frac{\partial W}{\partial I_2}(I_1\mathbf{I} - \mathbf{C}) + \frac{\partial W}{\partial I_3} I_3 \mathbf{C}^{-1}\right) \tag{4.49}$$

sin embargo cuando se toma como incompresible el material, las características constitutivas se pueden representar también con el tensor de deformación lagrangiana $\mathbf{E}$ como,

$$\begin{aligned} \mathbf{s} &= 2\left(\left(\frac{\partial W}{\partial I_1} + 2\frac{\partial W}{\partial I_2}(1 - \mathbf{I} : \mathbf{E})\right)\mathbf{I} - 4\frac{\partial W}{\partial I_2}\mathbf{E}\right) \\ s_{ij} &= 2\left(\left(\frac{\partial W}{\partial I_1} + 2\frac{\partial W}{\partial I_2}(1 - E_{kk})\right)\delta_{ij} - 4\frac{\partial W}{\partial I_2}E_{ij}\right) \end{aligned} \tag{4.50}$$

en la expresión anterior se puede notar que la relación $\mathbf{s}(\mathbf{E})$ no es lineal, aunque es de primer orden para $\mathbf{s}(\mathbf{E})$ y para $\mathbf{s}(\vec{\mathbf{u}})$ se puede apreciar que es de segundo orden y tampoco es lineal. Existen algunos casos que se pueden linealizar como es el caso de los modelos generalizados de Hooke.
El tensor tangente de cuarto orden $\mathcal{C} = \frac{\partial \mathbf{s}}{\partial \mathbf{E}}$, es usado para linealizar el comportamiento sobre un punto de operación como es el caso del método de Newton-Rhapson para resolver los elementos finitos, detalle que se verá más adelante ya que es necesario implementar dichos métodos. Por esto para un material incompresible se toma como,

$$\begin{aligned} \mathcal{C} = \frac{\partial \mathbf{s}}{\partial \mathbf{E}} = 4\Bigg(&\left(\frac{\partial^2 W}{\partial I_1^2} + \frac{\partial W}{\partial I_2} + \frac{\partial^2 W}{\partial I_2^2}(1 + (2\mathbf{I} : \mathbf{E} + 3)^2)\right)\mathbf{I} \otimes \mathbf{I} - \frac{\partial W}{\partial I_2}\mathbf{I} \\ &+ 4\frac{\partial^2 W}{\partial^2 I_2}(\mathbf{I} : \mathbf{E} - 1)\,\mathbf{I} \otimes \mathbf{E} \\ &+ 4\frac{\partial^W}{\partial I_2^2}(\mathbf{I} : \mathbf{E} + 2)\mathbf{E} \otimes \mathbf{I} + 4\frac{\partial^2 W}{\partial I_2^2}\mathbf{E} \otimes \mathbf{E}\Bigg) \end{aligned} \tag{4.51}$$

en donde el tensor identidad de cuarto orden $\mathbf{I}$ tiene sus componentes en el polidiad canónico como $\delta_{ik}\delta_{jl}\hat{\mathbf{e}}_i \otimes \hat{\mathbf{e}}_j \otimes \hat{\mathbf{e}}_k \otimes \hat{\mathbf{e}}_l$.
Algunos modelos de hiperelasticidad presentan la siguiente condición $\frac{\partial^2 W}{\partial I_i^2} = 0$, esto facilita el cálculo computacional de alguna manera puesto que $\mathcal{C}$ ya no dende del desplazamiento $\vec{\mathbf{u}}$ y con esto la ecuación anterior se ve reducida a:

$$\mathcal{C} = \frac{\partial \mathbf{s}}{\partial \mathbf{E}} = 4\frac{\partial W}{\partial I_2}\left(\mathbf{I}\otimes\mathbf{I} - \mathbf{I}\right) \tag{4.52}$$

Algunos modelos de material

Existen varios modelos de materiales hiperelásticos, de los más usados está el modelo de Mooney-Rivlin que define la función de densidad de deformación de energía W en función de los invariantes de $\mathbf{C}$,

$$W = b_1(I_1 - 3) + b_2(I_2 - 3) \tag{4.53}$$

donde b_1 y b_2 son constantes del material. Cuando se toma $b_2 = 0$ se tiene el llamado material Neo-Hooke. Las siguientes relaciones son para este material,

$$\mathbf{s} = 2\left(\left(b_1 + 2b_2\left(1 - \mathbf{I} : \mathbf{E}\right)\right)\mathbf{I} - 4b_2\mathbf{E}\right) \tag{4.54}$$

$$\mathcal{C} = 4b_2\left(\mathbf{I}\otimes\mathbf{I} - \mathbf{I}\right) \tag{4.55}$$

El modelo de Yeoh para elastómeros incompresibles también es implementado, su función de densidad de energía es

$$W = \sum_{i=1}^{3} C_i(I_1 - 3)^i \tag{4.56}$$

donde C_i son coeficientes del material que se ajustan de forma experimental en base a un ensayo de tensión, por lo genreal $2C_1 = \mu$.

Chapter 5

Implementación en FEM y BEM de la elasticidad lineal. Comparación de costos computacionales

La frontera y su interior.

5.1 Introducción

En las referencias de [45, 51] que se encuentran y reportan muchas pruebas usando diferentes técnicas numéricas como *elementos finitos* FEM, *boundary elements* BEM, *colocación puntual de esferas* PCMFS y otros más, para solucionar modelos elásticos de tejidos a una velocidad tal que se logre simular la interacción de estos órganos en tiempo real. En algunos trabajos se han utilizado diferentes esquemas para resolver los métodos númericos con el fin de incrementar la capacidad de solución en las máquinas actuales.

La estructura de este capítulo está propuesta de la siguiente manera, (1) se menciona las ecuaciones de la elasticidad utilizadas bajo comportamiento lineal, (2) planteamiento, solución y un ejemplo de la solución de la ecuación de la elasticidad usando FEM, (3) planteamiento, solución y el mismo ejemplo usando BEM y (4) se comparan los resultados.

Como las ecuaciones de elasticidad son muy extensas, se usará las notaciones más compactas encontradas en los libros de elasticidad [44]. La primera notación y más conocida es la vectorial, la segunda es la *notación indicial o de Einstein* que aunque no es tan conocida, los lectores que hayan trabajado con matemática tensorial si estarán familiarizados (ver capítulo 4).

5.2 Ecuación de Elasticidad

Esta ecuación sintetiza el comportaminto de un sólido partiendo del comportamiento de un elemento infinitecimal inmerso en un continuo de materia como se muestra a continuación. Las variables que describen el comportamiento de un sólido son: esfuerzos, deformaciones, desplazamientos y tensiones [8, 44, 49, 4].

Las variables esfuerzo σ_{ij} y deformación unitaria ϵ_{ij} son tensores de segundo orden mientras que los

desplazamientos u_i y las tracciones t_i son tensores de primer orden o vectores.

Los esfuerzos que se generan sobre el elemento lo obligan a despalzarse y/o deformarse de su posición inicial, las ecuaciones generales que describen los efectos son (1) ecuaciones de equilibrio5.1, (2) ecuaciones de constitucin5.3 y (3) ecuaciones cinemáticas5.5.

5.2.1 Ecuación de equilibrio

Cada elemento diferencial que forme parte del sólido deberá estar en equilibrio, esto es, igualar la ecuación dinámica a cero,

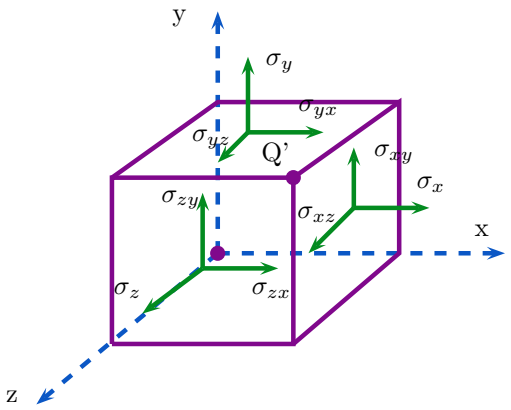


Figure 5.1: Tensor de esfuerzo

$$\sigma_{ij,j} + b_i = 0 \tag{5.1}$$

la notación vectorial de la ecuación anterior es,

$$\nabla \cdot \sigma + B = 0 \tag{5.2}$$

donde B es son las fuerzas externas de campo debidas a un campo gravitacional y/o electromagnético.

Hay que tener en cuenta que la ecuación de equilibrio en este caso es una condición que asegura el equilibrio, si el análisis fuera dinámico esta ecuación sería diferente de cero. Los ejemplos de este capítulo son para casos cuasi-estáticos en los cuales se puede asumir que la ecuación de equilibrio es igual a cero.

5.2.2 Ecuación de Constitución

Esta ecuación de constitución es la misma ecuación de la ley de Hooke Generalizada, en la cual se relacionan los esfuerzos con las deformaciones por medio del módulo de Poisson ν y el módulo de Young E que son propiedades intrínsecas del material, este modelo es también conocido como de Saint Venant-Kirchhoff.

$$\sigma_{ij} = \lambda \delta_{ij} \epsilon_{kk} + 2G\epsilon_{ij} \tag{5.3}$$

donde λ y G son los coeficientes de Lammé, dependen E y ν. En notación vectorial es,

$$\sigma = \lambda \operatorname{tr}(\epsilon) I + 2G\epsilon \tag{5.4}$$

en donde $\operatorname{tr}(\epsilon)$ es la traza del tensor ϵ, que no es más que la deformación volumétrica.

5.2.3 Ecuación Cinemática

Esta ecuación relaciona los despalzamientos u_i del sólido con las deformanciones unitarias ϵ_{ij} que el mismo sufre, este tensor es el tensor de ingeniería, en el cual las deformaciones son pequeñas y no se tiene en cuenta los efectos no lineales de segundo orden generados en el tensor de Cauchy-Green.

$$\epsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right) \tag{5.5}$$

en notación vectorial,

$$\epsilon = \frac{1}{2}\left(\nabla u + \nabla u^T\right) \tag{5.6}$$

5.2.4 Ecuación Diferencial de la Elasticidad Lineal

Sustituyendo la ecuación cinemática (eq. 5.5) en la ecuaciń de constitución (eg. 5.3) y luego está, en la ecuación de condición de equilibrio (eq. 5.1), se tiene,

$$0 = (\lambda + G)\, u_{j,ji} + G\, u_{i,jj} + b_i \tag{5.7a}$$

$$0 = (\lambda + G)\, \nabla\left(\nabla \cdot u\right) + G\, \nabla^2 u + B \tag{5.7b}$$

la ecuación anterior está sometida a unas condiciones de contorno, esto quiere decir que el sólido en análisis puede estar sometido a desplazamientos conocidos $\bar{u_i}$ o tracciones conocidas $\bar{t_i}$ sobre su superficie. Las partes de la superfice Γ que tiene como condiciones desplazamientos se denotan Γ_u mientras que las otras se denota por Γ_t.

5.3 Solución de PDEs sujetas a BVCs

La solución de ecuaciones diferenciales parciales (PDEs) sujetas a condiciones de contorno (BVCs), es un problema complejo matemáticamente, existen diversas técnicas análiticas para la solución de estas ecuaciones pero cuando se trata de satisfacer los BVCs, las soluciones analíticas se quedan bastante cortas, es aquí en donde entran los métodos numéricos. Dentro de los métodos computacionales más conocidos están las diferencias finitas (FDM) y los elementos finitos (FEM). Un método aproximádamente veinte años más reciente que los FEM es el método de los *boundary elements* (BEM) que conserva ciertas familiaridades con los FEM.

Un método númerico que se utiliza para la solución de las PDEs con BVCs es el método de los residuos ponderados y tanto los BEM como los FEM pueden ser vistos desde este punto de vista. Sin embargo existe otra forma de ver la formulación bajo el cálculo variacional, desde este punto de vista las leyes de conservación de energía funcionan a la perfección dandole un sentido más físico a la formulación.

La matemática de los residuos ponderados y el cálculo variacional puede ser relacionada junto con el método de las funciones de Green para solución de PDEs con BVCs. Los fundamentos de estos métodos fueron fuértemente establecidos por el *análisis funcional*, a comienzos y mediados del siglo pasado, y más bien hasta ahora conocidos y aplicados en ingeniería.

5.4 Elementos Finitos

El método de los FEM son una combinación del uso de: (1) el método de los residuos ponderados y (2) discretizar la región de análisis en pequeñas subreguiones, subregiones sobre las cuales existen unas (3) funciones definidas a trozos llamadas *funciones de forma*. Una subregión junto con su función de forma es un

elemento finito. El conjuto de todas las subregiones forman una malla en el dominio. La malla y todas las funciones de forma crean un espacio de aproximación, de la función que se desea encontrar en las PDEs (en el siguiente capítulo se profundiza en estas ideas).

A continuación se desarrolla un ejemplo sencillo el cual consta de una placa empotrada sobre el extremo izquierdo, sobre el extremo derecho se encuentra una distribución constate de fuerza hacia la derecha. Para este ejemplo se tomó el caso de esfuerzo plano.

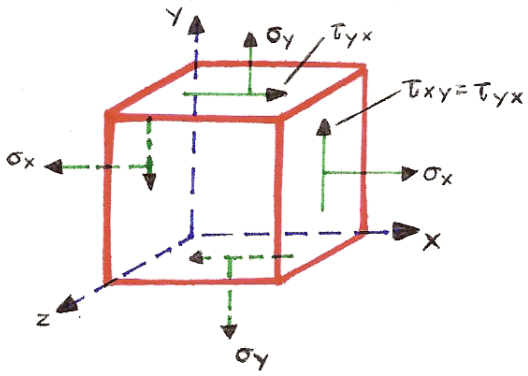


Figure 5.2: Idealización de un esfuerzo plano

5.4.1 Elemento Lineal Triangular

Para este ejemplo se utiliza el elemento triangular lineal de primer orden, este elemento relaciona las variables de la siguiente forma:

Constante de rigidez del elemento k: Como todo elemento finito, su aporte a todo el sistema se representa por la relación que existe entre los nodos (i, j, m) que comforman el elemento y los elementos que se ven afectados por estos nodos.

$$[k] = tA\,[B]^T\,[D]\,[B] \tag{5.8}$$

donde A es el área del elemento triangular, $[B]$ es un operador diferencial que relaciona el deplazamiento con la deformación dentro del elemento triangular (este operador es único para cada tipo de elemento finito), y $[D]$ es la matriz de constitución (es igual para todo tipo de elemento).

Matriz de Constitución D: Para el caso de esfuerzo plano la matriz se puede simplificar a,

$$[D] = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{5.9}$$

Matriz del Operador Diferencial B: Este operador diferencial depende de la función de forma que en este caso es un plano.

$$[B] = \frac{1}{2A}\begin{bmatrix} \beta_i & 0 & \beta_j & 0 & \beta_m & 0 \\ 0 & \gamma_i & 0 & \gamma_j & 0 & \gamma_m \\ \gamma_i & \beta_i & \gamma_j & \beta_j & \gamma_m & \beta_m \end{bmatrix} \tag{5.10}$$

donde p.e, $\beta_i = y_j - y_m$ y $\gamma_i = x_m - x_j$.

5.4.2 Solución y Posproceso

Después de generar la matriz de rigidez para cada elemento $[k]$ del mallado, se procede a sacar la matriz de rigidez del sistema global $[K]$ para obtener un sistema de ecuaciones de la forma

$$f = [K]\, u \tag{5.11}$$

para el posproceso los esfuerzos se calculan por medio de,

$$\sigma = [D]\,[B]\, u \tag{5.12}$$

5.4.3 Implementación en MATLAB

Primero, se toma como convención la representación de una malla, la cual tiene información acerca del número de elementos, el número de nodos que conforman el elemento y la información espacial de cada nodo. Por lo tanto, la malla se puede representar como un arreglo de n elementos que contienen en cada una de sus filas e, el número de cada nodo (i, j, m) que conforma el elemento e. Un arreglo de nodos que contiene en cada fila p las coordenadas espaciales del nodo p.

La implementación de un elemento finito es fácil, se comienza por la matriz de rigidez del elemento, $[k]$,

```
function k = matrizRigidezElementoTriangular(E,nu,h,nodos)
% MATRIZRIGIDEZELEMENTO Calcula la matriz de rigides de un elemento a
%           partir de las propiedades del material E y nu, h es el espesor
%           del elemento y nodos es una matriz de 2x3 con la ubicacion de
%           los nodos como columnas.

% Area del elemento triangular
A = det([1 1 1;nodos])/2;
% Matriz de relacin cinemtica
dNi = nodos(:,2)-nodos(:,3);
dNj = nodos(:,3)-nodos(:,1);
dNk = nodos(:,1)-nodos(:,2);
B = 1/2/A*[ dNi(2)       0  dNj(2)       0  dNk(2)       0
                0 -dNi(1)       0 -dNj(1)       0 -dNk(1)
           -dNi(1)  dNi(2) -dNj(1)  dNj(2) -dNk(1)  dNk(2)];
% Matriz de constitucion
D = E/(1-nu^2)*[1 nu 0;nu 1 0;0 0 (1-nu)/2];
% Matriz de rigidez
k = h*A*transpose(B)*D*B;
```

mfiles/matrizRigidezElementoTriangular.m

luego se ensambla la matriz del elemento $[k_e]$ dentro de la matriz de rigidez de todo el sistema $[K]$ que depende del mallado previamente generado,

```
function K = ensamblarElementoTriangularEnK(K,elem,k)
% ENSAMBLARELEMENTOTRIANGULARENK Ensambla el elemento definido por los
%           nodos numero elem y representado por k en la matriz general del
%           sistema K.


% Efecto de los nodos sobre la matriz general del sistema
for i = 1:3
    ii = elem(i);
    for j = 1:3
        jj = elem(j);
        K((2*ii-1):2*ii,(2*jj-1):2*jj) = K((2*ii-1):2*ii,(2*jj-1):2*jj) + ...
                                         k((2*i-1):2*i,(2*j-1):2*j);
    end
end
```

mfiles/ensamblarElementoTriangularEnK.m

para producir el ensamble total del sistema se utiliza entonces,

```
function K = ensamblarMatrizRigidezSistema(nodos,elementos)
% ENSAMBLARMATRIZRIGIDEZSISTEMA ensambla a partir de una matriz nodos de
%           2xn los n nodos de una malla organizados por los elementos que
%           es una matriz de mx3 donde m es el numero de elementos de la
%           malla.

% numero de elementos
numElem = size(elementos);
% numero de nodos
numNods = size(nodos);
% inicializa la matriz de rigidez del sistema
K = sparse(zeros(numNods(2)*2));
% ensamblar la matriz con los m elementos
for i = 1:numElem(1)
    % Espesor, coeficiente de poison y modulo de elasticidad constantes
    ki = matrizRigidezElementoTriangular(210e6,0.3,0.025,...
                    nodos(:,elementos(i,:)));
    % Ensamblar el elemento ki en la matriz K
    K = ensamblarElementoTriangularEnK(K,elementos(i,:),ki);
end
```

mfiles/ensamblarMatrizRigidezSistema.m

la función anterior es la que toma todo el costo computacional. Esta requiere de memoria como de tiempo para ensamblar el sistema a solucionar, en el caso de elasticidad lineal esto se calcula una vez como preproceso, en el caso no lineal, este es un permanente cálculo.

Como posproceso se tiene que calcular esfuerzos y deformaciones, lo cual se hace con,

```
function [stress,strain] = calcularEsfuerzosDeformacionesElemento(E,nu,nodos,u)
% CALCULARESFUERZOSDEFORMACIONESELEMENTO toma la informacin de un solo
%           elemento, necesaria para calcular los esfuerzos y
%           deformaciones.

% Area del elemento triangular
A = det([1 1 1;nodos])/2;
% Matriz de relacin cinemtica
dNi = nodos(:,2)-nodos(:,3);
dNj = nodos(:,3)-nodos(:,1);
dNk = nodos(:,1)-nodos(:,2);
B = 1/2/A*[ dNi(2)       0  dNj(2)       0  dNk(2)       0
                0 -dNi(1)       0 -dNj(1)       0 -dNk(1)
           -dNi(1)  dNi(2) -dNj(1)  dNj(2) -dNk(1)  dNk(2)];
% Matriz de constitucion
D = E/(1-nu^2)*[1 nu 0;nu 1 0;0 0 (1-nu)/2];
% Calcular las deformaciones unitarias en el elemento
strain = B*u;
% Calcular los esfuerzos en el elemento
stress = D*B*u;
```

mfiles/calcularEsfuerzosDeformacionesElemento.m

```
function [sigma,strain] = calcularEsfuerzosDeformaciones(K,F,U,elem,nodos)
% CALCULARESFUERZOSDEFORMACIONES  calcula las deformaciones STRAIN y
%           esfuerzos SIGMA de todo el sistema que representa la malla dada
%           por ELEM y NODOS.

% Tamao de la malla
numElem = size(elem);
% Inicaliza variables
sigma = zeros(numElem(1),4);
strain = zeros(numElem(1),3);
% Calcula elemento por elemento los esfuerzos y deformaciones de toda la
% malla
for i = 1:numElem
    u = reshape([2*elem(i,:)-1;2*elem(i,:)],[],1);
```

```
15      [sigma(i,1:3),strain(i,:)] = calcularEsfuerzosDeformacionesElemento(...
16                          210e6,0.3,...
17                          nodos(:,elem(i,:)),U(u));
18  end
```

mfiles/calcularEsfuerzosDeformaciones.m

5.4.4 Solución del problema

Primero se malla el sólido con 400 elementos triangulares, la línea roja representa el extremo que se encuentra totalmente empotrado y el cuadrado verde representa la distribución de $3000kN/m^2$ figura5.4.4. con la

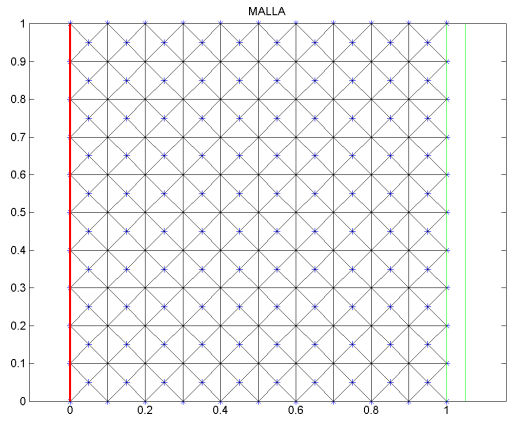


Figure 5.3: Malla del sólido con elementos triangulares

malla ya generada se corren los códigos de la subsección anterior, se halla la solución y se reliza el posproceso.

La matriz de rigidez $[K]$ es una matriz con baja población lo cual se puede apreciar en la siguiente figura 5.4.4, donde K es una matriz de (442×442) pero solo 5198 elementos son diferentes de cero, lo que quiere decir que solo el 2.66% de K tiene un valor significativo.

Una vez resuelto el sistema se puede visualizar de forma exagerada el sólido deformado (fig.5.4.4), los esfuerzos cortantes y normales (fig.5.4.4). Obtenida la matriz K, solo es necesario definir las condiciones de carga y restricciones para solucionar otro tipo de problemas. Como desventaja se tiene que la matriz K requiere un tratamiento especial para ser almacenada con el fin de que no ocupe demasiado espacio. Sin embargo, cuando se desea encontrar otra solución manipular esta matriz esparcida suele consumir más tiempo para multiplicaciones y sumas, se requiere de rutinas especiales.

5.5 Boundary Elements

Para la formulación de los elementos de contorno (**BEM**), se parte de la ecuación diferencial parcial que gobierna el equilibrio de un medio elástico, $\nabla \cdot \boldsymbol{\sigma} + \vec{\mathbf{f}} = 0$, que a su vez es la que define un operador lineal $\mathcal{L}$ sobre una función de desplazamiento $\vec{\mathbf{u}}$, $\mathcal{L} : \mathbb{R}^3 \ni \vec{\mathbf{u}}(\vec{\mathbf{x}}) \mapsto \vec{\mathbf{p}}(\vec{\mathbf{x}}) = \mathcal{L}(\vec{\mathbf{u}})$. Para comenzar se plantea con el

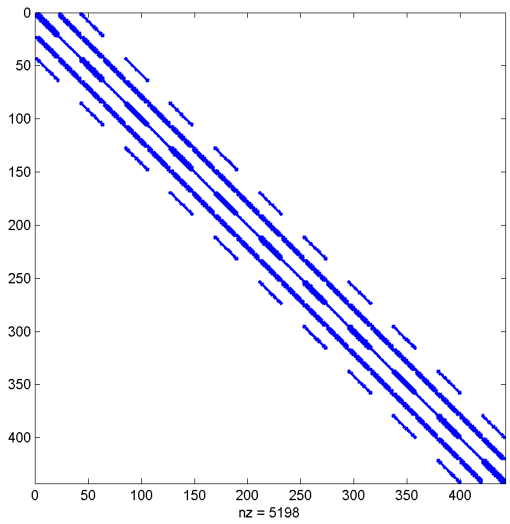


Figure 5.4: Matriz de Riguidez del Sistema K

principio de trabajos virtuales partiendo de la condición de equilibrio en el dominio Ω como sigue:

$$\begin{aligned}
\delta\mathbf{W}(\vec{\mathbf{u}}, \delta\vec{\mathbf{u}}) &= \int_\Omega \vec{\mathbf{f}} \cdot \delta\vec{\mathbf{u}}\, dv + \int_\Omega \boldsymbol{\sigma} : \delta\mathbf{E}\, dv - \int_\Gamma \vec{\mathbf{t}} \cdot \delta\vec{\mathbf{u}}\, da \\
&= -\int_\Omega (\nabla \cdot \boldsymbol{\sigma}) \cdot \delta\vec{\mathbf{u}}\, dv + \int_\Omega \boldsymbol{\sigma} : \delta\mathbf{E}\, dv - \int_\Gamma \vec{\mathbf{t}} \cdot \delta\vec{\mathbf{u}}\, da \\
&= \underbrace{\int_\Omega \boldsymbol{\sigma} : \delta\mathbf{E}\, dv}_{\delta\mathbf{W}^u_{int}} - \underbrace{\int_\Omega (\nabla \cdot \boldsymbol{\sigma}) \cdot \delta\vec{\mathbf{u}}\, dv - \int_\Gamma \vec{\mathbf{t}} \cdot \delta\vec{\mathbf{u}}\, da}_{\delta\mathbf{W}^u_{ext}} = 0
\end{aligned} \tag{5.13}$$

en donde $\delta\vec{\mathbf{u}}$ son los *desplazamientos virtuales* y estos forman el trabajo virtual de los desplazamientos virtuales. La primera integral en 5.13 es el trabajo interno $\delta\mathbf{W}^u_{int}$ y las dos últimas integrales el trabajo externo $\delta\mathbf{W}^u_{ext}$. Al intercambiar el orden de la variación funcional $\delta\mathbf{W}$ se obtiene:

$$\delta\mathbf{W}(\delta\vec{\mathbf{u}}, \vec{\mathbf{u}}) = \underbrace{\int_\Omega \delta\boldsymbol{\sigma} : \mathbf{E}\, dv}_{\delta\mathbf{W}^t_{int}} - \underbrace{\int_\Omega (\nabla \cdot \delta\boldsymbol{\sigma}) \cdot \vec{\mathbf{u}}\, dv - \int_\Gamma \delta\vec{\mathbf{t}} \cdot \vec{\mathbf{u}}\, da}_{\delta\mathbf{W}^t_{ext}} = 0 \tag{5.14}$$

en donde $\delta\vec{\mathbf{t}}$ son las *fuerzas virtuales* que generan el trabajo virtual de las fuerzas virtuales.

El siguiente paso toma en cuenta el *principio de Betti*, que consiste en igualar los trabajos internos de dos conjuntos de cargas (fuerzas o momentos) aplicadas en diferente orden, $\delta\mathbf{W}^u_{int} = \delta\mathbf{W}^t_{int}$, y notar que esos trabajos son recíprocos pues no importa el orden de aplicación. Además, en el caso lineal, donde el tensor de deformación y la relación constitutiva son lineales, es válido decir que los productos internos cumplen

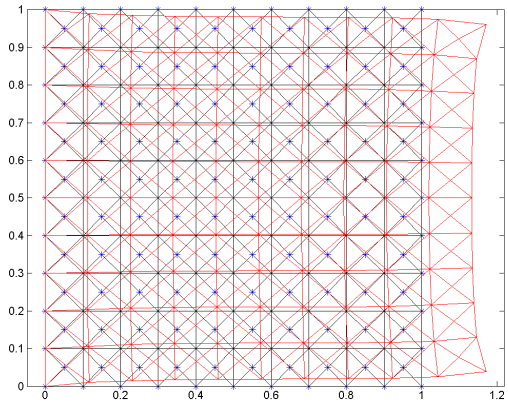
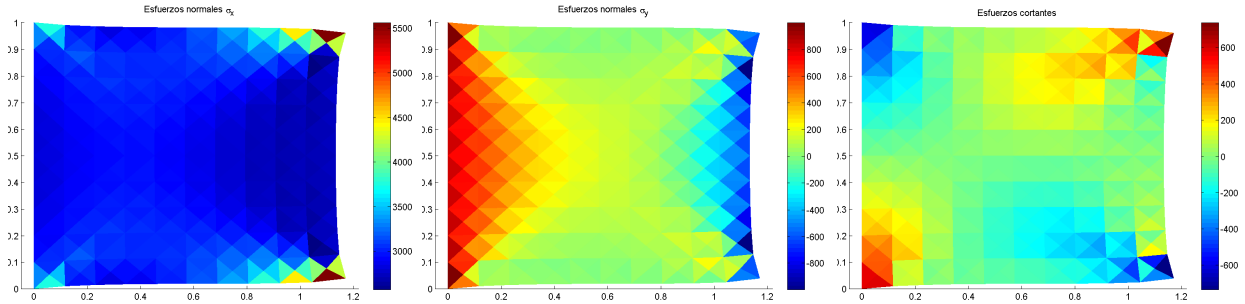


Figure 5.5: Sólido deformado a causa de la distribución de carga



a) b) c)

Figure 5.6: Postproceso cálculo de esferzos

$\int \boldsymbol{\sigma} : \delta\mathbf{E}\, dv = \int \delta\boldsymbol{\sigma} : \mathbf{E}\, dv$. Aplicando el principio de Betti se tiene,

$$\begin{aligned}
\mathcal{B}(\vec{\mathbf{u}}, \delta\vec{\mathbf{u}}) &= \delta\mathcal{W}(\vec{\mathbf{u}}, \delta\vec{\mathbf{u}}) - \delta\mathcal{W}(\delta\vec{\mathbf{u}}, \vec{\mathbf{u}}) \\
\mathcal{B}(\vec{\mathbf{u}}, \delta\vec{\mathbf{u}}) &= -\int_\Omega (\nabla \cdot \boldsymbol{\sigma}) \cdot \delta\vec{\mathbf{u}}\, dv + \int_\Omega \boldsymbol{\sigma} : \delta\mathbf{E}\, dv - \int_\Gamma \vec{\mathbf{t}} \cdot\, \delta\vec{\mathbf{u}}\, da \\
&\quad + \int_\Omega (\nabla \cdot \delta\boldsymbol{\sigma}) \cdot \vec{\mathbf{u}}\, dv - \int_\Omega \delta\boldsymbol{\sigma} : \mathbf{E}\, dv + \int_\Gamma \delta\vec{\mathbf{t}} \cdot\, \vec{\mathbf{u}}\, da \\
&= \int_\Omega (\nabla \cdot \delta\boldsymbol{\sigma}) \cdot \vec{\mathbf{u}}\, dv - \int_\Omega (\nabla \cdot \boldsymbol{\sigma}) \cdot \delta\vec{\mathbf{u}}\, dv + \int_\Gamma \delta\vec{\mathbf{t}} \cdot\, \vec{\mathbf{u}}\, da - \int_\Gamma \vec{\mathbf{t}} \cdot\, \delta\vec{\mathbf{u}}\, da = 0
\end{aligned} \tag{5.15}$$

para la ecuación 5.15 es necesario definir quien será $\vec{\mathbf{u}}$ y $\delta\vec{\mathbf{u}}$. Para esto se sabe que la función $\vec{\mathbf{u}}$ es la función incógnita que obedece a: la condición de dominio ($\nabla \cdot \boldsymbol{\sigma} + \vec{\mathbf{f}} = 0$ en $\vec{\mathbf{x}} \in \Omega$) y las condiciones de contorno de Dirchlet ($\vec{\mathbf{u}}(\vec{\mathbf{x}}) = \vec{\mathbf{u}}^*$ en $\vec{\mathbf{x}} \in \Gamma_u$) y Neumann ($\boldsymbol{\sigma} \cdot \hat{\mathbf{n}} = \vec{\mathbf{t}}^*$ en $\vec{\mathbf{x}} \in \Gamma_n$). De forma que se define en relación de la fuerza de cuerpo $\vec{\mathbf{f}}$ y el operador diferencial $\mathcal{L}$, el equilibrio como:

$$\mathcal{L}(\vec{\mathbf{u}}) = \nabla \cdot \boldsymbol{\sigma}(\vec{\mathbf{u}}) = -\vec{\mathbf{f}}(\vec{\mathbf{x}}) \tag{5.16}$$

Ahora por definición, aprovechando las ventajas de la función *delta de Dirac* $\delta_n(\vec{\mathbf{x}} - \vec{\mathbf{y}})$ y sabiendo que $\delta\vec{\mathbf{u}}$ es una función arbitraria cualquiera sobre las integrales de dominio, se define *la solución fundamental* $\delta\vec{\mathbf{u}}(\vec{\mathbf{x}}, \vec{\mathbf{y}})$

tal que:

$$\mathcal{L}(\delta\vec{\mathbf{u}}) \;=\; \nabla\cdot\delta\boldsymbol{\sigma} \;=\; \nabla\cdot\boldsymbol{\sigma}(\delta\vec{\mathbf{u}}) \;=\; -d\delta_n(\vec{\mathbf{x}}-\vec{\mathbf{y}})\hat{\mathbf{e}} \tag{5.17}$$

donde d es una constante arbitraria y $\hat{\mathbf{e}}$ es el vector unitario en dirección arbitraria. Acá la ecuación 5.17 no tiene condiciones de contorno pues representa un cuerpo elástico infinito donde la fuerza de cuerpo es cero en todo punto menos el punto arbitrario $\vec{\mathbf{y}}$, lugar donde existe una fuerza puntual unitaria de dirección $\hat{\mathbf{e}}$. La solución a esta ecuación es la *solución fundamental* $\delta\vec{\mathbf{u}}(\vec{\mathbf{x}},\vec{\mathbf{y}})$. Finalmente utilizando las ecuaciónes 5.16, 5.17 en 5.15 se puede encontrar la solución fundamental de los **BEM** para la elasticidad lineal como,

$$\mathcal{B}(\vec{\mathbf{u}},\delta\vec{\mathbf{u}}) \;=\; -\int_\Omega \delta_n(\vec{\mathbf{x}}-\vec{\mathbf{y}})\hat{\mathbf{e}}\cdot\vec{\mathbf{u}}(\vec{\mathbf{x}})\,dv \;+\; \int_\Omega \vec{\mathbf{f}}\cdot\delta\vec{\mathbf{u}}(\vec{\mathbf{x}},\vec{\mathbf{y}})\,dv \;+\; \int_\Gamma \delta\vec{\mathbf{t}}(\vec{\mathbf{x}},\vec{\mathbf{y}})\cdot\;\vec{\mathbf{u}}\,da - \int_\Gamma \vec{\mathbf{t}}\cdot\;\delta\vec{\mathbf{u}}(\vec{\mathbf{x}},\vec{\mathbf{y}})\,da \;=\; 0 \tag{5.18}$$

simplificando la función de Dirac y organizando,

$$c\,\hat{\mathbf{e}}\cdot\vec{\mathbf{u}}(\vec{\mathbf{y}}) \;+\; \int_\Gamma \delta\vec{\mathbf{t}}(\vec{\mathbf{x}},\vec{\mathbf{y}})\cdot\;\vec{\mathbf{u}}(\vec{\mathbf{x}})\,da \;=\; \int_\Gamma \vec{\mathbf{t}}(\vec{\mathbf{x}})\cdot\;\delta\vec{\mathbf{u}}(\vec{\mathbf{x}},\vec{\mathbf{y}})\,da + \int_\Omega \vec{\mathbf{f}}(\vec{\mathbf{x}})\cdot\delta\vec{\mathbf{u}}(\vec{\mathbf{x}},\vec{\mathbf{y}})\,dv \tag{5.19}$$

donde c es un escalar que depende de la ubicación de $\vec{\mathbf{y}}$ en el dominio Ω. Si el punto $\vec{\mathbf{y}}$ está al interior del dominio Ω, su valor es uno, si está por fuera del dominio es cero, pero si está justo encima del contorno su valor depende de si esta sobre un borde suave o si esta sobre alguna arista o vértice del dominio en cuyo caso el valor de c puede variar dependiendo del volumen que encierre la integral. En el caso de un borde suave el valor de $c = 1/2$. En el caso de un dominio 2D de forma cuadrada y que $\vec{\mathbf{y}}$ se encuentre sobre una esquina o vertice el valor de c es $1/4$. Si el dominio fuera un cubo en 3D y el punto de prueba estuviese sobre una esquina o vértices $c = 1/8$. Y si estuviese sobre una arista entonces $c = 1/4$.

En la ecuacion 5.19, las tres integrales se pueden ver como transformaciones de funciones, al igual que las transformadas de Laplace y Fourier, y por lo tanto a las funciones $\delta\vec{\mathbf{u}}(\vec{\mathbf{x}},\vec{\mathbf{y}})$ y $\delta\vec{\mathbf{t}}(\vec{\mathbf{x}},\vec{\mathbf{y}})$ se les llama *kernels integrales*, estos kernels se encuentran a continuación.

5.5.1 Solución Fundamental

La solución fundamental para el caso de hiperelasticidad lineal en el modelo de Saint Venant-Kirchhoff, sobre la condición de equilibrio en 5.17 y teniendo en cuenta que la relación esfuerzo-deformación para tal caso es $s_{ij} = \lambda E_{kk}\delta_{ij} + 2\mu E_{ij}$, lleva finalmente al operador lineal diferencial $\mathcal{L}$,

$$\begin{aligned}\mathcal{L}(u_i) \;&=\; (\lambda+\mu)u_{j,j,i} \;+\; \mu u_{i,jj}\\ \mathcal{L}(\vec{\mathbf{u_i}}) \;&=\; (\lambda+\mu)\nabla\nabla\cdot\vec{\mathbf{u}} \;+\; \mu\nabla^2\vec{\mathbf{u}}\end{aligned} \tag{5.20}$$

y generalizando ahora para modelos lineales el operador queda de la siguietne forma:

$$\mathcal{L}(u_i) \;=\; \alpha_1 u_{j,ji} \;+\; \alpha_2 u_{i,jj} \tag{5.21}$$

en donde α_1 y α_2 dependen del modelo elástico lineal utilizado. Se propone entonces la solución de Kevin, usando el vector de Galerkin $\vec{\mathbf{g}}$, que relaciona el campo de desplazamientos $\vec{\mathbf{u}}$ con el vector de Galerkin, mediante el operador diferencial $\mathcal{L}_g$ de la forma $u_i = \mathcal{L}_g(g_i) = \beta_1 g_{k,ik} + \beta_2 g_{i,kk}$, en donde β_1 y β_2 han de ser determinados en función de α_1 y α_2 con el fin de que el operador $\mathcal{L}$ se pueda transformar sobre $\vec{\mathbf{g}}$ como un operador bi-armónico, $\mathcal{L}\circ\mathcal{L}_g(\vec{\mathbf{g}}) = \nabla^4\vec{\mathbf{g}}$ (el símbolo $\circ$ denota la composición de los operadores, es decir, $\mathcal{L}(\mathcal{L}_g())$), con esto se llega a:

$$\begin{aligned}\mathcal{L}(u_i) \;&=\; \alpha_1 u_{j,ji} \;+\; \alpha_2 u_{i,jj} \;=\; 0\\ \mathcal{L}\circ\mathcal{L}_g(g_i) \;&=\; \alpha_1(\beta_1 g_{k,jk} \;+\; \beta_2 g_{j,kk})_{,ji} \;+\; \alpha_2(\beta_1 g_{k,ik} \;+\; \beta_2 g_{i,kk})_{,jj}\\ &=\; \alpha_1\beta_1 g_{k,jkji} \;+\; \alpha_1\beta_2 g_{j,kkji} \;+\; \alpha_2\beta_1 g_{k,ikjj} \;+\; \alpha_2\beta_2 g_{i,kkjj}\\ &=\; \underbrace{(\alpha_1\beta_1 \;+\; \alpha_1\beta_2 \;+\; \alpha_2\beta_1)}_{\text{si es }=0}\, g_{k,ikjj} \;+\; \underbrace{\alpha_2\beta_2}_{\text{si es }=1}\; g_{i,kkjj}\\ &=\; g_{i,kkjj} \;=\; 0\end{aligned} \tag{5.22}$$

Se concluye que se debe cumplir que, $\beta_1 = -\alpha_1/(\alpha_2(\alpha_1+\alpha_2))$ y $\beta_2 = 1/\alpha_2$. La ecuación 5.17 se puede plantear con el vector de Galerkin como $\nabla^4\vec{\mathbf{g}} + d\delta_n = 0$ y haciendo la sustitución $\vec{\mathbf{h}} = \nabla^2\vec{\mathbf{g}}$, queda como $\nabla^2\vec{\mathbf{h}} + d\delta_n = 0$, donde una de las muchas soluciones en 2D es en coordenadas polares, $\frac{1}{r}\frac{d}{dr}(r\frac{d\vec{\mathbf{h}}}{dr}) = \vec{\mathbf{0}}$,

$$\vec{\mathbf{h}}(\vec{\mathbf{x}},\vec{\mathbf{y}}) = -\frac{d}{2\pi}\ln r\,\hat{\mathbf{e}} \tag{5.23}$$

en donde $r = \|\vec{\mathbf{x}} - \vec{\mathbf{y}}\|$ es la distancia entre el punto de prueba $\vec{\mathbf{y}}$ y el punto de integración $\vec{\mathbf{x}}$, es posible resolver $\nabla^2\vec{\mathbf{g}} = \vec{\mathbf{h}}$ de la misma forma, usando coordenadas polares se tiene $\frac{1}{r}\frac{d}{dr}(r\frac{d\vec{\mathbf{h}}}{dr}) = -\frac{d}{2\pi}\ln r\,\hat{\mathbf{e}}$,

$$\vec{\mathbf{g}}(\vec{\mathbf{x}},\vec{\mathbf{y}}) = -\frac{d}{8\pi}r^2\ln r\,\hat{\mathbf{e}} \tag{5.24}$$

5.5.2 Kernels de desplazamiento y tracción

Identificado el vector de Galerkin, ahora se usa el operador $\mathcal{L}_g$ para encontrar el kernel de desplazamiento y sabiendo de antemano que $r_{,ij} = (\delta_{ij} - r_{,i}r_{,j})/r$, que $r_{,k}r_{,k} = 1$, que $(r^2\ln r)_{,ij} = (2\ln r + 1)\delta_{ij} + 2\,r_{,i}r_{,j}$ y quitando cualquier término constante que no dependa de r,

$$\begin{aligned}
u_i(\vec{\mathbf{x}},\vec{\mathbf{y}}) &= \beta_1(-\frac{d}{8\pi}r^2\ln r\,e_k)_{,ki} + \beta_2(-\frac{d}{8\pi}r^2\ln r\,e_i)_{,kk}\\
&= -\frac{d}{8\pi}\left(\beta_1(r^2\ln r)_{,ki}\delta_{kj} + \beta_2(r^2\ln r)_{,kk}\delta_{ij}\right)e_j\\
&= -\frac{d}{8\pi}\left(\beta_1((2\ln r + 1)\delta_{ij} + 2\,r_{,i}r_{,j}) + \beta_2(4\ln r + 4)\delta_{ij}\right)e_j\\
&= -\frac{d}{8\pi}\left((\beta_1 + 4\beta_2 + 2\ln r(\beta_1 + 2\beta_2))\delta_{ij} + 2\beta_1 r_{,i}r_{,j}\right)e_j\\
&= \frac{d}{4\pi}\left((\beta_1 + 2\beta_2)\ln\frac{1}{r}\,\delta_{ij} - \beta_1 r_{,i}r_{,j}\right)e_j
\end{aligned} \tag{5.25}$$

el kernel de desplazamiento U_{ij} queda como,

$$U_{ij}(\vec{\mathbf{x}},\vec{\mathbf{y}}) = \frac{d}{4\pi}\left((\beta_1 + 2\beta_2)\ln\frac{1}{r}\,\delta_{ij} - \beta_1 r_{,i}r_{,j}\right) \tag{5.26}$$

Falta por determinar el kernel de tracción T_{ij},

$$\begin{aligned}
t_i &= \sigma_{ij}n_j\\
&= C_{ijkl}\,E_{kl}\,n_j = \frac{1}{2}C_{ijkl}\,(u_{k,l} + u_{l,k})\,n_j = \frac{1}{2}C_{ijkl}\,(U_{kn,l} + U_{ln,k})\,e_n\,n_j\\
&= \frac{d}{8\pi}C_{ijkl}\,(\,(\beta_1 + 2\beta_2)\,(\ln\frac{1}{r})_{,l}\,\delta_{kn} - \beta_1(r_{,k}r_{,n})_{,l} + (\beta_1 + 2\beta_2)\,(\ln\frac{1}{r})_{,k}\,\delta_{ln} - \beta_1(r_{,l}r_{,n})_{,k})\,e_n\,n_j\\
&= \frac{d}{8\pi}C_{ijkl}\left((\beta_1 + 2\beta_2)\left((\ln\frac{1}{r})_{,l}\,\delta_{kn} + (\ln\frac{1}{r})_{,k}\,\delta_{ln}\right) - \beta_1\left((r_{,k}r_{,n})_{,l} + (r_{,l}r_{,n})_{,k}\right)\right)e_n\,n_j\\
&= \frac{d}{8\pi}C_{ijkl}\left((\beta_1 + 2\beta_2)\left(\frac{-r_{,l}}{r}\,\delta_{kn} + \frac{-r_{,k}}{r}\,\delta_{ln}\right) - \beta_1\left(2\,r_{,kl}r_{,n} + r_{,k}r_{,nl} + r_{,l}r_{,nk}\right)\right)e_n\,n_j\\
&= \frac{d}{8\pi r}C_{ijkl}\,(\,(\beta_1 + 2\beta_2)\,(-r_{,l}\,\delta_{kn} - r_{,k}\,\delta_{ln}) - \beta_1\,(2\,\delta_{kl}\,r_{,n} + r_{,k}\,\delta_{nl} + r_{,l}\,\delta_{nk} - 4\,r_{,l}\,r_{,n}\,r_{,k}))\,e_n\,n_j\\
&= \frac{d}{4\pi r}C_{ijkl}\,(\,(\beta_1 + \beta_2)\,(-r_{,l}\,\delta_{kn} - r_{,k}\,\delta_{ln}) - \beta_1\,(\,\delta_{kl}\,r_{,n} - 2\,r_{,l}\,r_{,n}\,r_{,k}))\,e_n\,n_j\\
&= \frac{d}{4\pi}\left((\beta_1 + 2\beta_2)\ln\frac{1}{r}\,\delta_{ij} - \beta_1 r_{,i}r_{,j}\right)
\end{aligned} \tag{5.27}$$

5.5.3 Formulación del elemento

En resumen este método se puede ver como un caso especial de los FEM si se toma la ecuación de los residuos ponderados desde la *forma débil*, el principio de Betti, y se vuelve a aplicar la segunda identidad de Green de la integral por partes para dos soluciones de la ecuación de dominio, se tiene que el método de los BEM quedará reducido a una ecuación integral de contorno (**BIE**), que satisface las condiciones de contorno. En el caso de los BEM la funcón de ponderación del error es llamada *kernel* y físicamente tiene como significado, la solución de la ecuación diferencial a resolver pero con la diferencia de que no tiene condiciones de contorno. Es decir, para el caso de la elasticidad el kernel puede ser la solución de un sólido infinito en el cual se conoce el punto de aplicación p de una fuerza infinita sobre una area cero, entonces la solución es el desplazamiento causado por dicha fuerza infinita en el punto Q que puede ser cualquiera.

La BIE general de la elasticidad relaciona los deplazamientos u^* y tracciones t^* generados por los kernels y los despalzamientos u y tracciones t generados por el sistema a solucionar, esto es,

$$u_i\left(p\right) \, + \, \int_{\Gamma} T_{ij}\left(p,Q\right) \, u_i\left(Q\right) \, d\Gamma \, = \, \int_{\Gamma} U_{ij}\left(p,Q\right) \, t_i\left(Q\right) \, d\Gamma \tag{5.28}$$

donde los kernels son presentados como tensores de desplazamiento U_{ij} y esfuerzo T_{ij}. Para solucionar la ecuación anterior se tiene que discretizar el contorno Γ en subelementos Γ_e y obetener un sistema de ecuaciones parecido al que se tiene en los FEM,

$$c_i \, u_i\left(p\right) \, + \, \sum_{i=1}^{n} \int_{\Gamma_e} T_{ij}\left(p,Q\right) d\Gamma \, \cdot \, u_i\left(Q\right) \, = \, \sum_{i=1}^{n} \int_{\Gamma_e} U_{ij}\left(p,Q\right) d\Gamma \, \cdot \, t_i\left(Q\right) \tag{5.29}$$

donde c_i depende de la ubicación de p, si esta en un punto interior del sólido o se encuentra sobre el borde del sólido. En realidad c_i son los coeficientes que al final ajustan las BVCs de la PDE. La ecuación anterior se puede expresar como un sistema de ecuacines de la forma,

$$\left[H\right] \cdot u \, = \, \left[G\right] \cdot t \tag{5.30}$$

en donde,

$$H_{ij} = \begin{cases} c_i \, + \, \int_{\Gamma_e} T_{ij}\left(p,Q\right) d\Gamma, & \text{si } i = j \\ \int_{\Gamma_e} T_{ij}\left(p,Q\right) d\Gamma, & \text{si } i \neq j \end{cases} \tag{5.31}$$

$$G_{ij} = \int\limits_{\Gamma_e} U_{ij}\left(p,Q\right) d\Gamma \tag{5.32}$$

una vez armado el sistema anterior, con las BVCs se puede reorganizar el sistme dejando de un lado las condiciones conocidas y del otro las desconocidas $b \, = \, Ax$.

5.5.4 Elemento Constante de Contorno

Este elemento solo tiene un nodo sobre el cual se especifica el desplazamiento y la tracción. Un ejemplo de como se discretiza con este elemento se muestra a continuación (figura 5.5.4),

Para poder encontrar $[H]$ y $[G]$ es necesario calcular integrales de línea las cuales no son sencillas de integrar de forma analítica, por lo tanto lo que se usa generalmente es usar una cuadratura numeŕica con la de Gauss-Legendre de cuarto orden o superior. Cuando la integral es impropia se suele usar integración logartmica o a veces es posible encontrar una solución analítica (ver figura 5.5.4).

La integral es el efecto que tiene la carga aplicada en el punto p, sobre todo el elemento Γ_e (ver figura 5.5.4).

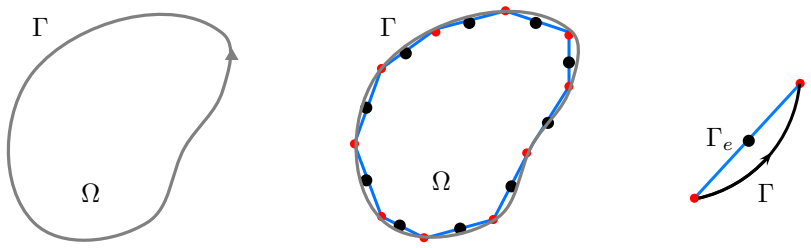


Figure 5.7: Sólido discretizado solo en la frontera para resolver por BEM

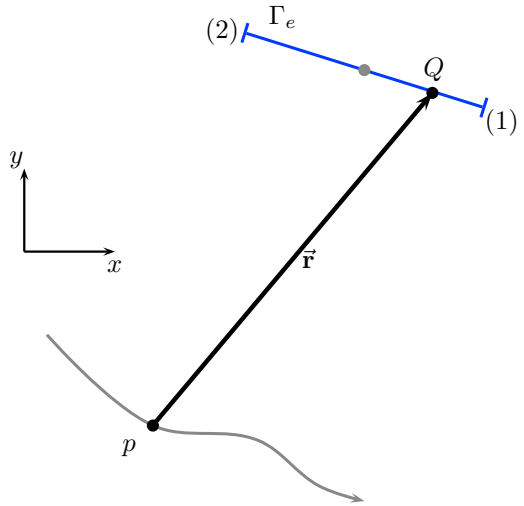


Figure 5.8: Representación de la integral de línea

5.5.5 Implementación en MATLAB

La estructura es la siguiente: funciones que calculan los kernels [51],

$$U_{ij} = \frac{1}{8\pi G(1-\nu)} \left[(3-4\nu) \ln\left(\frac{1}{r}\right) \delta_{ij} + r_{,i} r_{,j} \right] \tag{5.33}$$

$$T_{ij} = -\frac{1}{4\pi r(1-\nu)} \left[\frac{\partial r}{\partial n} \left((1-2\nu)\,\delta_{ij} + 2\,r_{,i}\,r_{,j} \right) - (1-2\nu)\,(r_{,j}\,n_i - r_{,i}\,n_j) \right] \tag{5.34}$$

```
function po = kernelTrac(p,Q,nu,n)
% KERNELTRAC calcula el tensor del kernel de traccion.

r = norm(Q - p);
dr_xy = (Q - p)/r; % Grad(r)
dr_n = dr_xy*transpose(n); % Grad(r)*n: derivada direccional de r en n
po = - 1/(4*pi*(1 - nu)*r)*...
        (dr_n*((1 - 2*nu)*eye(2) + 2*transpose(dr_xy)*dr_xy) -...
        (1 - 2*nu)*(transpose(dr_xy)*n - transpose(n)*dr_xy));
```

mfiles/kernelTrac.m

```
function uo = kernelDesp(p,Q,G,nu)
% KERNELDESP calcula el tensor que representa el kernel de deplazamiento

r = norm(Q - p);
dr_xy = (Q - p)/r;

uo = 1/(8*pi*G*(1-nu))*((3-4*nu)*log(1/r)*eye(2) + transpose(dr_xy)*dr_xy);
```

mfiles/kernelDesp.m

funciones que integran los kernels a lo largo de un elemento e por causa de una carga hipottica unitaria en un punto p, estas puede ser nmericas si es entre un punto p que no se encuentra dentro del elemento e o analticas (para el caso de los elementos constantes) para el caso en que p se encuentra entre los puntos del elemento e.

```
function [H,G] = integrar_Nodo_Elemento(p,nodosElem,normal,nu,Ge,cita,w,modo)
% INTEGRAR_NODO_ELEMENTO calcula la integral a lo largo de los nodos del
%           elemento NODOSELEM usando la cuadratura de Gauss-Legendre
%           definida por W y CITA.

longInter = (nodosElem(2,:) - nodosElem(1,:))*0.5;
medInter = (nodosElem(2,:) + nodosElem(1,:))*0.5;
n = length(w);
H = zeros(2);
G = H;
R = norm(longInter);
switch modo
    case 'numeric'
        for i = 1:n
            Q = longInter*cita(i) + medInter;
            H = H + w(i)*kernelTrac(p,Q,nu,normal)*R;
            G = G + w(i)*kernelDesp(p,Q,Ge,nu)*R;
        end
    case 'analytic'
        H = 0.5*eye(2);
        k = 0.25*R/pi/Ge/(1-nu);
        G = k*((3-4*nu)*(1-log(R))*eye(2) + transpose(longInter)*longInter/(R^2));
    otherwise
        error('Los unicos modos son: numeric y analytic')
end
```

mfiles/integrar_Nodo_Elemento.m

la función anterior ha de ser evaluada sobre cada uno de los elementos que conforman la malla de contorno con el fin de encontrar $[H]$ y $[G]$

```
function [H,G] = ensamblarMatriciesBIESistemaCons(nodos,elementos,normales,Ge,nu)

[I,w,cita] = gauss_q(@(x)1,-1,1,10);

n = length(elementos(:,1));
for i = 1:n
    for j = 1:n
        if i == j
            modo = 'analytic';
        else
            modo = 'numeric';
        end
        [H([2*i-1 2*i],[2*j-1 2*j]),G([2*i-1 2*i],[2*j-1 2*j])] =...
            integrar_Nodo_Elemento(...
                nodos(elementos(i,2),:),...
                nodos(elementos(j,[1,3]),:),normales(j,:),nu,Ge,cita,w,modo);
    end
end
```

mfiles/ensamblarMatriciesBIESistemaCons.m

5.6 Comparación entre BEM y FEM

Las soluciones son comparables en la siguiente figura 5.6, en donde se compara la deformación similar de ambos dominios, para un mallado pobre como el ejemplo del presente capítulo. Como se puede ver las

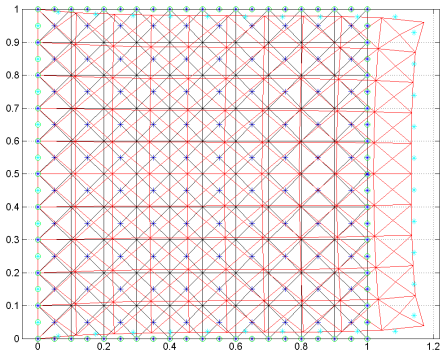


Figure 5.9: Comparación de resultados entre BEM y FEM

soluciones son semejantes aunque la matriz de solución con BEM es solo de (40×40), se puede observar tambien que solo es necesario discretizar el borde y si se desea conocer la solución en el dominio se debe realizar tantas integrales sobre todo el borde Γ con puntos que se necesiten al interior del dominio Ω. Esto complica el costo computacional en el método de los BEM en la etapa de posproceso, aunque a su vez esta característica sirve para evaluar grietas internas y fracturas, mediante integrales singulares.

Las matrices densas y no simétricas que se obtienen en los BEM se resuelven tambien por un método iterativo como ocurre con los FEM. Y al igual que los FEM basta con resolverlo una vez, esto es debido a que la linealidad del sistema permite simplemente tener que multiplicar una respuesta unitaria por un escalar. En definitiva no existe diferencia en el performance entre FEM y BEM debido a esto.

Por último el análisis lineal de los FEM y los BEM no es suficiente para el modelamiento de tejidos. La principal limitante es que las deformaciones son grandes y no se puede considerar el tensor de deformación unitaria de ingeniería. El siguiente capítulo hace el análisis de FEM para los modelos no lineales de geometría y material.

Chapter 6

Implementación FEM de la elasticidad no-lineal

Espacios de aproximación!!!

A continuación la solución por elementos finitos mediante un ejemplo estático [4, 8]:

6.1 Formulación del elemento finito

El problema básico es un dominio Ω_τ limitado por dos tipos de bordes: la frontera de Dirichlet Γ_u y la frontera Neumann Γ_n. Donde Γ_u se conoce también como la condición esencial o forzada y Γ_n se conoce como la condición normal o necesaria. Se intenta entonces satisfacer para toda configuración las siguientes ecuaciones:

$$\overline{\nabla} \cdot \boldsymbol{\sigma} + \vec{\mathbf{f}} = 0 \text{ en } \Omega_\tau \tag{6.1}$$

$$\vec{\mathbf{u}}^* - \vec{\mathbf{u}} = 0 \text{ en } \Gamma_u \tag{6.2}$$

$$\vec{\mathbf{t}}^* - \boldsymbol{\sigma} \cdot \vec{\mathbf{n}} = 0 \text{ en } \Gamma_n \tag{6.3}$$

El inconveniente de las condiciones anteriores es que desde el principio se desconoce la forma del dominio en la configuración deformada Ω_τ a excepción de la frontera Γ_u que si se conoce (ver figura6.1). Para comenzar se analiza utilizando los principios variacionales y se define la siguiente variación del funcional sobre el dominio:

$$\delta\mathcal{W}(\vec{\mathbf{u}}; \delta\vec{\mathbf{u}}) = \int_{\Omega_\tau} [\overline{\nabla} \cdot \boldsymbol{\sigma} + \vec{\mathbf{f}}] \cdot \delta\vec{\mathbf{u}}\, dv + \int_{\Gamma_n} [\vec{\mathbf{t}}^* - \boldsymbol{\sigma} \cdot \vec{\mathbf{n}}] \cdot \delta\vec{\mathbf{u}}\, da = 0 \tag{6.4}$$

que hará que se satisfagan las condiciones de frontera y dominio para cualquier $\delta\vec{\mathbf{u}}$ con excepción de los puntos sobre Γ_u para los cuales $\delta\vec{\mathbf{u}} = 0$. Simplificando después de unas manipulaciones se llega a:

$$\delta\mathcal{W}(\vec{\mathbf{u}}; \delta\vec{\mathbf{u}}) = \int_{\Omega_\tau} \boldsymbol{\sigma} : \overline{\nabla}\delta\vec{\mathbf{u}}\, dv + \int_{\Omega_\tau} \vec{\mathbf{f}} \cdot \delta\vec{\mathbf{u}}\, dv + \int_{\Gamma_n} \vec{\mathbf{t}}^* \cdot \delta\vec{\mathbf{u}}\, da = 0 \tag{6.5}$$

expresando ahora en términos de la indeformada

$$\delta\mathcal{W}(\vec{\mathbf{u}}; \delta\vec{\mathbf{u}}) = \int_{\Omega_0} \mathbf{s} \cdot \mathbf{F}^{\mathbf{T}} : \nabla\delta\vec{\mathbf{u}}\, dv + \int_{\Omega_0} \vec{\mathbf{f_0}} \cdot \delta\vec{\mathbf{u}}\, dv + \int_{\Gamma_{n_0}} \vec{\mathbf{t}}^{(\mathbf{N})*} \cdot \delta\vec{\mathbf{u}}\, da = 0 \tag{6.6}$$

la ecuación anterior debe resolverse para solucionar la deformacion en el dominio pero es una ecuación no lineal, la solución se encuentra usando Newton-Rhapson, pero antes será reorganizada tomando las ultimas integrales como el trabajo externo y dejándolas del lado derecho como $\mathcal{R}$,

$$\int_{\Omega_0} \mathbf{s} \cdot \mathbf{F}^{\mathbf{T}} : \nabla \delta \vec{\mathbf{v}} dv \; = \; \mathcal{R} \cdot \delta \vec{\mathbf{u}} \tag{6.7}$$

ahora se aproxima el lado izquierdo que contiene al gradiente de deformaciones $\mathbf{F}$, el segundo tensor de esfuerzos de Piola-Kirchhof $\mathbf{s}$ y definiendo $\mathcal{F}(\vec{\mathbf{u}}) \; = \; \mathbf{s}(\vec{\mathbf{u}}) \cdot \mathbf{F}^{\mathbf{T}}(\vec{\mathbf{u}})$ se llega a la siguiente aproximación de primero orden,

$$\begin{aligned} \mathcal{F}(\vec{\mathbf{u}} + \delta\vec{\mathbf{u}}) \; &= \; \mathcal{F}(\vec{\mathbf{u}}) \; + \; \delta\mathcal{F}(\vec{\mathbf{u}}; \delta\vec{\mathbf{u}}) \\ &= \; \mathbf{s}(\vec{\mathbf{u}}) \cdot \mathbf{F}^{\mathbf{T}}(\vec{\mathbf{u}}) \; + \; \mathbf{s}(\vec{\mathbf{u}}) \cdot \delta\mathbf{F}^{\mathbf{T}}(\vec{\mathbf{u}}; \delta\vec{\mathbf{u}}) \; + \; \delta\mathbf{s}(\vec{\mathbf{u}}; \delta\vec{\mathbf{u}}) \cdot \mathbf{F}^{\mathbf{T}}(\vec{\mathbf{u}}) \end{aligned} \tag{6.8}$$

usando la derivada direccional o de Gateaux se tienen los siguientes resultados,

$$\begin{aligned} \delta\mathbf{F}(\vec{\mathbf{u}}; \delta\vec{\mathbf{u}}) \; &= \; \lim_{\xi \to 0} \frac{d}{d\xi} \left(\mathbf{I} - \nabla^{\mathbf{T}}(\vec{\mathbf{u}} + \xi\delta\vec{\mathbf{u}}) \right) \\ &= \; \nabla^{\mathbf{T}} \delta\vec{\mathbf{u}} \end{aligned} \tag{6.9}$$

$$\begin{aligned} \delta\mathbf{E}(\vec{\mathbf{u}}; \delta\vec{\mathbf{u}}) \; &= \; \frac{1}{2} \lim_{\xi \to 0} \frac{d}{d\xi} \left((\mathbf{F} + \xi\delta\mathbf{F}(\vec{\mathbf{u}}; \delta\vec{\mathbf{u}}))^{\mathbf{T}} \cdot (\mathbf{F} + \xi\delta\mathbf{F}(\vec{\mathbf{u}}; \delta\vec{\mathbf{u}})) - \mathbf{I} \right) \\ &= \; \frac{1}{2} (\mathbf{F}^{\mathbf{T}} \cdot \delta\mathbf{F}(\vec{\mathbf{u}}; \delta\vec{\mathbf{u}}) + \delta\mathbf{F}^{\mathbf{T}}(\vec{\mathbf{u}}; \delta\vec{\mathbf{u}}) \cdot \mathbf{F}) \\ &= \; \mathbf{F}^{\mathbf{T}} \cdot \nabla^{\mathbf{T}} \delta\vec{\mathbf{u}} \; + \; \nabla\delta\vec{\mathbf{u}} \cdot \mathbf{F} \end{aligned} \tag{6.10}$$

$$\begin{aligned} \delta\mathbf{s}(\vec{\mathbf{u}}; \delta\vec{\mathbf{u}}) \; &= \; \frac{\partial \mathbf{s}}{\partial \mathbf{E}} : \delta\mathbf{E}(\vec{\mathbf{u}}; \delta\vec{\mathbf{u}}) \\ &= \; \frac{\partial \mathbf{s}}{\partial \mathbf{E}} : \mathbf{F}^{\mathbf{T}} \cdot \nabla^{\mathbf{T}} \delta\vec{\mathbf{u}} \end{aligned} \tag{6.11}$$

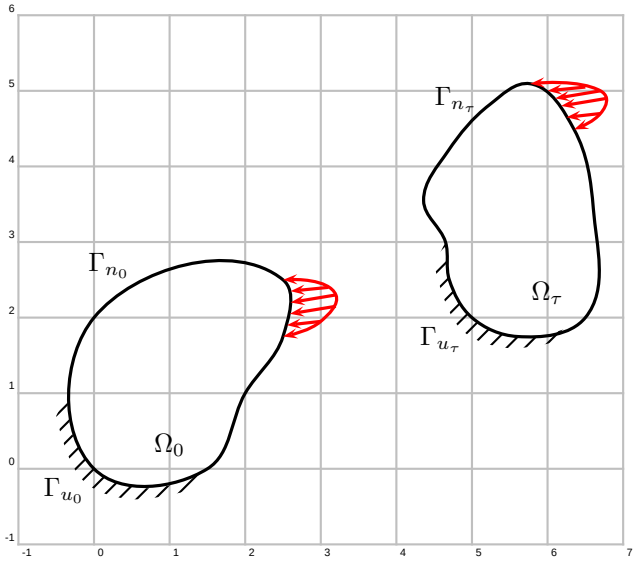


Figure 6.1: Espacio de aproximación

en donde $\frac{\partial \mathbf{s}}{\partial \mathbf{E}}$ presenta una doble simetría y permite escribir la expresión 6.7 como,

$$\int_{\Omega_0} \mathbf{s}(\vec{\mathbf{u}}) : \mathbf{F}^{\mathbf{T}}(\vec{\mathbf{u}}) \cdot \nabla^{\mathbf{T}} \delta\vec{\mathbf{v}}\, dv + \int_{\Omega_0} \mathbf{s}(\vec{\mathbf{u}}) : \nabla\delta\vec{\mathbf{u}} \cdot \nabla^{\mathbf{T}} \delta\vec{\mathbf{v}}\, dv + \int_{\Omega_0} \nabla\delta\vec{\mathbf{u}} \cdot \mathbf{F}(\vec{\mathbf{u}}) : \frac{\partial \mathbf{s}}{\partial \mathbf{E}}(\vec{\mathbf{u}}) : \mathbf{F}^{\mathbf{T}}(\vec{\mathbf{u}}) \cdot \nabla^{\mathbf{T}} \delta\vec{\mathbf{v}}\, dv = \mathcal{R} \cdot \delta\mathbf{v} \tag{6.12}$$

la ecuación 6.12 tiene tres variables que se pueden ver como independientes, $\vec{\mathbf{u}}$, $\delta\vec{\mathbf{u}}$ y $\delta\vec{\mathbf{v}}$, desde que $\vec{\mathbf{u}}$, la primera función sea correcta el valor de $\delta\vec{\mathbf{v}}$ será irrelevante puesto que satisfará todas las condiciones en el dominio y $\delta\vec{\mathbf{u}}$ será cero. Sin embargo si $\vec{\mathbf{u}}$ no es correcta entonces la dirección de búsqueda será $\delta\vec{\mathbf{u}} \neq 0$. Es importante observar que en la solución la primera integral del lado izquierdo $\int_{\Omega_0} \mathbf{F}(\vec{\mathbf{u}}) \cdot \mathbf{s}(\vec{\mathbf{u}}) : \nabla\delta\vec{\mathbf{v}}\, dv$ es igual al lado derecho de la ecuación, $\mathcal{R} \cdot \delta\mathbf{v}$, y que las otras integrales se desvanecen.

Usando el método de Ritz para la aproximación de la solución, se define la aproximación de la solución como una combinación lineal de funciones de forma,

$$\vec{\mathbf{u}} = \underline{\underline{\mathcal{U}}} \cdot \underline{\mathcal{N}} \tag{6.13}$$

$$u_i = \mathcal{U}_i^j \mathcal{N}^j \tag{6.14}$$

donde $\underline{\mathcal{N}} = [\mathcal{N}^1, \ldots, \mathcal{N}^m]^{\mathbf{T}}$ representa las m funciones de forma, $\underline{\underline{\mathcal{U}}}$ representa los valores de las combinaciones lineales sobre la función $\vec{\mathbf{u}}$, $i = 1, 2, 3$ y $j = 1, 2, \ldots, m$. Con esto ahora se pueden definir,

$$\begin{aligned} \nabla\vec{\mathbf{u}} &= \underline{\underline{\mathcal{U}}} \cdot \nabla\underline{\mathcal{N}} \\ u_{j,i} &= \mathcal{U}_j^k \mathcal{N}_{,i}^k \end{aligned} \tag{6.15}$$

de la misma forma $\delta u_i = \delta\mathcal{U}_i^j \mathcal{N}^j$, $\delta u_{j,i} = \delta\mathcal{U}_j^k \nabla\mathcal{N}_{,i}^k$ y usando el método de Galerkin en donde $\delta\vec{\mathbf{v}}$ se escoge con la misma base $\underline{\mathcal{N}}$ con la que se aproxima $\vec{\mathbf{u}}$ y completar un mismo número de ecuaciones como de incógnitas

$$\int_{{}^e\Omega_0} \mathbf{s}({}^e\vec{\mathbf{u}}) : \mathbf{F}^{\mathbf{T}}({}^e\vec{\mathbf{u}}) \cdot \nabla^{\mathbf{T}} \delta\vec{\mathbf{v}}\, dv = \int_{{}^e\Omega_0} \mathbf{s}(\vec{\mathbf{u}}) : (\mathbf{I} + \underline{\underline{\mathcal{U}}} \cdot \nabla\underline{\mathcal{N}}) \cdot \nabla^{\mathbf{T}} \underline{\mathcal{N}} \cdot \delta\underline{\underline{\mathcal{V}}}^{\mathbf{T}}\, dv \tag{6.16}$$

la siguiente expresión $\mathbf{s} : (\mathbf{I} - \underline{\underline{\mathcal{U}}} \cdot \nabla\underline{\mathcal{N}}) \cdot \nabla^{\mathbf{T}} \underline{\mathcal{N}} \cdot \delta\underline{\underline{\mathcal{U}}}^{\mathbf{T}}$ se puede reducir por medio del siguiente isomorfismo: $\mathbf{A} : \mathbf{B}$ *es equivalente a* $\overline{\mathbf{A}} \cdot \overline{\mathbf{B}}$*, donde* $\mathbf{A}$ *es un tensor de segundo orden simétrico.* Defínase el mapeo $f : \mathrm{Sym}_3 \ni \mathbf{A} \mapsto \overline{\mathbf{A}} \in \mathbb{R}^6$ que define $f(\mathbf{A}) = [A_{11}, A_{22}, A_{33}, A_{12}, A_{23}, A_{31}]$ y $g : \mathbb{R}^{3\times3} \ni \mathbf{B} \mapsto \overline{\mathbf{B}} \in \mathbb{R}^6$ que define $g(\mathbf{B}) = [B_{11}, B_{22}, B_{33}, B_{12} + B_{21}, B_{23} + B_{32}, B_{31} + B_{13}]$. Al volver a la expresión 6.16 y utilizar el isomorfismo

$\bar{\mathbf{s}} = f(\mathbf{s})$ y el siguiente isomorfismo,

$$
\begin{aligned}
g((\mathbf{I}+\underline{\underline{\mathcal{U}}}\cdot\nabla\underline{\underline{\mathcal{N}}})\cdot\nabla^{\mathbf{T}}\underline{\underline{\mathcal{N}}}\cdot\delta\underline{\underline{\mathcal{U}}}^{\mathbf{T}}) &= g((\nabla^{\mathbf{T}}\underline{\underline{\mathcal{N}}}\cdot\delta\underline{\underline{\mathcal{U}}}^{\mathbf{T}} + \underline{\underline{\mathcal{U}}}\cdot\nabla\underline{\underline{\mathcal{N}}}\cdot\nabla^{\mathbf{T}}\underline{\underline{\mathcal{N}}}\cdot\delta\underline{\underline{\mathcal{U}}}^{\mathbf{T}}) \\
&= \left[\mathcal{N}^k_{,j}\delta_{il} + \mathcal{U}^m_l\mathcal{N}^m_{,i}\mathcal{N}^k_{,j}\right]\cdot\left[\delta\mathcal{U}^k_l\right] \\
&= \begin{bmatrix} \delta\mathcal{U}^k_1\mathcal{N}^k_{,1} \\ \delta\mathcal{U}^k_2\mathcal{N}^k_{,2} \\ \delta\mathcal{U}^k_3\mathcal{N}^k_{,3} \\ \delta\mathcal{U}^k_1\mathcal{N}^k_{,2}+\delta\mathcal{U}^k_2\mathcal{N}^k_{,1} \\ \delta\mathcal{U}^k_2\mathcal{N}^k_{,3}+\delta\mathcal{U}^k_3\mathcal{N}^k_{,2} \\ \delta\mathcal{U}^k_3\mathcal{N}^k_{,1}+\delta\mathcal{U}^k_1\mathcal{N}^k_{,3} \end{bmatrix} + \begin{bmatrix} \mathcal{U}^k_l\mathcal{N}^k_{,1}\delta\mathcal{U}^m_l\mathcal{N}^m_{,1} \\ \mathcal{U}^k_l\mathcal{N}^k_{,2}\delta\mathcal{U}^m_l\mathcal{N}^m_{,2} \\ \mathcal{U}^k_l\mathcal{N}^k_{,3}\delta\mathcal{U}^m_l\mathcal{N}^m_{,3} \\ \mathcal{U}^k_l\mathcal{N}^k_{,2}\delta\mathcal{U}^m_l\mathcal{N}^m_{,1}+\mathcal{U}^k_l\mathcal{N}^k_{,1}\delta\mathcal{U}^m_l\mathcal{N}^m_{,2} \\ \mathcal{U}^k_l\mathcal{N}^k_{,3}\delta\mathcal{U}^m_l\mathcal{N}^m_{,2}+\mathcal{U}^k_l\mathcal{N}^k_{,2}\delta\mathcal{U}^m_l\mathcal{N}^m_{,3} \\ \mathcal{U}^k_l\mathcal{N}^k_{,1}\delta\mathcal{U}^m_l\mathcal{N}^m_{,3}+\mathcal{U}^k_l\mathcal{N}^k_{,3}\delta\mathcal{U}^m_l\mathcal{N}^m_{,1} \end{bmatrix} \\
&= \begin{bmatrix} \mathcal{N}^k_{,1} & 0 & 0 \\ 0 & \mathcal{N}^k_{,2} & 0 \\ 0 & 0 & \mathcal{N}^k_{,3} \\ \mathcal{N}^k_{,2} & \mathcal{N}^k_{,1} & 0 \\ 0 & \mathcal{N}^k_{,3} & \mathcal{N}^k_{,2} \\ \mathcal{N}^k_{,3} & 0 & \mathcal{N}^k_{,1} \end{bmatrix}\begin{bmatrix} \delta\mathcal{U}^k_1 \\ \delta\mathcal{U}^k_2 \\ \delta\mathcal{U}^k_3 \end{bmatrix} \\
&\quad + \begin{bmatrix} \mathcal{U}^k_1\mathcal{N}^k_{,1}\mathcal{N}^m_{,1} & \mathcal{U}^k_2\mathcal{N}^k_{,1}\mathcal{N}^m_{,1} & \mathcal{U}^k_3\mathcal{N}^k_{,1}\mathcal{N}^m_{,1} \\ \mathcal{U}^k_1\mathcal{N}^k_{,2}\mathcal{N}^m_{,2} & \mathcal{U}^k_2\mathcal{N}^k_{,2}\mathcal{N}^m_{,2} & \mathcal{U}^k_3\mathcal{N}^k_{,2}\mathcal{N}^m_{,2} \\ \mathcal{U}^k_1\mathcal{N}^k_{,3}\mathcal{N}^m_{,3} & \mathcal{U}^k_2\mathcal{N}^k_{,3}\mathcal{N}^m_{,3} & \mathcal{U}^k_3\mathcal{N}^k_{,3}\mathcal{N}^m_{,3} \\ \mathcal{U}^k_1\mathcal{N}^k_{,2}\mathcal{N}^m_{,1}+\mathcal{U}^k_1\mathcal{N}^k_{,1}\mathcal{N}^m_{,2} & \mathcal{U}^k_2\mathcal{N}^k_{,2}\mathcal{N}^m_{,1}+\mathcal{U}^k_2\mathcal{N}^k_{,1}\mathcal{N}^m_{,2} & \mathcal{U}^k_3\mathcal{N}^k_{,2}\mathcal{N}^m_{,1}+\mathcal{U}^k_3\mathcal{N}^k_{,1}\mathcal{N}^m_{,2} \\ \mathcal{U}^k_1\mathcal{N}^k_{,3}\mathcal{N}^m_{,2}+\mathcal{U}^k_1\mathcal{N}^k_{,2}\mathcal{N}^m_{,3} & \mathcal{U}^k_2\mathcal{N}^k_{,3}\mathcal{N}^m_{,2}+\mathcal{U}^k_2\mathcal{N}^k_{,2}\mathcal{N}^m_{,3} & \mathcal{U}^k_3\mathcal{N}^k_{,3}\mathcal{N}^m_{,2}+\mathcal{U}^k_3\mathcal{N}^k_{,2}\mathcal{N}^m_{,3} \\ \mathcal{U}^k_1\mathcal{N}^k_{,1}\mathcal{N}^m_{,3}+\mathcal{U}^k_1\mathcal{N}^k_{,3}\mathcal{N}^m_{,1} & \mathcal{U}^k_2\mathcal{N}^k_{,1}\mathcal{N}^m_{,3}+\mathcal{U}^k_2\mathcal{N}^k_{,3}\mathcal{N}^m_{,1} & \mathcal{U}^k_3\mathcal{N}^k_{,1}\mathcal{N}^m_{,3}+\mathcal{U}^k_3\mathcal{N}^k_{,3}\mathcal{N}^m_{,1} \end{bmatrix}\begin{bmatrix} \delta\mathcal{U}^m_1 \\ \delta\mathcal{U}^m_2 \\ \delta\mathcal{U}^m_3 \end{bmatrix} \\
&= (\underline{\underline{\mathbf{B}}}_{L0_m}+\underline{\underline{\mathbf{B}}}_{L1_m})\cdot\delta\underline{\underline{\mathcal{U}}}_m = \underline{\underline{\mathbf{B}}}_{L_m}\cdot\delta\underline{\underline{\mathcal{U}}}_m \qquad (6.17)
\end{aligned}
$$

finalmente la integral de la ecuación 6.16 es reescrita como

$$\int_{^e\Omega_0}\mathbf{s}(\vec{\mathbf{u}}):(\mathbf{I}+\underline{\underline{\mathcal{U}}}\cdot\nabla\underline{\underline{\mathcal{N}}})\cdot\nabla^{\mathbf{T}}\underline{\underline{\mathcal{N}}}\,dv\;\cdot\;\delta\underline{\underline{\mathcal{V}}}^{\mathbf{T}} = \int_{^e\Omega_0}\bar{\mathbf{s}}\cdot\underline{\underline{\mathbf{B}}}_L\,dv\;\cdot\;\delta\underline{\underline{\mathcal{V}}}^{\mathbf{T}} \qquad (6.18)$$

en donde $\underline{\underline{\mathbf{B}}}_L = [\underline{\underline{\mathbf{B}}}_{L_1},\underline{\underline{\mathbf{B}}}_{L_2},\cdots,\underline{\underline{\mathbf{B}}}_{L_m}]$.

Continuando con las otras integrales de 6.12 se pueden compactar a

$$
\begin{aligned}
\int_{^e\Omega_0}\nabla\delta\vec{\mathbf{u}}\cdot\mathbf{F}(\vec{\mathbf{u}}):\frac{\partial\mathbf{s}}{\partial\mathbf{E}}(\vec{\mathbf{u}}):\mathbf{F}^{\mathbf{T}}(\vec{\mathbf{u}})\cdot\nabla^{\mathbf{T}}\delta\vec{\mathbf{v}}\,dv &= \int_{^e\Omega_0}\nabla\delta\vec{\mathbf{u}}\cdot\mathbf{F}(\vec{\mathbf{u}}):\mathcal{C}(\vec{\mathbf{u}}):\mathbf{F}(\vec{\mathbf{u}})^{\mathbf{T}}\cdot\nabla^{\mathbf{T}}\delta\vec{\mathbf{v}}\,dv \\
&= \delta\underline{\underline{\mathcal{U}}}\;\cdot\;\int_{^e\Omega_0}\underline{\underline{\mathbf{B}}}_L^{\mathbf{T}}\cdot\underline{\underline{\mathcal{C}}}(\vec{\mathbf{u}})\cdot\underline{\underline{\mathbf{B}}}_L\,dv\;\cdot\;\delta\underline{\underline{\mathcal{V}}}^{\mathbf{T}}
\end{aligned}
$$

$$
\begin{aligned}
\int_{^e\Omega_0}\nabla\delta\vec{\mathbf{u}}\cdot\mathbf{s}(\vec{\mathbf{u}}):\nabla^{\mathbf{T}}\delta\vec{\mathbf{v}}\,dv &= \int_{^e\Omega_0}[\delta\mathcal{U}^m_k]\cdot\left[s_{ij}\mathcal{N}^m_{,i}\mathcal{N}^n_{,j}\right]\cdot[\delta\mathcal{V}^n_l]\;dv \\
&= \delta\underline{\underline{\mathcal{U}}}\;\cdot\;\int_{^e\Omega_0}\underline{\underline{\mathbf{B}}}_{NL}:\mathbf{s}\,dv\;\cdot\;\delta\underline{\underline{\mathcal{V}}}^{\mathbf{T}} \qquad (6.19)
\end{aligned}
$$

con esto la ecuación 6.12 se convierte en el sistema,

$$
\delta\underline{\underline{\mathcal{U}}}\;\cdot\;\underbrace{\left(\int_{^e\Omega_0}\underline{\underline{\mathbf{B}}}_L^{\mathbf{T}}(\underline{\underline{\mathcal{U}}})\cdot\underline{\underline{\mathcal{C}}}(\underline{\underline{\mathcal{U}}})\cdot\underline{\underline{\mathbf{B}}}_L(\underline{\underline{\mathcal{U}}})\,dv + \int_{^e\Omega_0}\underline{\underline{\mathbf{B}}}_{NL}(\underline{\underline{\mathcal{U}}}):\mathbf{s}(\underline{\underline{\mathcal{U}}})\,dv\right)}_{\text{Matriz de Rigidez}}\cdot\delta\underline{\underline{\mathcal{V}}}^{\mathbf{T}} = \underbrace{\left(\mathcal{R} - \int_{^e\Omega_0}\bar{\mathbf{s}}(\underline{\underline{\mathcal{U}}})\cdot\underline{\underline{\mathbf{B}}}_L(\underline{\underline{\mathcal{U}}})\,dv\right)}_{\text{Fuerzas externas menos internas}}\cdot\delta\underline{\underline{\mathcal{V}}}^{\mathbf{T}}
$$

$$\delta\underline{\underline{\mathcal{U}}}\;\cdot\;(\mathbf{K}_L+\mathbf{K}_{NL}) = \Delta\mathcal{R} \qquad (6.20)$$

y por último las iteraciones de Newton-Rhapson, teniendo en cuenta que $\delta\underline{\underline{\mathcal{U}}} = {}^{i+1}\underline{\underline{\mathcal{U}}} - {}^{i}\underline{\underline{\mathcal{U}}}$ quedan

$$ {}^{i+1}\underline{\underline{\mathcal{U}}} = {}^{i}\underline{\underline{\mathcal{U}}} + \mathbf{K}^{-1} \cdot \Delta\mathcal{R} \tag{6.21}$$

6.2 Comparación entre deformación lineal y deformación no-lineal

Las figuras de este capítulo fueron generadas en MATLAB, usando los programas implemetados en este trabajo (ver Anexos), teniendo en cuenta las ecuaciones desarrolladas en este capítulo y las relaciones constitutivas del final del cuarto capítulo. La renderización de las figuras se hizo con PSTricks. En la figura 6.2, se observa el estado indeformado del lado izquierdo y luego de aplicar las fuerzas y restricciones que se observan, se obtiene la figura de la derecha en donde aparecen dos tipos de configuraciones defomadas. En la figura de la derecha, la malla que aparece en color fuerte es el modelo no-lineal y la malla que aparece con color tenue es el modelo lineal.

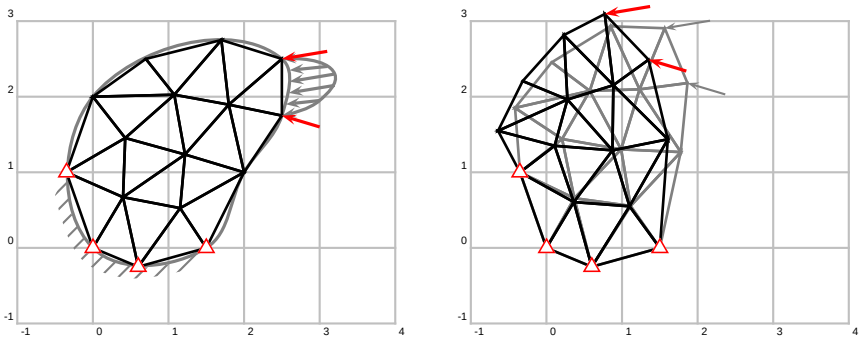


Figure 6.2: Espacio de aproximación

Esta figura muestra la necesidad de usar el modelo no-lineal en geometría, debido a las discrepancias entre las deformaciones de ambas configuraciones (modelo lineal y modelo no-lineal). Este modelo es coherente en la medida en que a mayor deformacion el material pierde su rigidez tangencial. Las no-linealidades de material se vuelven importantes bajo el modelo de Yeoh.

Chapter 7

Descripción de la aplicación

Que se hizo

En este capítulo se describe la aplicación desarrollada, los resultados obtenidos, detalles de la implementación, métodos y algoritmos utlizados. Finalmente se colocan algunos screenshots de la aplicación y los resultados en tiempo obtenidos.

La aplicación en general se puede describir como el uso de dos hilos, el primero es el algoritmo de generación de fuerzas (parte derecha fig.7) y el segundo el manejo de eventos del dispositivo háptico que se ejecuta en paralelo todo el tiempo (parte izquierda fig.7).

El algoritmo de generación de fuerzas comienza cargando el modelo del órgano o tejido a partir de un poliedro que representa la frontera del sólido a modelar, esta es preprocesada mediante un mallado volumétrico o de superficie dependiendo del método numérico. A su vez el hilo del dispositivo háptico envía señales de posición de la herramienta relativas al sólido, evaluando así el predicado de colisión. En caso de ocurrir una colisión, se genera entonces las condiciones de contorno. Una vez impuestas las condiciones de contorno, se procede a resolver los elementos finitos **FEM**, los cuales generaran la forma del órgano en estado deformado y además las fuerzas sobre la herramienta por causa de la reacción y/o deformación del tejido. Finalmente se envian las fuerzas generadas al hilo del dispositivo y el algorimo de generación vuelva a su estado de evaluación de la colisión.

7.1 Modelo tridimensional del órgano o tejido

El modelo tridimensional utilizado es un poliedro representado por una sopa de triángulos, los cuales se organizan en el mejor de los casos en una estructura Half-Edge, estructura implementada en la libreria CGAL. Los formatos que se pueden utilizar son: `.obj`, `.off`, `.stl` y `.wrl`. La ventaja de estos formatos es que son importados y exportados desde diferetnes tipos de programas de CAD-CAM-CAE, modeladores gráficos y editores de realidad virtual.

En este preproceso, si se logra formar la estructura Half-Edge, es posible optimizar la detección de colisiones, ya que esta estructura permite moverse de forma adyacente sobre las vencindades de un vértice, segmento o cara del poliedro.

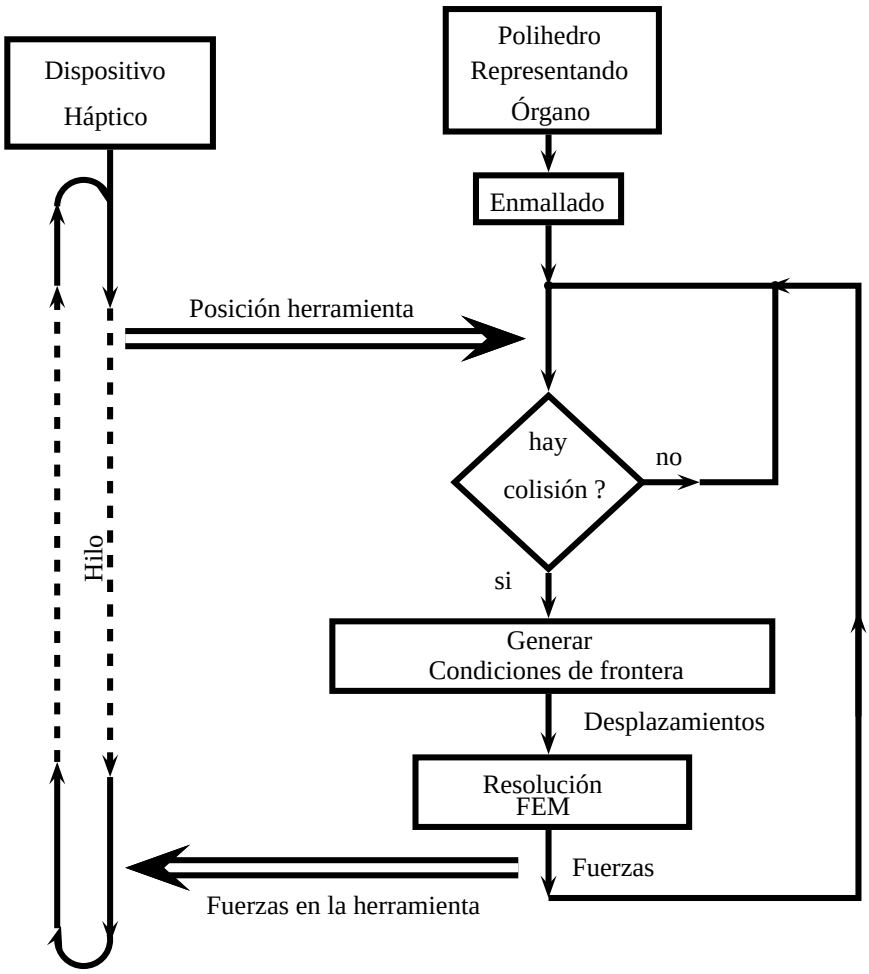


Figure 7.1: Descripción de la aplicación

7.2 Enmallado del modelo del órgano

En el caso de poder generar un manifold-2d orientado y cerrado es posible definir un dominio volumétrico, sobre el cual generar una malla. La malla es creada usando los malladores de CGAL, basados en *Silver Exudation* y Delaunay. La malla generada debe ser menor a 500 vértices para lograr un respuesta cercana a un 1 milisegundo y las relaciones de aspecto de cada tetraedro deben ser adecuadas para la condición de la matriz de rigidez.

La robustez de este mallado es importante en la solución de los elementos finitos que se vera más adelante, en donde un solo tetraedro con las relaciones de aspecto mal condicionadas es suficiente para que el método no converja. La robustez de este paso es generada gracias al *kernel geométrico* de CGAL, `Exact_predicates_exact_constructions`, que requiere ser compilado con GMP, LEDA o CORE, librerías con aritmética casi-exacta y exacta. Aunque este kernel es el más lento, este proceso se hace como preproceso y no afecta el desempeño del algoritmo, solo asegura que la malla sea adecuada, evitando jacobianos singulares y problemas en la etapa de FEM que se ejecuta en tiempo real.

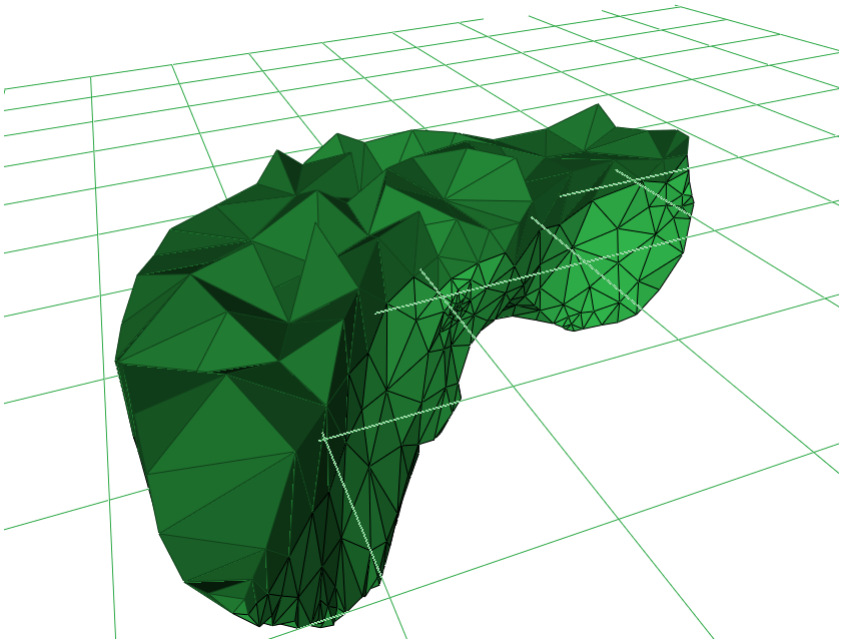


Figure 7.2: Enmallado de un hígado, con 146 tetrahedros

La triangulación que representa la malla es un refinamiento de una triangulacón regular, la clase que se utiliza de CGAL es `CGAL::Mesh_polyhedron_3<Gt>` y genera la estructura de triangulación. En esta etapa se debe construir un árbol de búsqueda de colisiones usando la clase `AABB_tree<AT>` donde `AT` es el parámetro que define un poliedro o una sopa de triángulos que se resuelve en tiempo de compilación. En esta parte del documento es importante recalcar que no siempre es posible generar la estructura Half-Edge y se debe usar una sopa de triángulos, esto es debido a la forma compleja del órgano, cuando esto ocurre no es posible optimizar la colisión.

7.3 Posición Herramienta

La posición de la herramienta es actualizada mediante un hilo, usando QThread, una clase de Qt, la cual permite utlizar hilos independiente de la plataforma. Se implementó una interfaz que heredandola permite añadir cualquier dispositivo a la aplicación, la aplicación se prueba con un joystick de la consola de video juegos PS2 de SONY y el joystick 3D-force de Logithech. Este hilo de posición se ejecuta paralelo al algoritmo de generación de fuerzas de manera que al detectar movimiento por parte del usuario, modifica las variables de posición de la herramienta dentro de la escena.

7.4 Detección de colisión

El paquete *(Axis-Aligned Bounding Box) AABB tree* crea una estructura estática que contiene la malla volumétrica tridimensional, en donde se crea una estructura Half-Edge a partir de la malla volumétrica de la frontera del sólido que se obtiene en la etapa de enmallado. Cuando se puede crear la Half-Edge es posible utilizar la clase de CGAL, `AABB_polyhedron_triangle_primitive<GeomTraits,Polyhedron>`, la cual es parametrizable con un kernel geométrico y un poliedro. La búsqueda de la colisión es llevada a cabo y

permite optimizar el proceso de colisión y generación de condiciones de contorno.

7.5 Generación de las condiciones de contorno

Como el mallado tiene en su frontera solo triángulos en el espacio, la colisión que se utiliza a esta altura del algoritmo es la intersección de un segmento de línea (que en este caso representa la herramienta) y el poliedro que representa el órgano. La intersección será entonces un punto o un segmento de línea fig.7.5, en el caso de un punto este podrá estar: 1) si el punto intersección coincide con un vértices entonces el desplazamiento se coloca en su totalidad sobre el punto, en la dirección de la línea que representa la herramienta, 2) si el punto queda sobre una arista de un par de triángulos para este caso se debe colocar el desplazamiento sobre los dos vértices que contienen al segmento que forma la arista de forma tal que la función de forma sea coherente al desplazamiento de la herramienta y 3) si el punto cae sobre el interior de un triángulo o cara, entonces el desplazamiento se coloca coherente a la función de forma sobre los tres vértices de la cara.

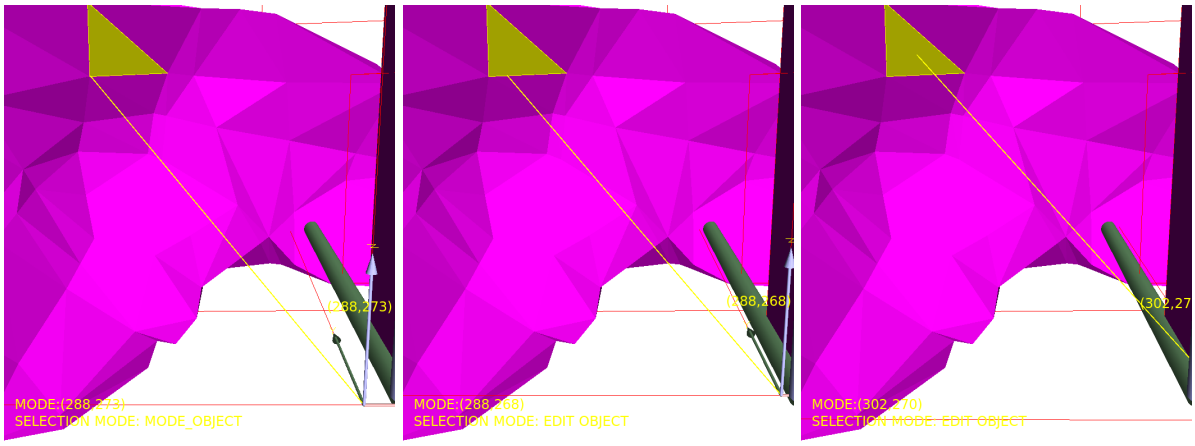


Figure 7.3: Posibles intersecciones

7.6 Resolución de la elasticidad por elementos finitos

Este es el paso más importante, en este punto se soluciona la ecuación que relaciona los desplazamientos y las fuerzas. Este paso consta de un proceso iterativo, el de Newton-Rhapson sobre el cual se encuentra el equilibrio de la malla que representa el sólido. En cada iteración se debe resolver un sistema matricial de la forma $A \cdot x = b$, que representa el cuello de botella de toda la aplicación. El resultado de este paso es el campo de desplazamientos de todo el tejido.

$$^{i+1}\underline{\underline{\mathcal{U}}} = {}^{i}\underline{\underline{\mathcal{U}}} + \mathbf{K}^{-1} \cdot \Delta\mathcal{R} \tag{7.1}$$

Este sistema consta de una matriz cuadrada esparcida de gran tamaño (un máximo 1500x1500), la solución de este sistema es por métodos iterativos, basados en el espacio de Krylov, el cual propone unas bases ortogonales conjugadas que encuentran la solución en máximo el número de filas de la matriz, pero suele pasar que con 17 a 20 iteraciones la solución es lo suficientemente precisa con BiCGSTAB. El solver y las matrices fueron utilizadas con la libreria `gmm++`, que permite utilizar aritmética modular sobre el albebra

lineal.

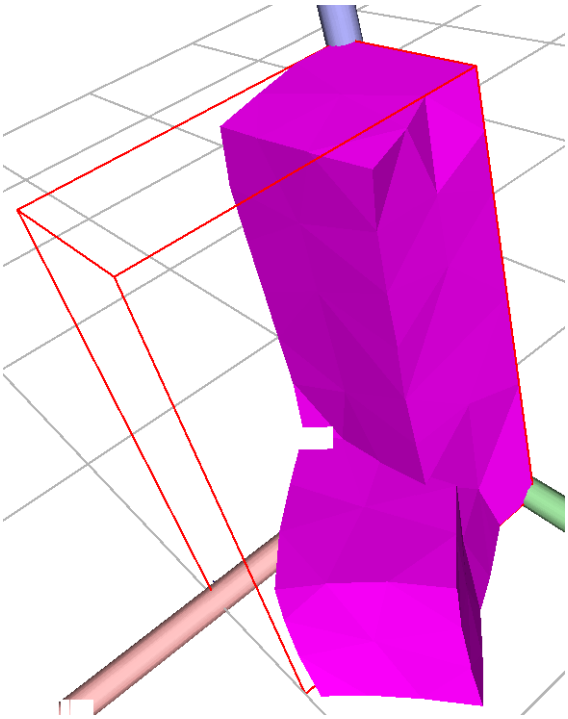


Figure 7.4: Sólido deformado después del paso de resolución de la elasticidad

Los desplazamientos se encuentran al resolver el siguiente sistema,

$$\begin{bmatrix} f_k \\ f_u \end{bmatrix} = \begin{bmatrix} K_{kk} & \vdots & K_{ku} \\ \cdots & \cdots & \cdots \\ K_{uk} & \vdots & K_{uu} \end{bmatrix} \cdot \begin{bmatrix} u_k \\ u_u \end{bmatrix} \tag{7.2}$$

donde los subíndices kk son los vértices de la malla a los cuales se les conoce el desplazamiento, ya sea por que son nodos que descansan sobre puntos soporte de la malla o porque pertencen al desplazameinto por causa del contacto de la herramienta. Los subíndices uu son los vértices que sus desplazamientos son una incógnita y se encuentra de la siguiente forma (observese que si se conoce el desplazamiento se desconoce la fuerza en el nodo y si se conoce la fuerza se desconoce el desplazamiento),

$$u_u = K_{uu}^{-1} \cdot (f_u - K_{uk} \cdot u_k) \tag{7.3}$$

para la cual f_u es cero cuando no se tiene en cuenta la gravedad y otras fuerzas externas.

7.7 Generación de fuerzas en la herramienta

La fuerza en la herramienta es un posproceso que se obtiene después de tener el campo vectorial de desplazamientos en los nodos. Se puede encontrar la otra ecuación que quedó pendiente en el sistema de eq.7.2, para el cual se puede encontrar las fuerzas de la herramienta,

$$f_k = K_{kk} \cdot u_k - K_{ku} \cdot u_u \tag{7.4}$$

este paso se puede reemplazar por los vértices que solo involucran los nodos de los triángulos afectados por la herramienta, y es un paso constante y sencillo, que queda optimizado por su simpleza,

$$f_{herr} = K_{herr \quad herr} \cdot u_{herr} - K_{herr \quad u} \cdot u_u \tag{7.5}$$

donde *herr* son los vértices que involucran la colisión de la herramienta.

Chapter 8

Conclusiones, Logros y Trabajos Futuros

Lo que queda

8.1 Conclusiones

- CGAL fue fundamental para la implementación, toma de tiempos, robustez de los algoritmos y manejo de formatos de entrada y salida. En definitiva CGAL permite tratar el diseño de la aplicación dándole robustez y optimización en el área de la geometría computacional y soporte para trabajar con FEM y BEM.
- Los elementos de contorno BEM, tiene una formulación compleja. Para el modelo lineal los BEM resultan más rápido que los FEM pero tienen la desventaja que en el caso no-lineal la formulación requiere conocer nodos interiores y se vuelve inviable. El posproceso de los BEM es demasiado costoso.
- Las altas discrepancias entre los modelos: 1) lineal en geometría y material, 2) geometría no-lineal y material lineal, y 3) geometría no-lineal y material no-lineal, obligan a que los modelos tengan que utilizar por lo menos las no-linealidades geométricas para que sean verosímiles en un ambiente de realidad aumentada.
- La detección de colisión entre el sólido y la herramienta usando las estructuras Half-Edge y un árbol AABB sobre los triángulos que representan al sólido, permite encontrar las condiciones de contorno de los elementos finitos de forma rápida ya que evita la fuerza bruta. Además una vez encontrada la colisión es posible actualizarla usando las propiedades de la estructura Half-Edge, ventaja que reduce drásticamente el predicado de colisión evitando así el uso de la búsqueda en el árbol AABB.
- Las pruebas con la GPU (en arquitectura de 12 y 96 núcleos) presentaron un cuello de botella en las mallas que se utlizaron, esto debido a que la aceleración en el cálculo se hace evidente cuando las mallas son superiores a 4000 vértices, pero las mallas más grandes que se utilizaron en la aplicación desarrollada fueron a lo sumo de 500 vértices. El bus de datos y los tiempos de copia de las matrices entre la memoria principal y la memoria de la GPU llevan a que no se obtenga ventaja usando la GPU. Se utilizó CUDA Sparse, una tarjeta nVIDIA GeForce 9400M de 12 núcleos y al final una tarjeta nVIDIA de 96 núcleos GeForce GT 240.
- El análisis cuasi-estático es suficiente para modelar y no es necesario hacer análisis de los eigenmodes, esto es debido a que los movimientos que se toman en la herramienta.

8.2 Logros

- Implementación de FEM resolviendo la elasticidad lineal bajo el modelo Saint Venant-Kirchhoff. Queda para futuras aplicaciones código en MATLAB sencillo y trasparente de interpretar, para renderizar y mallar (con ayuda de DISMESH) de cualquier dominio en 2D o 3D con lectores de `.obj` y `.off`.
- Implementación de BEM resolviendo la elasticidad lineal bajo el modelo Saint Venant-Kirchhoff. Queda para futuras aplicaciones código en MATLAB sencillo y trasparente de interpretar, para renderizar y mallar (con ayuda de DISMESH) de cualquier dominio en 2D o 3D con lectores de `.obj` y `.off`.
- Formulación general de la elasticidad lineal en BEM.
- Formulación del elemento finito para la elasticidad no-lineal en general.
- Implementación no-lineal de geometría en la ecuación de elasticidad utilizando el tensor de Cauchy-Green. Quedando código en MATLAB sencillo y trasparente. Implementación en C++ bajo programación genérica.
- Implementación no-lineal de material en la ecuación de elasticidad utilizando los modelos de Mooney-Rivlin, Neo-Hookean y Yeoh. Quedando código en MATLAB sencillo y trasparente. Implementación en C++ bajo programación genérica código complejo de entender sin conocimientos de `templates`, pero hasta 100x más rápido que el de MATLAB (solamente resolviendo el sistema $Ax = b$ por BIGSTAB).
- El ciclo del hilo del dispositivo háptico se logró bajar a 10 milisegundos. Aún 10 veces por encima de la velocidad objetivo requerida en una aplicación háptica.
- Desarrollo de una librería que integra cualquier dispositivo háptico, el motor de colisiones basado en los algoritmos de CGAL, el mallador volumétrico y superficial basado en CGAL, y la solución por elementos finitos. Manejo de la escena con dos widgets para renderizar la escena en forma de árbol informativo y bajo el contexto de OpenGL (ver Anexos).

8.3 Trabajos Futuros

- *Una primera aplicación de entrenador quirúrgico.* Integración con los trabajos de [32, 36], el código desarrollado en MATLAB sería fundamental para el rápido entendimiento y avance en el modelamiento del tejido. La librería en C++ sirve para acoplar el dispositivo háptico al modelo del tejido.
- *Solución de elasticidad, usando programación en paralelo y programación genérica* . Optimización del código en C++. Elb uso de las tarjetas gráficas solo se utilizó para la manipulación de matrices esparcidas, es posible hacer la paralelización del algoritmo para que sea resulto de forma optima usando las tarjetas gráficas. Proponer un esquema sobre la solución por Newton-Rhapson, la cual es el cuello de botella ya que en su interior resulve por otro metodo iterativo basado en los métodos de Krylov cada paso de Newton-Rhapson. Se propone entonces usar el estado anterior de la malla como punto de partida en el siguiente paso de Newton-Rhapson, esto no se hizo y puede bajar significativamente las iteraciones.
- *Modelamiento mecánica multicuerpo.* La librería ha sido diseñada no solo para esta aplicación, si no en general para la animación de la dinámica multicuerpo. Es posible investigar haciendo uso de ella.
- *Disminuir el tiempo de la solución de elasticidad empleando redes neuronales.* Entrenar una red neuronal que copie el comportamiento del tejido y así suprimir todos los metódos iterativos. Proponer la arquitectura ideal y el tamaño de la red serían objetivos claros. La arquitectura podria ser estudiada en base a el grafo que forma la conectividad de la malla tridimensional. La red se puede entrenar con los modelos implementados en esta tesis.

- *Modelamiento del corte del tejido u órgano con XFEM* La fractura y avance de grietas en tejidos es un tema que falta por investigar, los elementos XFEM pueden ser una solución para el corte en un procedimiento quirúrgico.

Appendix A

VISUALIZACION DE LA ESCENA

luces, cámara, acción...

A.1 Creación de una escena

Una escena esta compuesta por luces, cámaras y objetos. La escena maneja cada uno de estos elementos, modificando sus propiedades como posiciones y orientación, existen manejadores de escena como OpenSG u OpenSceneGraph, pero sin mayor complicación se extendió el visualizador de OpenGL, libQGLViewer, para manejar la escena. La escena se maneja basicamente con tres clases: Items, Scenes y Viewers.

A.1.1 Viewers

Los *Viewers* son básicamente visualizadores de la escena y pueden ser Widgets para el manejo y edición de sus contenidos o renderizadores de la escena sobre un contexto de OpenGL, estas clases forman parte de la interfaz gráfica de usuario. Los widgets en especial son visualizadores del árbol de escena, estos visulizadores

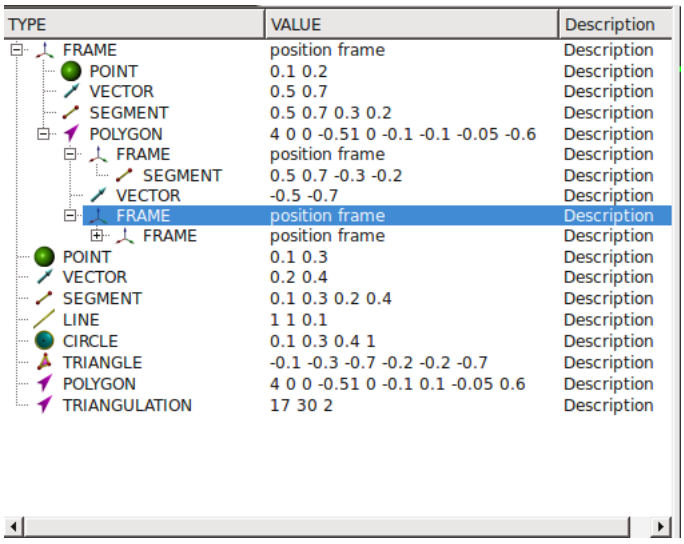


Figure A.1: Scene tree viewer

permites la ubicación, selección, revision de la información de cada objeto en la escena, sobre estos se puede modificar los parentezcos, el cambio de formas o primitivas geométricas, las posiciones, orientaciones, colores, la posibilidad de ser o no renderizado, la inclusión dentro del motor de colisiones y muchas otras cosas más. Por otra parte el renderizador encargado de visualizar la escena de forma tridimensional, se encarga tambien de capturar la interacción con el usuario, como lo son las acciones de seleccion, arrastre y el manejo del joystick, para comunicar cualquier cambio a la escena directamente. El concepto principal estará dado por:

```
1  template <typename S>
2  class SceneGLViewer : public QGLViewer {
3  public:
4      SceneGLViewer(S* _s, QWidget *parent = 0);
5
6  protected:
7      void init() {  ...; s->init();    }
8      void draw() {  ...; s->draw();    }
9
10     void drawWithNames() {  ...; s->drawWithNames(select_mode); ...;    }
11     void endSelection(const QPoint &/*point*/) {  ...;
12         s->endSelection(nbHits,selectBuffer(),select_mode);
13         ...;
14     }
15
16     void keyPressEvent(QKeyEvent *e);
17
18     void mousePressEvent(QMouseEvent *e);
19     void mouseMoveEvent(QMouseEvent *e);
20     void mouseReleaseEvent(QMouseEvent *e);
21
22 private:
23     S* s;
24 };
```

Anexo_Computacion_grafica/sceneGLViewer_concept.hpp

la escena finalemente visualizada de la fig:A.1, sería

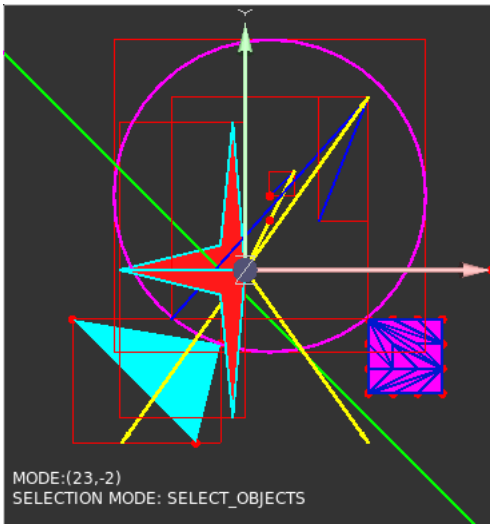


Figure A.2: Scene openGL viewer

A.1.2 Scenes

Las *Scenes* son una interfaz que permite la navegación y edición fácil, tiene tres finalidades: 1) poder usar las Widget de visualización de Qt, como QtTreeView, 2) interfazar los Items con Viewer y 3) generar la escena en XML y a partir de un documento DOM generar la scena. Esta clase tambien se encarga de los motores de colision, la conexión del joystick con cualquier elemento de la escena, activar el motor FEM-BEM para la generación de fuerzas.

```
1  template <typename K_ItemTraits = K_standard>
2  class SceneModel : public QAbstractItemModel
3  {
4  public:
5      typedef SceneItem<K_ItemTraits>                    Item;
6      typedef QDomNode                                   DomNode;
7      typedef QString                                    DomString;
8      typedef typename Item::GeoObj                      GeoObj;
9
10     typedef SceneDealerDOM<SceneModel>                 DealerDom;
11
12     SceneModel(QObject *parent = 0) : QAbstractItemModel(parent);
13
14     ¬SceneModel();
15
16     void addDomScene(const char* filename) {}
17
18     template <typename T>
19     Item* appendObject(T* ptr_obj, Item* parent=0) {}
20
21     QVariant data(const QModelIndex &index, int role) const {}
22     Qt::ItemFlags flags(const QModelIndex &index) const {}
23     QVariant headerData(int section, Qt::Orientation orientation,
24                         int role = Qt::DisplayRole) const {}
25     QModelIndex index(int row, int column,
26                       const QModelIndex &parent = QModelIndex()) const {}
27     QModelIndex parent(const QModelIndex &index) const {}
28     int rowCount(const QModelIndex &parent = QModelIndex()) const {}
29     int columnCount(const QModelIndex &parent = QModelIndex()) const {}
30
31     bool setData(const QModelIndex &index, const QVariant &value,
32                  int role = Qt::EditRole) {}
33     bool insertRows(int /*row*/, int /*count*/,
34                     const QModelIndex &parent = QModelIndex()) {}
35     bool removeRows(int position, int rows,
36                     const QModelIndex &parent = QModelIndex()) {}
37     void init() { rootItem->init(); }
38     void draw() { rootItem->draw(); }
39     void drawWithNames(Selection_mode sm = SELECT_OBJECTS) {}
40     void endSelection(int n, unsigned int* selItems, Selection_mode sm = SELECT_OBJECTS);
41     QString getTextStatus();
42
43 private:
44     Item *getItem(const QModelIndex &index) const;
45
46     typename std::vector<GeoObj*> objsList;
47     typename std::vector<Item*> itemsList;
48     typename std::vector<Item*> selectedItems;
49     Item *tmp_ptr_item;
50     Item *rootItem;
51     std::string text_status;
52 };
```

Anexo_Computacion_grafica/sceneModel_concept.hpp

La escena esta escrita de forma que se pueda acoplar cualquier tipo de objeto que se ajuste al concepto de *item* descrita por el parametro de la plantilla `K_ItemTraits`. Se escribieron dos tipos de scena, una para estructuras tipo árbol y otra para estructuras tipo lista donde no se pueden colocar parentezcos entre objetos gráficos.

A.1.3 Items

Los ítems son los nodos que contiene la escena, deben cumplir el concepto para poder ser en lazado a la escena y que pueda ser interpretado como elementos DOM, el manejo de los iconos, los streams necesarios para su manipulación. La generalidad de los objetos se alcaza por un apuntador a la clase de usuario, que debe cumplir la sintaxis del concepto `GeoObj` para acoplar los métodos de edición e interación con el usuario.

Bibliography

[1] Pierre Alliez, Laurent Rineau, Stphane Tayeb, Jane Tournois, and Mariette Yvinec, *3D mesh generation*, CGAL User and Reference Manual, CGAL Editorial Board, 3.8 ed., 2011, http //www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html#Pkg Mesh_3. 25

[2] Pierre Alliez, Stphane Tayeb, and Camille Wormser, *Aabb tree*, CGAL User and Reference Manual, CGAL Editorial Board, 3.8 ed., 2011, http //www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html#Pkg AABB_tree. 25

[3] M.A Audette, V. Hayward, O. Astley, M. Doyon, G.A. McCallister, and K. Chinzei, *A pc-based system architecture for real-time finite element-based tool-specific surgical simulation*, ICS Internation Congress Series **1268** (2004), 378–383. 5

[4] K.J Bathe, *Finite element procedures*, Prentice Hall, 1996. 3, 4, 9, 13, 37, 53

[5] E Bayro-Corrochano, *Geometric Computing: For wavelet transforms, robot vision, learning, control and action*, Springer, 2010. 3

[6] C.A Berbia and J. Dominguez, *Boundary elements: An introductory course*, WITPress, 1998. 11

[7] Jean-Daniel Boissonant, Olivier Devillers, Silvain Pion, Monique Teillaud, and Mariette Yvinec, *Triangulation in CGAL*, Computational Geometry: Theory and Applications **22** (2002), no. 8, 5–19. 21

[8] J. Bonet and R.D. Wood, *Nonlinear continuum mechanics for finite element analysis*, 2nd edition ed., Cambridge, 2008. 4, 37, 53

[9] Matthias Bsken, *2D range and neighbor search*, CGAL User and Reference Manual, CGAL Editorial Board, 3.8 ed., 2011, http //www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html#Pkg PointSet2. 24

[10] Stéphane Cotin, Hervé Delingette, and Nicholas Ayache, *Real-time elastic deformations of soft tissues for surgery simulation*, 1998. 4, 5, 10

[11] ———, *A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation*, Visual Computer **16** (2000), no. 8, 437–452. 10

[12] C. Courtecuisse, H. Jung, J. Allard, C. Duriez, D.Y Lee, and S. Cotin, *GPU-based real-time soft tissue deformation with cutting and haptic feedback*, Progress in Biophysics and Molecular Biology **103** (2010), 159–168. 14

[13] S. De and K.J. Bathe, *The method of finite spheres with improved numerical integration*, Computers and Structures (2001). 11

[14] Gilles Debunne, *libQGLViewer - A 3D OpenGL viewer library based on Qt*, May 2003. 4

[15] C. Dick, J. Georgii, and R. Westermann, *A real-time multigrid finite hexahedra method for elasticity simulation using CUDA*, Simulation Modelling Practice and Theory **19** (2011), 801–816. 14

[16] C. Ericson, *Real-time collision detection*, Elsevier, 2005. 14

[17] F. Hartmann, *Introduction to Boundary Elements: theory and applications*, Springer-Verlag, 1989. 12

[18] Susan Hert, Michael Hoffmann, Lutz Kettner, Silvain Pion, and Michael Seel, *An adaptable and extensible geometry kernel*, Computational Geometry: Theory and Applications **38** (2007), no. 8, 16–36. 3, 24

[19] Silicion Graphics Inc, *OpenGL*, May 1992. 4

[20] P. Jimnez, F. Thomas, and C. Torras, *3D Collision Detection: A Survey*, Computers and Graphics **25** (2001), no. 2, 269–285. 13

[21] Lutz Kettner, *Using generic programming for designing a data structure for polyhedral surfaces*, Computational Geometry: Theory and Applications **13** (1999), no. 8, 65–90. 21

[22] ———, *3D polyhedral surfaces*, CGAL User and Reference Manual, CGAL Editorial Board, 3.8 ed., 2011, http //www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html#Pkg Polyhedron. 25

[23] ———, *Halfedge data structures*, CGAL User and Reference Manual, CGAL Editorial Board, 3.8 ed., 2011, http //www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html#Pkg HDS. 25

[24] Lutz Kettner, Kurt Mehlorn, Sylvain Pion, Stefan Schirra, and Chee Yap, *Classroom examples of robustness problems in geometric computations*, Computational Geometry: Theory and Applications **40** (2008), no. 8, 61–78. xiii, 19

[25] Yi-Je Lim and Suvranu De, *Real time simulation of nonlinear tissue response in virtual surgery using the point collocation-based method of finite spheres*, Computer Methods in Applied Mechanics and Engineering (2007). 5

[26] K. Maleknehad and H. Mesgarani, *Numerical method for a nonlinear boundary integral equation*, Applied Mathematics and Computation **182** (2006), 1006–1009. 5

[27] Wouter Mollemans, Filip Schutyser, Nasser Nadjmi, and Paul Suetens, *Very fast soft tissue predictions with mass tensor model for maxillofacial surgery planning systems*, ICS International Congress Series **1281** (2005), 491–496. 10

[28] C. Monserrat, U. Meier, M. Alca niz, F. Chinesta, and M.C Juan, *A new approach for the real-time simulation of tissue deformations in surgery simulation*, Computer Methods and Programs in Biomedicine **64** (2001), 77–85. 4

[29] Qt community Nokia, *Qt*, May 1992. 4

[30] nvidia corporation, *CUSPARSE Library*, NVIDIA Corporation, 2010. 3, 5

[31] Mathieu Desbrun Of, Gilles Debunne, U. Of So. Cal, Marie paule Cani, and Alan H. Barr, *Dynamic real-time deformations using space time adaptive sampling gilles debunne*, 2001. 8, 9

[32] María Luisa Pinto, *Análisis e implementación de una interfaz háptica en entornos virtuales*, PhD dissertation, Universidad Nacional de Colombia, Bogotá, Junio 2009, Este trabajo es parte del RAQ. xiii, 1, 2, 3, 4, 66

[33] Sylvain Pion and Monique Teillaud, *3D triangulation data structure*, CGAL User and Reference Manual, CGAL Editorial Board, 3.8 ed., 2011, http //www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html#Pkg TDS3. 24

[34] ———, *3D triangulations*, CGAL User and Reference Manual, CGAL Editorial Board, 3.8 ed., 2011, http //www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html#Pkg Triangulation3. 24

[35] Sylvain Pion and Mariette Yvinec, *2D triangulation data structure*, CGAL User and Reference Manual, CGAL Editorial Board, 3.8 ed., 2011, http //www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html#Pkg TDS2. 24

[36] Daniel Andrés Ramirez, *Diseño de una plataforma robótica paralela 6-dof para asistente quirurgico en cirugías de reconstrucción cráneo-facial*, PhD dissertation, Universidad Nacional de Colombia, Bogotá, Octubre 2009, Este trabajo es parte del RAQ. xiii, 1, 2, 3, 66

[37] Laurent Rineau, *2D conforming triangulations and meshes*, CGAL User and Reference Manual, CGAL Editorial Board, 3.8 ed., 2011, http //www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html#Pkg Mesh2. 25

[38] Laurent Rineau and Mariette Yvinec, *3D surface mesh generation*, CGAL User and Reference Manual, CGAL Editorial Board, 3.8 ed., 2011, http //www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html#Pkg SurfaceMesher3. 25

[39] J. Sanders and E. Kandrot, *CUDA by Example: An introduction to general-purpose gpu programming*, Addison-Wesley, 2010. 3

[40] Jonathan Richard Shewchuk, *Robust Adaptive Floating-Point Geometric Predicates*, Proceedings of the Twelfth Annual Symposium on Computational Geometry, Association for Computing Machinery, May 1996, pp. 141–150. xiii, 20

[41] ———, *Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator*, Applied Computational Geometry: Towards Geometric Engineering (Ming C. Lin and Dinesh Manocha, eds.), Lecture Notes in Computer Science, vol. 1148, Springer-Verlag, May 1996, From the First ACM Workshop on Applied Computational Geometry, pp. 203–222. 16, 17, 25

[42] Dave Shreiner, *OpenGL Programming Guide: The official guide to learning OpenGL, versions 3.0 and 3.1*, seventh ed., Addison-Wesley, 2008. 3

[43] Kenneth Sundaraj, *Real-time dynamic simulation and 3d interaction of biological tissue : Application to medical simulators*, PhD dissertation, INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE, Department of French, January 2004, This is a full PHDTHESIS entry. 3, 9, 10, 12, 13

[44] Larry A. Taber, *Nonlinear theory of elasticity: Applications in biomechanics*, first ed., World Scientific Publishing, Toh Tuck Link, Singapore, January 2004, Soft biologica tissues models. 3, 4, 9, 37

[45] Y.M. Tang, A.F. Zhou, and K.C. Hui, *Comparison of fem and bem for interactive objetive simulation*, Computer-Aided Design **38** (2006), 874–886. 12, 37

[46] Z.A Taylor, O. Comas, M. Cheng, and J. Passenger, *On modelling of anisotropic viscoelasticity for soft tissue simulation: Numerical solution and GPU execution*, Medical Image Analysis **13** (2009), 234–244. 14

[47] D. Terzopoulos and K. Waters, *Physically-based facial modeling analysis and animation*, Journal of Visualization and Computer Animation (1990). 8

[48] The CGAL Project, *CGAL user and reference manual*, 3.8 ed., CGAL Editorial Board, 2011, http //www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html. 5

[49] A.C Ugural and S.K Fenster, *Advanced strength and applied elasticity*, Prentice Hall, 2003. 37

[50] D. Vandervoorde and N.M. Josuttis, *C++ Templates - The Complete Guide*, Addison-Wesley, 2002. 3

[51] P. Wang, A.A. Becker, I.A. Jones, A.T. Glover, S.D. Bendford, C.M. Greenhalgh, and M. Vloeberghs, *A virtual reality surgery simulation of cutting and retractions in neurosurgery with force-feedback*, Computer Methods and Programs in Biomedicine **84** (2006), 11–18. 5, 12, 37, 49

[52] P. Wang, A.A. Becker, I.A. Jones, A.T. Glover, S.D. Benford, C.M. Greenhalgh, and M. Vloeberghs, *Virtual reality simulation of surgery with haptic feedback based on the boundary element method*, Computers and Structures **85** (2007), 331–339. 5

[53] Mariette Yvinec, *2D triangulations*, CGAL User and Reference Manual, CGAL Editorial Board, 3.8 ed., 2011, http //www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html#Pkg Triangulation2. 24