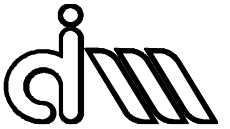
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Departamento de Ingeniería Mecánica y de Materiales



Trabajo Fin de Máster en Ingeniería Mecánica

Resolución de problemas de grandes desplazamientos mediante elementos finitos utilizando mallados cartesianos independientes de la geometría.

Presentado por: D. Francisco Javier Paredes Mera

Dirigido por: Dr. D. Manuel Tur Valiente

Valencia, febrero de 2017

TRABAJO FIN DE MÀSTER

Resolución de problemas de grandes desplazamientos mediante elementos finitos utilizando mallados cartesianos independientes de la geometría.

---

para la obtención del grado de

Master en Ciencias de Ingeniería Mecánica

presentado por

D. Francisco Javier Paredes Mera

en el

Departamento de Ingeniería Mecánica y de Materiales de la Universitat Politécnica de Valéncia

dirigido por

Dr. D. Manuel Tur Valiente

Valencia, febrero 2017

Resumen

El departamento de Ingeniería Mecánica y de Materiales ha estado trabajando en los últimos años en el desarrollo de un código de elementos finitos basado en mallados cartesianos independientes de la geometría (cgFEM), para resolver problemas elásticos en 2D y 3D. El código se ha programado utilizando MATLAB, a través de diferentes proyectos final de carrera, tesis de master y proyectos de investigación.

La industria de los elementos finitos siempre se encuentra en constante desarrollo y este código cgFEM busca sacar ventaja con la optimización de datos generados al realizar el mallado de elementos cartesianos. Este tipo de mallado optimizado busca ahorrar tiempo de cálculo que a la vez esto se traduce en la minimización del coste computacional de la solución. El código se ha realizado para resolver problemas de optimización.

El objetivo de este Trabajo Fin de Master es el desarrollo del código para resolver el problema elástico de grandes desplazamientos. Para conseguir este objetivo se ha programado en primer lugar un elemento finito estándar en el software MATLAB. Esto permite comprobar el código generado

y validarlo para un punto de integración cualquiera. Posteriormente se ha adaptado al programa cgFEM.

Una vez realizado lo anterior se han resuelto unos problemas ejemplos para comparar nuestros resultados con un software comercial como lo es ANSYS y concluir los resultados obtenidos.

Abstract

The Department of Mechanical and Materials Engineering has been working in recent years on the development of a finite element code based on independent geometry cartesian grid (cgFEM), to solve elastic problems in 2D and 3D. The code has been programmed using MATLAB, through different final projects, master thesis and research projects.

The finite element industry is always in constant development and its cgFEM code seeks the advantage with the optimization of the data generated to perform the design of cartesians elements. This type of optimized mesh seeks to save calculation time which at the same time results in minimization of the computational cost of the solution. The code has been made to solve optimization problems.

The objective of this Master's End Work is the development of the code to solve the elastic problem of large displacements. To achieve this, a standard fine element has been programmed in the MATLAB software first. This allows you to check the generated and valid code for any integration point. It has subsequently adapted to the cgFEM program.

Once done the above have been solved some problems refer to our results with commercial software such as ANSYS and conclude the results obtained.

Resum

El departament d'Enginyeria Mecànica i de Materials ha estat treballant en els últims anys en el desenvolupament d'un codi d'elements finits basat en mallats catesians independents de la geometria (cgFEM), per resoldre problemes elàstics en 2D i 3D. El codi s'ha programat utilitzant MATLAB, a través de diferents projectes finals de carrera, tesi de màster i projectes d'investigació.

La indústria dels elements finits sempre es troba en constant desenvolupament i el codi cgFEM busca l'millorar les formulacions tradicionals amb l'optimització de les dades generades per realitzar malles d'elements cartesians. Aquest tipus de mallat optimitzat busca estalviar temps de càlcul que alhora es tradueix en la minimització del cost computacional de la solució. El codi s'ha aplicat per resoldre problemes d'optimització.

L'objectiu d'aquest Treball Fi de Màster és el desenvolupament del codi per resoldre el problema elàstic de grans desplaçaments. Per obtenir aquest objectiu s'ha programat en primer lloc un element fi estàndard en el programa MATLAB. Això permet comprovar el codi generat i validarlo per a un punt

d'integració qualsevol. Posteriorment s'ha adaptat al programa cgFEM.

Un cop fet això s'han resolt alguns problemes de referencia i sì lan comparat els nostres resultats amb els d'un programa comercial com és ANSYS.

TABLA DE CONTENIDO

Resumen 3

Resum 7

1. Introducción 11

1.1 Antecedentes. 11

1.2 Objetivos 12

2. Formulación de Grandes Desplazamientos con elementos Finitos. 13

2.1 Medidas de la deformación 16

2.2 Medidas de la tensión 18

2.3 Principio de los trabajos virtuales 20

2.4 Modelo de Saint-Venant del material. 25

2.5 Modelo de Neo-Hookean del material 26

3. Programa cgFEM. 28

4. Implementación de grandes desplazamientos en cgFEM. 33

4.1 Desarrollo del código en Matlab. 33

4.2 Función sd2Dsolid 35

4.3 Función ShapeFunc4N_loc. 39

4.4 Funcion derN 40

4.5 Función Jaco 41

4.6 Función dibuja_cuad 43

4.7 Modelo de Saint-Venant del material. 44

4.7.1 Función K_sd4NSolid 44

4.7.2 Función fint_sd4Nsolid 48

4.8 Modelo de Neo-Hookean del material 51
4.8.1 Función K_ld4Nsolid 51
4.8.2 Función fint_ld4Nsolid 54
4.9 Código en cgFEM 56
5. Ejemplos y comparación con ANSYS 57
5.1 Cubo. 57
5.2 Esfera Hueca. 67
5.3 Placa con Agujero. 78
5.4 Análisis de Resultados. 88
6. Conclusiones 89
BIBLIOGRAFIA. 91
ANEXO 92

1. Introducción

1.1 Antecedentes.

La industria de los elementos finitos siempre se encuentra en el desarrollo de nuevas técnicas computacionales que puedan minimizar costes de cálculo. La Universidad Politécnica de Valencia en su departamento de Ingeniería Mecánica y de Materiales ha desarrollado el código de elementos finitos con mallados cartesianos independientes de la geometría llamado cgFEM.

A diferencia del método tradicional, en las mallas cartesianas la geometría del sólido deformable no coincide con la malla y se deben utilizar técnicas de integración especiales. Actualmente el código puede resolver problemas de elasticidad lineal en pequeños desplazamientos. En este trabajo se trata de implementar la formulación de grandes desplazamientos en MATLAB y adaptarla para su uso en el programa de mallas cartesianas, cgFEM.

Con el fin de obtener este objetivo en primer lugar se desarrolla la formulación de grandes desplazamientos para un caso 2D con elementos finitos estándar (cuadrilátero de 4 nodos). Se realiza la programación en MATLAB y se resuelven problemas con solución conocida para comprobar la implementación realizada. Con esto se

puede verificar que el cálculo para un punto de integración genérico se realiza correctamente.

Posteriormente las funciones generadas para el cálculo de la fuerza elástica y la matriz tangente en un punto de integración, se adaptan para ser utilizadas en cgFEM, con unos cambios menores en el programa.

1.2 Objetivos

- Revisión de la teoría de los métodos de resolución de problemas de grandes desplazamientos.
- Programación del método para un elemento finito estándar utilizando MATLAB.
- Adaptación al programa cgFEM (FEAVOX).
- Resolución de problemas de ejemplo y comparación con resultados del software ANSYS.

2. Formulación de Grandes Desplazamientos con elementos Finitos.

Se va a realizar un repaso de la teoría y formulación usada para resolver el problema de grandes desplazamientos en MATLAB.

El código a programar se enfoca en resolver problemas de grandes deformaciones de un sólido con comportamiento elástico lineal.

El problema de grandes deformaciones se puede definir como el cambio de la configuración del sólido al deformarse afectando a la distribución de las cargas que transmite, de manera que la relación entre desplazamientos y deformaciones no es lineal. [1]

La figura 1 representa esquemáticamente la configuración inicial de un sólido (sin deformar), es decir, la posición **X** que ocupan las partículas del sólido en un espacio euclideo. Cuando el sólido cambia el lugar que ocupan sus partículas debido a las acciones que recibe, se obtiene su configuración **x** en otro instante de tiempo a través de la siguiente transformación: [2]

$$\mathbf{x} = \varphi_t(\mathbf{X}) = \varphi(\mathbf{X}, t) \qquad (1)$$

La ecuación 1 corresponde a una descripción Lagrangiana del movimiento de los sólidos. Normalmente se utilizan letras mayúsculas para referirse a magnitudes de la configuración inicial y letras minúsculas para la configuración deformada. La posición en la configuración deformada se puede escribir de la siguiente manera:

$$\mathbf{x}(\mathbf{X}) = \varphi_t(\mathbf{X}) = \mathbf{X} + \mathbf{u}(\mathbf{X}) \qquad (2)$$

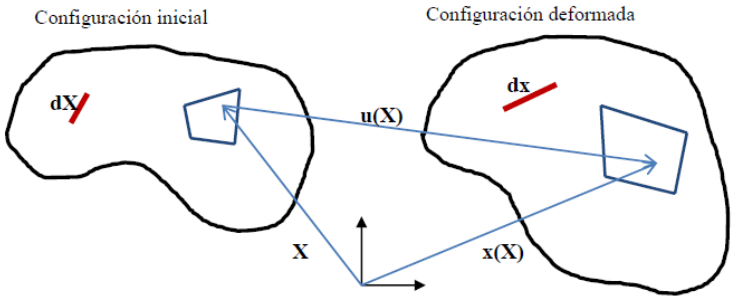


Figura 1. [2]

En la figura 1 también se dibuja esquemáticamente el lugar que ocupa un elemento cuadrilátero de cuatro nodos en la configuración inicial y deformada.

Se define el gradiente de deformaciones **F** como la matriz jacobiana de la transformación entre la configuración inicial y la deformada. Se calcula del siguiente modo: [2]

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \mathrm{Grad}\varphi(\mathbf{X}) \tag{3}$$

$$\mathbf{F} = \begin{bmatrix} \frac{\partial x_1}{\partial X_1} & \frac{\partial x_1}{\partial X_2} & \frac{\partial x_1}{\partial X_3} \\ \frac{\partial x_2}{\partial X_1} & \frac{\partial x_2}{\partial X_2} & \frac{\partial x_2}{\partial X_3} \\ \frac{\partial x_3}{\partial X_1} & \frac{\partial x_3}{\partial X_2} & \frac{\partial x_3}{\partial X_3} \end{bmatrix} \tag{4}$$

El gradiente de deformaciones relaciona como se transforma una línea de la configuración inicial d**X** en la configuración deformada d**x**:

$$\mathbf{dx} = \mathbf{F}\,\mathbf{dX} \tag{5}$$

El determinante J de **F** se denomina Jacobiano de la transformación y si la transformación de coordenadas tiene propiedad de biyectividad, el jacobiano es positivo, lo que indica que la matriz **F** tiene inversa. [2]

En lo que sigue se asume que la transformación de coordenadas tiene siempre un jacobiano positivo, J > 0.

La ecuación 5 relaciona como se transforma un vector de la configuración inicial a la deformada. Las ecuaciones que relacionan áreas en la configuración inicial **dA** en dirección normal **N** y deformada **da** con dirección n (formula de Nanson), y volúmenes dv y dV son: [2]

$$\mathbf{da} = \mathbf{n}\, da = J\mathbf{F}^{-T}\mathbf{N}\, dA = J\mathbf{F}^{-T}\mathbf{dA} \tag{6}$$

$$\mathrm{dv} = \mathrm{J}\ \mathrm{dV} \tag{7}$$

2.1 Medidas de la deformación

El gradiente de deformación contiene toda la información sobre la deformación local de un cuerpo sin embargo hay ocasiones en las que otras medidas proporcionan información más conveniente. Por este motivo se han definido diferentes medidas de la deformación.

El teorema de la descomposición polar permite descomponer el vector gradiente como un alargamiento más una rotación o una rotación más un alargamiento, del siguiente modo: [2]

$$\mathbf{F} = \mathbf{R}\,\mathbf{U} = \mathbf{V}\,\mathbf{R} \tag{8}$$

Donde R es un tensor ortogonal unitario, tensor de rotación, y **U** y **V** son dos tensores simétricos definidos positivos, denominados tensor de alargamiento derecho e izquierdo si el determinante de **F** es positivo los tensores **U** y **V** tiene tres autovalores positivos, denominados alargamientos principales. Los dos tensores tienes los mismos autovalores.

La acción del tensor **R** sobre un vector diferencial consiste en rotarlo, sin modificar su magnitud. Por contra, tanto **U** como **V** actúan sobre vectores diferenciales deformándolos. [2]

Deformación de Green-Lagrange

El tensor derecho de Cauchy-Green es un tensor simétrico, definido positivo que se define del siguiente modo:

$$\mathbf{C} = \mathbf{F}^T\mathbf{F} = \mathbf{U}^T\mathbf{U} \tag{9}$$

Donde se ha considerado la descomposición polar de **F**. El tensor **U** tiene tres autovalores positivos λi con sus autovectores asociados **T**i que permite expresar: [2]

$$\mathbf{U} = \sum_{i=1}^{3} \lambda_i \, \mathbf{T}_i \otimes \mathbf{T}_i \tag{10}$$

$$\mathbf{C} = \sum_{i=1}^{3} \lambda_i^2 \, \mathbf{T}_i \otimes \mathbf{T}_i \tag{11}$$

El tensor de deformaciones de Green-Lagrange que es el que vamos a usar en el código se calcula en la configuración inicial y se define como:

$$\mathbf{E} = \frac{\mathbf{1}}{\mathbf{2}} \left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right) \tag{128}$$

2.2 Medidas de la tensión

Tensión de Cauchy

El vector de tracciones **t** se calcula como la relación entre la fuerza que aparece en la configuración deformada d**f** y el área da con dirección normal n, en la misma configuración.

$$\mathbf{t} = \frac{\mathrm{d}\mathbf{f}}{\mathrm{da}} \tag{13}$$

El tensor de tensiones de Cauchy **σ** es el correspondiente a la tracción anterior:

$$\mathbf{t} = \boldsymbol{\sigma}\,\mathbf{n} \tag{14}$$

Segundo tensor de Piola-Kirchhoff

Se puede definir el primer tensor de tensiones de Piola-Kirchhoff, **P**, como la relación fuerza en la configuración actual d**f** y el área de la configuración inicial dA que tiene dirección **N**. Aplicando la fórmula de Nanson (ecuación 6)

$$\begin{aligned} \mathrm{d}\mathbf{f} &= \boldsymbol{\sigma}\mathbf{n}da = \boldsymbol{\sigma}J\mathbf{F}^{-T}\mathbf{N}dA \\ \mathbf{P} &= \boldsymbol{\sigma}J\mathbf{F}^{-T} \end{aligned} \tag{15}$$

El tensor **P** no es simétrico y tiene más sentido definir el segundo tensor de tensiones Piola-Kirchhoff **S**, como la relación entre la fuerza transformada de la configuración actual a la inicial y el área inicial dA. [2]

$$\mathbf{F}^{-1}\mathrm{d}\mathbf{f} = \mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{n}da = J\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-T}\mathbf{N}dA$$
$$\mathbf{S} = J\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-T} \tag{16}$$

2.3 Principio de los trabajos virtuales

En elementos finitos se suele utilizar el principio de los trabajos virtuales para desarrollar las ecuaciones que permiten calcular los vectores nodales equivalentes de las fuerzas internas debidas a la deformación y las fuerzas externas.

El principio de los trabajos virtuales establece que el trabajo virtual generados por las tensiones internas es igual al trabajo virtual de las acciones externas:

$$\delta W_{\mathrm{int}}(\mathbf{x}) - \delta W_{\mathrm{ext}} = 0 \tag{17}$$

Trabajo virtual de las fuerzas internas

Los trabajos virtuales se pueden calcular en la configuración deformada o en la inicial, como se muestra a continuación, dando lugar a dos formulaciones: la formulación lagrangiana actualizada y la formulación lagrangiana total. Ambas formulaciones son equivalentes

y dan lugar a la misma solución, si se resuelve el problema de forma exacta. Sin embargo, al resolver mediante elementos finitos, la solución obtenida no es la misma debido a la inversa de la matriz jacobiana de las transformaciones que intervienen en cada caso. [2]

Trabajo virtual en la configuración inicial

Para derivar el principio de los trabajos virtuales se parte de la ecuación de equilibrio de tensiones, expresada en la configuración inicial. Asumiendo que no hay fuerzas volumétricas ni inerciales se cumple que:

$$\begin{aligned} \text{Div}\,\mathbf{P} &= 0 \\ \frac{\partial P_{11}}{\partial X} + \frac{\partial P_{12}}{\partial Y} &= 0 \\ \frac{\partial P_{21}}{\partial X} + \frac{\partial P_{22}}{\partial Y} &= 0 \end{aligned} \tag{18}$$

El trabajo virtual se obtiene integrando por partes aplicando el teorema de la divergencia e introduciendo las condiciones de contorno de Neumann. La expresión del trabajo virtual de las fuerzas internas calculado en la configuración inicial es:

$$\int_{V_0} \mathrm{Div}\mathbf{P}(\mathbf{x}) \cdot \delta\mathbf{u} dV = 0 \tag{19}$$

$$\delta W_{\mathrm{int}}(\mathbf{x}) = \int_{V_0} \mathbf{P} \cdot \mathrm{Grad}\delta\mathbf{u} dV = 0 \tag{20}$$

Operando a partir de la expresión anterior y teniendo en cuenta la definición del segundo tensor de tensiones de Piola-Kirchhoff y que es simétrico se tiene:

$$\begin{aligned} \mathbf{P} \cdot \mathrm{Grad}\delta\mathbf{u} = \mathbf{F}\,\mathbf{S} \cdot \mathrm{Grad}\delta\mathbf{u} = \mathbf{S} \cdot \mathbf{F}^T \ \mathrm{Grad}\delta\mathbf{u} \ = \\ \mathbf{S} \cdot 1/2 \,(\mathrm{Grad}^{\mathrm{T}}\delta\mathbf{u}\ \mathbf{F} + \mathbf{F}^T \ \mathrm{Grad}\delta\mathbf{u}) \end{aligned} \tag{21}$$

La expresión entre paréntesis en la ecuación anterior es la variación del tensor de tensiones de Green-Lagrange debido a los desplazamientos virtuales

$$\delta\mathbf{E} = \ 1/2 \ (Grad^T \delta\boldsymbol{u}\ \mathbf{F} + \ \mathbf{F}^T \ \mathrm{Grad}\delta\mathbf{u}) \tag{22}$$

Por lo tanto, el trabajo virtual de las fuerzas internas se escribe como:

$$\delta W_{\text{int}}(\mathbf{x}) = \int_{V_0} \mathbf{S} \cdot \delta \mathbf{E} dV \tag{23}$$

El vector de fuerzas internas se obtiene directamente de la ecuación 23, después de definir una ecuación constitutiva del material (relación entre la tensión **S** y la deformación) y de calcular la integral numéricamente. La contribución de un elemento y el trabajo virtual total se puede escribir como:

$$\begin{gathered} \delta W_{\text{int}}^{e}(\mathbf{x}) = \delta \mathbf{q}^{T} \mathbf{f}_{\text{int}}(\mathbf{x}) \\ \delta W_{\text{int}}(\mathbf{x}) = \sum_{\forall e} \delta W_{\text{int}}^{e} = \delta \mathbf{Q}^{T} \mathbf{F}_{\text{int}}(\mathbf{x}) \end{gathered} \tag{24}$$

donde **q** es la variable que contiene las coordenadas de los nodos de un elemento en la configuración deformada, **Q** las coordenadas de todos los nodos de la malla, y **F**int se obtiene ensamblando la contribución de cada elemento de la malla.

Trabajo virtual en la configuración actual

Existen dos planteamientos para derivar el trabajo virtual de las fuerzas internas en la configuración deformada. La primera es análoga al planteamiento del apartado anterior, partiendo de las

ecuaciones de equilibrio de tensiones en la configuración actual (utilizando el tensor de tensiones de Cauchy) e integrando el producto de los desplazamientos virtuales por la divergencia de la tensión. El segundo es partir de la ecuación 20 y aplicar las siguientes relaciones: [2]

$$\begin{aligned} \mathbf{P} \cdot \mathrm{Grad}\delta\mathbf{u} &= J\boldsymbol{\sigma}\mathbf{F}^{-T}\mathrm{Grad}\delta\mathbf{u} = J\boldsymbol{\sigma} \cdot \mathrm{Grad}\delta\mathbf{u}\mathbf{F}^{-1} \\ &= J\boldsymbol{\sigma} \cdot \mathrm{Grad}\delta\mathbf{u} \\ dV &= JdV_0 \end{aligned} \tag{25}$$

Teniendo en cuenta la simetría del tensor de tensiones de Cauchy y la ecuación anterior, se obtiene la siguiente expresión:

$$\delta W_{\mathrm{int}}(\mathbf{x}) = \int_{V_0} \boldsymbol{\sigma} \cdot \mathrm{grad}\delta\mathbf{u} dV = \int_{V_0} \boldsymbol{\sigma} \cdot \frac{1}{2}(\mathrm{grad}\delta\mathbf{u} + \mathrm{grad}^{\mathrm{T}}\delta\mathbf{u}) dV \tag{26}$$

Trabajo virtual de las fuerzas externas

Considerando sólo las condiciones de contorno de Nuemann, es decir, las fuerzas aplicadas en la superficie t, el trabajo virtual de las fuerzas externas se puede calcular en la configuración inicial o en la deformada. En este trabajo se asume que la tensión esta aplicada en la configuración sin deformar. [2]

$$\delta W_{\mathrm{ext}}(\mathbf{x}) = \int_{S_0} \mathbf{t} \cdot \delta\mathbf{u} dA \tag{27}$$

2.4 Modelo de Saint-Venant del material.

En el modelo de Saint-Venant la hipótesis de partida es la tensión de Piola-Kirchhoff y el tensor de deformación de Green es lineal e igual a la de pequeños desplazamientos. Este modelo es válido solo cuando las deformaciones son pequeñas, pero se puede utilizar para grandes rotaciones o desplazamientos.

Este modelo no es válido para grandes deformaciones ya que la tensión tiende a 0 cuando el volumen tiende a 0.

La fuerza interna de acuerdo a las ecuaciones 23 y 24 se define de la siguiente forma:

$$\mathbf{f}_{\text{int}} = \int_{V_0} \mathbf{S} \cdot \delta\mathbf{E} dV \tag{28}$$

Donde la relación entre **S** y **E** es la misma que en pequeños desplazamientos.

$$\mathbf{S} = 2 \cdot \mu \cdot \mathbf{E} + \lambda_d \cdot tr(\mathbf{E}) \cdot \mathbf{I}$$
$$\mu = \frac{E_y}{2(1+\nu)} \quad \lambda_d = \frac{6(1-2\cdot\nu)}{E_y} \tag{29}$$

Y E_y es el Modulo de Young y v es el coeficiente de Poisson.

La matriz tangente se define de la siguiente forma:

$$\mathbf{K}_T = \int_V (\mathrm{DP} \cdot \Delta\mathbf{u}) \cdot \mathrm{Grad}\delta\mathbf{u} dV \tag{30}$$

$$\mathrm{D}(\mathbf{F} \cdot \mathbf{S}) \cdot \Delta\mathbf{u} = \mathrm{D}(\mathbf{F}) \cdot \mathbf{S} + \mathbf{F} \cdot \mathrm{D}(\mathbf{S}) = \mathrm{Grad}\Delta\mathbf{u} \cdot \mathbf{S} + \mathbf{C} \cdot \Delta\mathbf{E}$$

$$\mathbf{K}_T = \int_V \mathrm{Grad}\Delta\mathbf{u} \cdot \mathbf{S} \cdot \mathrm{Grad}\delta\mathbf{u} dV + \int_V \delta\mathbf{E} \cdot \mathbf{C} \cdot \Delta\mathbf{E}\, dV \tag{31}$$

2.5 Modelo de Neo-Hookean del material.

En el modelo de Neo-Hookean la energía de deformación se define de tal forma que se cumpla que cuando **J** → ∞ entonces **W** → ∞, y cuando **J** → 0 la **W** → ∞. En general:

$$\mathbf{W} = g(\mathbf{J}) + \frac{1}{2} \cdot \mu \cdot (\mathbf{I}_c - 3) \tag{32}$$

Esto explica que la tensión (**σ**) → - ∞ para **J** → 0, y para **σ** → ∞.

Existen modelos para g(**J**). En este trabajo se utiliza:

$$\mathrm{g}(\mathbf{J}) = \mathrm{c}(\mathbf{J}^2 - 1) - d \cdot \ln \mathbf{J} - \mu \cdot \ln \mathbf{J}$$

$$c = \frac{\lambda}{4} \quad d = \frac{\lambda}{2} \tag{33}$$

La fuerza interna y la matriz tangente se definen de la siguiente forma:

$$\mathbf{f}_{\text{int}} = \int_{V_0} \boldsymbol{\sigma} \cdot \text{grad}\delta\mathbf{u} \cdot dV \tag{34}$$

$$\mathbf{K}_T = \int_V (\text{grad}\Delta\mathbf{u} \cdot \boldsymbol{\sigma} \cdot \text{grad}\delta\mathbf{u} + \text{grad}\delta\mathbf{u} \cdot \mathbf{C} \cdot \text{grad}\Delta\mathbf{u}) dV \tag{35}$$

Donde:

$$\boldsymbol{\sigma} = \frac{\lambda}{2J} \cdot \left(J^{\mathbf{2}} - 1\right) \cdot I + \frac{\mu}{J} \cdot (b - I) \tag{36}$$

$$b = \mathbf{F} \cdot \mathbf{F}^T$$

3. Programa cgFEM.

El termino cgFEM proviene del inglés cartesian grid finite element method, que se traduce como el método de elementos finitos de mallado cartesiano. En cgFEM la malla y la geometría son independientes. La geometría se genera a partir del modelo CAD del sistema. Normalmente se utilizan NURB's para definirla (non-uniform rational B-spline). La malla del cgFEM tiene una estructura jerárquica para hacer un mallado y refinamiento h-adaptativo más eficiente.

Esto consiste en una serie de mallas cartesianas en la que el primer Nivel (0) llamada la malla de aproximación consiste en un solo elemento que cubre todos límites que contiene el dominio de interés del objeto a estudiar. Las siguientes mallas de Nivel (i) llamadas las mallas de integración son obtenidas dividiendo el Nivel (i-1) en dos en cada dirección, por lo tanto, esta contiene $2^{d \times i}$ elementos, donde d representa la dimensión del problema.

Esto es computacionalmente eficiente porque los elementos son relacionados por una relación de paternidad y de proximidad, por lo tanto, la cantidad de información decrece. [3]

La malla final se obtiene luego de ensamblar los elementos de los diferentes niveles creados.

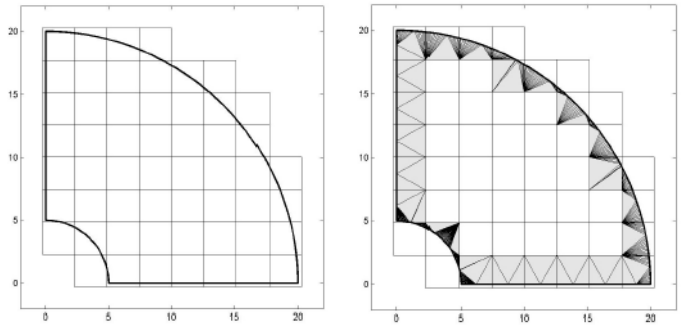


Figura 2. Lado izquierdo malla de aproximación y derecha malla de integración. [4]

Los elementos internos que están localizados en el interior del dominio son tratados como elementos finitos estandar. Los elementos de los bordes son integrados usando una cuadratura de Gauss triangular en todos los subdominios con la triangulación de Delauny como se muestra en la figura 3, o con reglas de cuadratura especiales que permiten integrar de forma exacta el dominio.

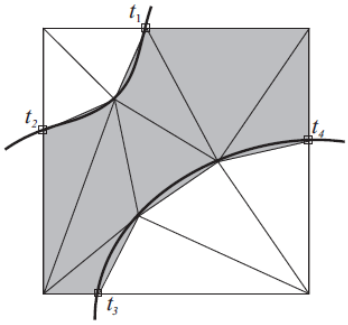


Figura 3. [4]

A continuación, se muestra un ejemplo de un objeto mallado con sus niveles respectivos.

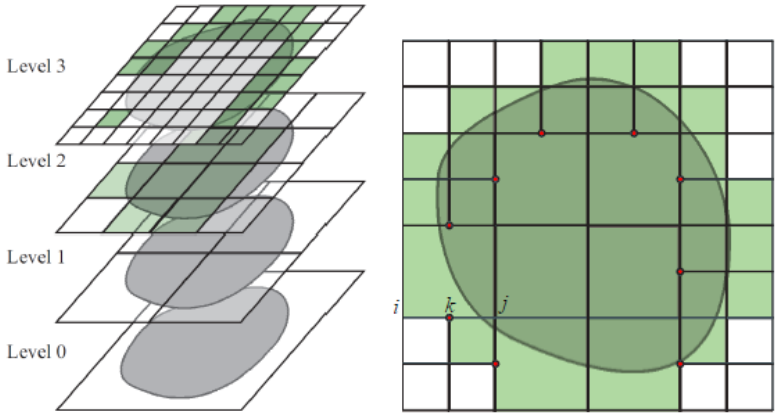


Figura 4. [5]

Hay tres opciones de refinamiento en el código de cgFEM, el primero es un refinamiento geométrico que se basa en la curvatura de los bordes del dominio, la segunda es un mallado h-adaptativo, y el tercero es uno que se utiliza solo en imágenes médicas basado en unos datos estadísticos de acuerdo a la escala de grises de los pixeles de estas.

Cuando se combinan los diferentes niveles de mallas, hay nodos de los elementos más pequeños que se encuentran en las líneas los elementos más grandes, estos nodos son llamados los “hanging nodos”. Para asegurar la continuidad del campo de desplazamientos,

estos nodos tienen que ser añadidos al sistema de ecuaciones. Estas ecuaciones restrictivas se llaman Restricciones MultiPunto (MPC).

Como el cgFEM está diseñado para 2D y luego de varios años de desarrollo y estudio por parte del Departamento de Ingeniería Mecánica de la Universidad Politécnica de Valencia y sabiendo que los problemas en la vida real son en 3D se ha desarrollado un nuevo código llamado FEAVOX.

De este código se utiliza la estructura de datos y se hacen dos cambios fundamentales:

1. Se realizan cambios en las funciones que calculan la fuerza interna y la matriz tangente, definiendo los puntos de integración, las nuevas deformaciones y tensiones para sus respectivos ensamblados.
2. Método de resolución de un sistema no lineal mediante Newton-Raphson. Se determina el vector fuerza interna $\mathbf{F}_{\mathbf{int}}(\mathbf{x})$ y el vector de fuerza externa $\mathbf{F}_{\mathbf{ext}}(\mathbf{x})$.

   El residuo de la ecuación $\mathbf{R}(\mathbf{x})$ se define:

$$\mathbf{R}(\mathbf{x}) = \mathbf{F}_{\mathbf{int}(\mathbf{x})} - \mathbf{F}_{\mathbf{ext}(\mathbf{x})} \tag{37}$$

   Conociendo la posición inicial **x** se procede al cálculo iterativo del vector:

$$\mathbf{R(x) = R(x_i) + \frac{\partial R(x_0)}{\partial x} \cdot \Delta x} \tag{38}$$

$$\mathbf{\Delta x = -\left[\frac{\partial R(x_0)}{\partial x}\right]^{-1} R(x_i)} \tag{39}$$

$$\mathbf{x_{i+1} = x_i + \Delta x} \tag{40}$$

El procedimiento se repite hasta que se produce la convergencia. El criterio para verificar la convergencia es en base a calcular la norma de residuo **R**, que se define como el equilibrio de la fuerza en los nodos**.** Se normaliza el residuo utilizando la norma del vector de fuerzas externas.

$$\mathbf{\frac{\|R(x)\|}{\|F_{ext}\|}} < Tolerancia \tag{41}$$

4. Implementación de grandes desplazamientos en cgFEM.

4.1 Desarrollo del código en Matlab.

Se ha desarrollado un código para un elemento cuadrilátero con comportamiento elástico no lineal. Para lo cual se crea una carpeta que contenga los programas para el código principal y los datos necesarios para la resolución del problema.

En primer lugar, se definen las coordenadas de los puntos en un archivo llamado XYZ, luego se definen los números de nodo en otro archivo llamado TOP y las condiciones de Dirchlet en el archivo DIR.

Por ejemplo, para resolver un problema con un solo elemento como el representado en la figura 5, se definen los siguientes ficheros de datos:

DIR		
NODO	DIRECCION	RESTRICCION
1	1	0
1	2	0
2	2	0
4	1	0

XYZ	
X	Y
0.0	0.0
2.0	0.0
2.0	1.0
0.0	1.0

TOP			
1	2	3	4

TABLA 1, 2 Y 3

La dirección 1 es para la coordenada x y el número 2 para la coordenada y.

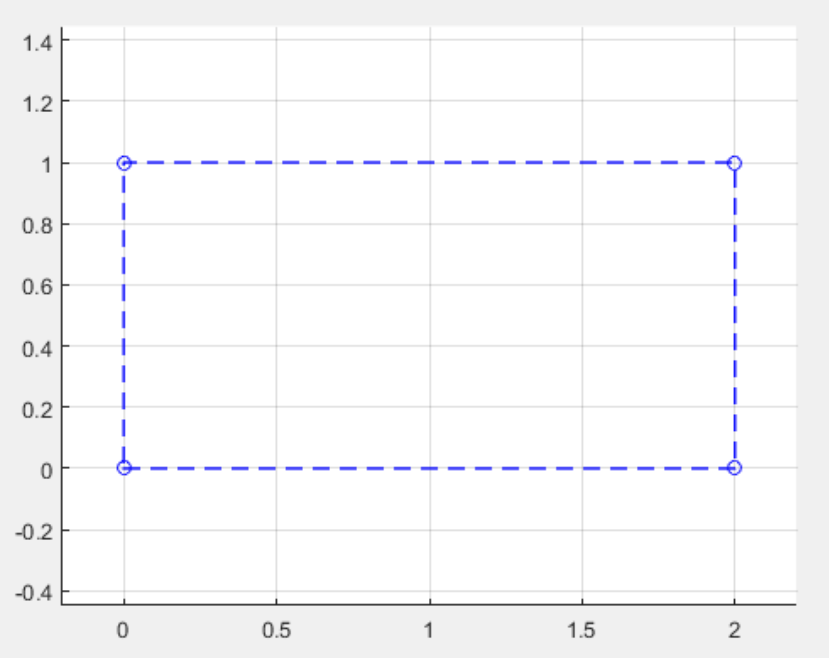


FIGURA 5

La figura 5 representa el cuadrilátero dibujado en MATLAB.

El código principal está compuesto de varias funciones que las vamos a ir detallando una por una. [6]

4.2 Función sd2Dsolid

Este es el programa central donde se va a resolver el problema de grandes desplazamientos. Está compuesto por otras funciones que se explican luego de esta.

Se empieza cargando los grados de libertad, leyendo los ficheros XYZ, TOP y leyendo las condiciones de Dirichlet. También se dibuja por primera vez la posición sin deformar

Los datos de salida van a ser los desplazamientos, la norma del residuo y la figura deformada.

```
%-- Grados de libertad por nodo
Ngdl = 2;

%-- Lee la malla de elementos finitos (TOP,
XYZ)
load TOP
load XYZ

%-- Número de elementos: filas del fichero
TOP
Ne = size(TOP,1);
%-- Número de nodos: filas del fichero XYZ
Nn = size(XYZ,1);


%-- Leer condiciones de Dirichlet
```

```
%-- Genera el vector gdlA que contiene los
grados de libertad activos
load DIR
gdlA(1:Ngdl*Nn) = 1;
for i=1:size(DIR,1)
    node = DIR(i,1);
    dir = DIR(i,2);
    gdlA((node-1)*Ngdl + dir ) = 0;
end
gdlA=gdlA>0;

%-- Dibuja posición sin deformar
dibuja_cuad(1,TOP,XYZ,1)
axis equal
```

Luego se asigna manualmente las fuerzas externas a aplicar, indicando el nodo y la dirección, como por ejemplo:

```
Nodo = 2;
Gdl=1;
Fex(Ngdl*(Nodo-1)+Gdl) = -10;
Nodo = 3;
Gdl=1;
Fex(Ngdl*(Nodo-1)+Gdl) = 20;
```

Luego viene la resolución del sistema aplicando Newton-Raphson.

Se hace el cálculo del vector de fuerzas internas debido a las deformaciones elásticas.

```
U = XYZ*0;
XYZ_def = XYZ;
```

```
Fint =
fint_sd4Nsolid(TOP,XYZ,U,E,nu,XYZ_def,lambd
a,G);
```

La variable XYZ_def toma un nuevo valor para cada iteración, y es donde se almacena la nueva posición deformada del elemento.

Se calcula el residuo entre la fuerza interna y fuerza externa.

```
R = Fint - Fex;
```

Se almacena la norma del residuo en la variable nr.

```
disp('Norma del residuo:')
nr_ini = norm(R);
nr = nr_ini;
nr
```

```
Iter = 1;
Q = reshape(XYZ',[],1);
XYZ_def = XYZ;
Uv = reshape(U,[],1);
```

Para la iteración se usa un bucle while donde se evaluará la norma del residuo. Se realizará la solución para el sistema de ecuaciones, se actualizarán los desplazamientos y las posiciones de los nodos, y luego se volverá a calcular la norma del residuo.
Cuando el resultado haya convergido se procederá a dibujar el elemento deformado.

```
while ((nr/nr_ini > 1e-8) && (Iter<10))
    %-- Matriz tangente ya ensamblada
    K =
K_sd4Nsolid(TOP,XYZ,E,nu,XYZ_def,lambda,G);

    %-- Solución sistema de ecuaciones
    DQ = - K(gdlA,gdlA) \ R(gdlA);

    %-- Desplazamientos
    Uv(gdlA) = DQ;
    U = reshape(Uv,Ngdl,Nn)';

    %-- Actualizar posición
    Q(gdlA) = Q(gdlA) + DQ;
    XYZ_def = reshape(Q,Ngdl,Nn)';

    %-- Cálculo del residuo con la solución
modificada

Fint=fint_sd4Nsolid(TOP,XYZ,U,E,nu,XYZ_def,
lambda,G);
    R = Fint - Fex;

    Iter
    disp('     - Norma del residuo:')
    nr = norm(R(gdlA));
    nr/nr_ini


    Iter = Iter + 1;

    dibuja_cuad(1,TOP,XYZ_def,1)
```

```
    pause
end
dibuja_cuad(1,TOP,XYZ_def,1)
end
```

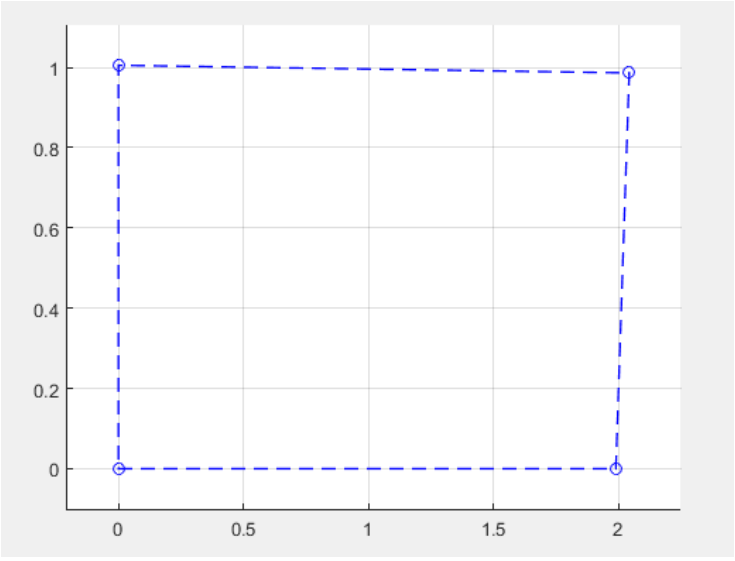


Figura 6.

4.3 Función ShapeFunc4N_loc.

Datos de entrada: ξ y η.

Datos de salida: $N = [N1 \quad N2 \quad N3 \quad N4]$.

Esta función define las funciones de forma del elemento en coordenadas locales psi y eta.

N1 = (1 - ξ)(1 — η)/4

N2 = (1 + ξ)(1 — η)/4

N3 = (1 + ξ)(1 + η)/4

N4 = (1 - ξ)(1 + η)/4

4.4 Funcion derN

Datos de entrada: ξ y η.

Datos de salida: $dN = \begin{bmatrix} \frac{\partial N1}{\partial \xi} & \frac{\partial N2}{\partial \xi} & \frac{\partial N3}{\partial \xi} & \frac{\partial N4}{\partial \xi} \\ \frac{\partial N1}{\partial \eta} & \frac{\partial N2}{\partial \eta} & \frac{\partial N3}{\partial \eta} & \frac{\partial N4}{\partial \eta} \end{bmatrix}$.

Esta función define las derivadas de las funciones de forma respecto de las coordenadas locales.

$$\frac{\partial N1}{\partial \xi} = -(1 - \eta)/4$$

$$\frac{\partial N2}{\partial \xi} = (1 - \eta)/4$$

$$\frac{\partial N3}{\partial \xi} = (1 + \eta)/4$$

$$\frac{\partial N4}{\partial \xi} = -(1+\eta)/4$$

$$\frac{\partial N1}{\partial \eta} = -(1-\xi)/4$$

$$\frac{\partial N2}{\partial \eta} = -(1+\xi)/4$$

$$\frac{\partial N3}{\partial \eta} = (1+\xi)/4$$

$$\frac{\partial N4}{\partial \eta} = (1-\xi)/4$$

4.5 Función Jaco

Datos de entrada: TOP, XYZ, ξ, η, ie.

Datos de salida: J y dN_xy.

Esta función calcula la matriz jacobiana de la transformación de coordenadas locales a globales. La matriz jacobiana viene dada por:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathrm{x}}{\partial \xi} & \frac{\partial \mathrm{y}}{\partial \xi} \\ \frac{\partial \mathrm{x}}{\partial \eta} & \frac{\partial \mathrm{y}}{\partial \eta} \end{bmatrix}$$

Donde:

$$\frac{\partial x}{\partial \xi} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial \xi} \cdot x_i$$

$$\frac{\partial y}{\partial \xi} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial \xi} \cdot y_i$$

$$\frac{\partial x}{\partial \eta} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial \eta} \cdot x_i$$

$$\frac{\partial y}{\partial \eta} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial \eta} \cdot y_i$$

En esta función también se calculan las derivadas de las funciones de forma respecto de las coordenadas globales a través de la inversa del Jacobiano.

$$\begin{Bmatrix} \frac{\partial N_i}{\partial \mathrm{x}} \\ \frac{\partial N_i}{\partial \mathrm{y}} \end{Bmatrix} = \mathbf{J}^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix}$$

Donde:

$$dN_{xy} = \begin{bmatrix} \frac{\partial N1}{\partial x} & \frac{\partial N2}{\partial x} & \frac{\partial N3}{\partial x} & \frac{\partial N4}{\partial x} \\ \frac{\partial N1}{\partial y} & \frac{\partial N2}{\partial y} & \frac{\partial N3}{\partial y} & \frac{\partial N4}{\partial y} \end{bmatrix}$$

En el código se obtiene primero vectores columna de las componentes X y Y del archivo XYZ.

```
X = XYZ(TOP(ie,:),1);
Y = XYZ(TOP(ie,:),2);
```

Luego se obtiene los elementos del Jacobiano, para luego con su inversa obtener las derivadas de las funciones de forma respecto a las coordenadas globales.

```
J(1,1) = dN(1,:)*X;
J(1,2) = dN(2,:)*X;
J(2,1) = dN(1,:)*Y;
J(2,2) = dN(2,:)*Y;

iJ = inv(J);

for k=1:4
    dN_XY(:,k) = iJ*dN(:,k);
end
```

4.6 Función dibuja_cuad

Es la función que dibuja los elementos y trabaja con los siguientes valores.

dibuja_cuad(nf,TOP,XYZ)

Datos de entrada:

- nf: Número de la figura
- TOP: elementos de la malla

- XYZ: coordenadas de los nodos

- op: 1 - Borra la figura

2 - No borra la figura

Datos de Salida:

Imagen del elemento.

4.7 Modelo de Saint-Venant del material.

4.7.1 Función K_sd4NSolid

Datos de entrada: TOP, XYZ, E, ν, XYZ_def, λ, μ

Datos de salida: K_T.

Calcula la matriz tangente de cada elemento y la ensambla para formar la matriz K global.

Primero se define la matriz del material D que está en deformación plana:

$$\mathbf{D} = \frac{1}{(1+\nu)(1-2\nu)} \begin{bmatrix} E(1-\nu) & E\nu & 0 \\ E\nu & E(1-\nu) & 0 \\ 0 & 0 & \dfrac{E(1-2\nu)}{2} \end{bmatrix}$$

En el caso 3D esta matriz es:

$$\mathbf{D} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \nu/(1-\nu) & \nu/(1-\nu) & 0 & 0 & 0 \\ \nu/(1-\nu) & 1 & \nu/(1-\nu) & 0 & 0 & 0 \\ \nu/(1-\nu) & \nu/(1-\nu) & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & (1-2\nu)/2(1-\nu) & 0 & 0 \\ 0 & 0 & 0 & 0 & (1-2\nu)/2(1-\nu) & 0 \\ 0 & 0 & 0 & 0 & 0 & (1-2\nu)/2(1-\nu) \end{bmatrix}$$

Donde E es el módulo de Young y ν es el módulo de Poisson. Se definen los puntos de integración para nuestro cuadrilátero

```
ng = 3;
wg = [5/9 8/9 5/9];
xg = [-sqrt(3/5) 0 sqrt(3/5)];
```

Luego se define el gradiente de deformaciones. En el código se obtiene primero vectores columna de las componentes X y Y del archivo XYZ_def que contiene la configuración deformada.

```
x = XYZ_def(TOP(ie,:),1);
y = XYZ_def(TOP(ie,:),2);
```

La matriz XYZ_def es la matriz donde se van almacenando los desplazamientos de los nodos.

```
F(1,1) = dN_XY(1,:)*x;
F(1,2) = dN_XY(2,:)*x;
F(2,1) = dN_XY(1,:)*y;
F(2,2) = dN_XY(2,:)*y;
```

Luego se obtiene la deformación de Green Lagrange

```
Eg = 1/2*(F'*F - I);
```

Donde I es la matriz identidad. Luego se obtiene la tensión de Piola-Kirchoff.

```
S = lambda*sum(diag(Eg))*I + 2*G*Eg;
```

Donde lambda y G (μ) se obtienen de la siguiente manera

```
lambda = E*ν/(1+ν)/(1-2*ν);
G = E/2/(1+ ν);
```

Luego se obtiene la matriz B_L

$$\mathbf{B\_L} = \begin{bmatrix} F_{11}N_{k,1} & F_{21}N_{k,1} \\ F_{12}N_{k,2} & F_{22}N_{k,2} \\ F_{11}N_{k,2} + F_{12}N_{k,1} & F_{21}N_{k,2} + F_{22}N_{k,1} \end{bmatrix}$$

Para nuestro caso del cuadrilátero las líneas del código serian:

B_L=[F(1,1)*dN_XY(1,1) F(2,1)*dN_XY(1,1) F(1,1)*dN_XY(1,2)
F(2,1)*dN_XY(1,2) F(1,1)*dN_XY(1,3) F(2,1)*dN_XY(1,3)
F(1,1)*dN_XY(1,4) F(2,1)*dN_XY(1,4) ; ...

F(1,2)*dN_XY(2,1) F(2,2)*dN_XY(2,1) F(1,2)*dN_XY(2,2)
F(2,2)*dN_XY(2,2) F(1,2)*dN_XY(2,3) F(2,2)*dN_XY(2,3)
F(1,2)*dN_XY(2,4) F(2,2)*dN_XY(2,4) ; ...

F(1,1)*dN_XY(2,1) + F(1,2)*dN_XY(1,1) F(2,1)*dN_XY(2,1) +
F(2,2)*dN_XY(1,1) F(1,1)*dN_XY(2,2) + F(1,2)*dN_XY(1,2)
F(2,1)*dN_XY(2,2) + F(2,2)*dN_XY(1,2) ...
F(1,1)*dN_XY(2,3) + F(1,2)*dN_XY(1,3) F(2,1)*dN_XY(2,3) +
F(2,2)*dN_XY(1,3) F(1,1)*dN_XY(2,4) + F(1,2)*dN_XY(1,4)
F(2,1)*dN_XY(2,4) + F(2,2)*dN_XY(1,4)];

La matriz G:

```
for im=1:4;
  for km=1:4;
    Gik = dN_XY(:,im)'*S*dN_XY(:,km);
    MG(2*im-1:2*im,2*km-1:2*km)=eye(2)*Gik;
  end
end
```

La matriz tangente del elemento se calculará de la siguiente forma:

```
K_el = K_el + wg(ig)*wg(jg)*(MG+B_L'*D*B_L)*det(J);
```

Luego se procede al ensamblado de la matriz.

```
% Grados de libertad del elemento
gdle = [2*TOP(ie,1)-1 2*TOP(ie,1)
2*TOP(ie,2)-1 2*TOP(ie,2) 2*TOP(ie,3)-1
2*TOP(ie,3) 2*TOP(ie,4)-1 2*TOP(ie,4)]';

gdlfe = reshape(gdle*ones(1,8),[],1);
gdlce = reshape(ones(8,1)*gdle',[],1);
K_el = reshape(K_el,[],1);
K = [K; K_el];
gdlf = [gdlf; gdlfe];
gdlc = [gdlc; gdlce];

% Ensamblado de la matriz
K = sparse(gdlf,gdlc,K);
```

4.7.2 Función fint_sd4Nsolid

Datos de entrada: TOP, XYZ, E, ν, XYZ_def.

Datos de salida: F_{int}

Calcula el vector de fuerzas internas de cada elemento y lo ensambla para formar el vector de Fuerza interna global

Primero se define la matriz del material D que está en deformación plana:

$$\mathbf{D} = \frac{1}{(1+\nu)(1-2\nu)}\begin{bmatrix} E(1-\nu) & E\nu & 0 \\ E\nu & E(1-\nu) & 0 \\ 0 & 0 & \dfrac{E(1-2\nu)}{2} \end{bmatrix}$$

En el caso 3D esta matriz es:

$$\mathbf{D} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\begin{bmatrix} 1 & \nu/(1-\nu) & \nu/(1-\nu) & 0 & 0 & 0 \\ \nu/(1-\nu) & 1 & \nu/(1-\nu) & 0 & 0 & 0 \\ \nu/(1-\nu) & \nu/(1-\nu) & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & (1-2\nu)/2(1-\nu) & 0 & 0 \\ 0 & 0 & 0 & 0 & (1-2\nu)/2(1-\nu) & 0 \\ 0 & 0 & 0 & 0 & 0 & (1-2\nu)/2(1-\nu) \end{bmatrix}$$

Donde E es el módulo de Young y ν es el módulo de Poisson.

Se definen los puntos de integración para nuestro cuadrilátero

```
ng = 3;
wg = [5/9 8/9 5/9];
```

```
xg = [-sqrt(3/5) 0 sqrt(3/5)];
```

Luego se define el vector de desplazamientos U_el del elemento.

```
Uel=[U(TOP(ie,1),1) U(TOP(ie,1),2)
U(TOP(ie,2),1) U(TOP(ie,2),2)
U(TOP(ie,3),1) U(TOP(ie,3),2)
U(TOP(ie,4),1) U(TOP(ie,4),2)]';
```

Luego se define el gradiente de deformaciones. En el código se obtiene primero vectores columna de las componentes X y Y del archivo XYZ_def que contiene la configuración deformada.

```
x = XYZ_def(TOP(ie,:),1);
y = XYZ_def(TOP(ie,:),2);
```

La matriz XYZ_def es la matriz donde se van almacenando los desplazamientos de los nodos.

```
F(1,1) = dN_XY(1,:)*x;
F(1,2) = dN_XY(2,:)*x;
F(2,1) = dN_XY(1,:)*y;
F(2,2) = dN_XY(2,:)*y;
```

Luego se obtiene la deformación de Green Lagrange

```
Eg = 1/2*(F'*F - I);
```

Donde I es la matriz identidad. Luego se obtiene la tensión de Piola-Kirchoff.

```
S = lambda*sum(diag(Eg))*I + 2*G*Eg;
Sv = [S(1,1);S(2,2);S(1,2)];
```

Donde lambda y G (μ) se obtienen de la siguiente manera

```
lambda = E*ν/(1+ν)/(1-2*ν);
G = E/2/(1+ ν);
```

Luego se obtiene la matriz B_L

$$\mathbf{B\_L} = \begin{bmatrix} F_{11}N_{k,1} & F_{21}N_{k,1} \\ F_{12}N_{k,2} & F_{22}N_{k,2} \\ F_{11}N_{k,2} + F_{12}N_{k,1} & F_{21}N_{k,2} + F_{22}N_{k,1} \end{bmatrix}$$

Para nuestro caso del cuadrilátero las líneas del código serian:

```
B_L=[F(1,1)*dN_XY(1,1)  F(2,1)*dN_XY(1,1)  F(1,1)*dN_XY(1,2)
F(2,1)*dN_XY(1,2)     F(1,1)*dN_XY(1,3)  F(2,1)*dN_XY(1,3)
F(1,1)*dN_XY(1,4)  F(2,1)*dN_XY(1,4)  ; ...

F(1,2)*dN_XY(2,1)  F(2,2)*dN_XY(2,1)  F(1,2)*dN_XY(2,2)
F(2,2)*dN_XY(2,2)  F(1,2)*dN_XY(2,3)  F(2,2)*dN_XY(2,3)
F(1,2)*dN_XY(2,4)  F(2,2)*dN_XY(2,4)  ; ...

F(1,1)*dN_XY(2,1) + F(1,2)*dN_XY(1,1) F(2,1)*dN_XY(2,1) +
F(2,2)*dN_XY(1,1)  F(1,1)*dN_XY(2,2) + F(1,2)*dN_XY(1,2)
F(2,1)*dN_XY(2,2) + F(2,2)*dN_XY(1,2) ...
F(1,1)*dN_XY(2,3) + F(1,2)*dN_XY(1,3) F(2,1)*dN_XY(2,3) +
F(2,2)*dN_XY(1,3)  F(1,1)*dN_XY(2,4) + F(1,2)*dN_XY(1,4)
F(2,1)*dN_XY(2,4) + F(2,2)*dN_XY(1,4)];
```

El vector de fuerza interna de un elemento se calculará de la siguiente forma:

```
fel_el = fel_el +
wg(ig)*wg(jg)*det(J)*B_L'*Sv;
```

Luego se procede al ensamblado del vector

```
% Grados de libertad del elemento
gdle = [2*TOP(ie,1)-1 2*TOP(ie,1)
2*TOP(ie,2)-1 2*TOP(ie,2) 2*TOP(ie,3)-1
2*TOP(ie,3) 2*TOP(ie,4)-1 2*TOP(ie,4)]';


fel = [fel; fel_el];
gdl = [gdl; gdle];

% Ensamblado del vector
Fint = accumarray(gdl,fel);
```

4.8 Modelo de Neo-Hookean del material.

4.8.1 Función K_Id4Nsolid

Datos de entrada: TOP, XYZ, E, v, XYZ_def.

Datos de salida: K_T.

Calcula la matriz tangente de cada elemento y la ensambla para formar la matriz K global.

Se definen los puntos de integración para nuestro cuadrilátero

```
ng = 3;
```

```
wg = [5/9 8/9 5/9];
xg = [-sqrt(3/5) 0 sqrt(3/5)];
```

Se definen los nodos de las posiciones deformadas y sin deformar del elemento.

```
% Coordenadas de los nodos del elemento
Xe = XYZ(node,:);

% Posición deformada de los nodos del elemento
xe = XYZ_def(node,:);
```

La matriz XYZ_def es la matriz donde se van almacenando los desplazamientos de los nodos.

Se definen la matriz F y B_L.

```
F = eye(2);
            for k=1:4
                dN_x(:,k) = J\dN(:,k);
                F = F - (xe(k,:)-
Xe(k,:))'*dN_x(:,k)';
            end
            F = inv(F);
            detJ = det(F);
            detj = det(J);

            for k=1:4
                B_L(1,2*k-1) = dN_x(1,k);
                B_L(1,2*k) = 0;
                B_L(2,2*k-1) = 0;
                B_L(2,2*k) = dN_x(2,k);
                B_L(3,2*k-1) = dN_x(2,k);
                B_L(3,2*k) = dN_x(1,k);
            end
```

Se define el tensor de tensiones de Cauchy

```
s = lambda/(2*detJ)*(detJ^2-1)*eye(2) +
Gm/detJ*(F*F'-eye(2));
sv = [s(1,1); s(2,2); s(1,2)];

for k=1:4
for i=1:4
Gik = dN_x(:,i)'*s*dN_x(:,k);
G(2*i-1:2*i,2*k-1:2*k) = Gik*eye(2);
end
end
```

Se define la matriz del material.

```
D = 1/detJ*[2*Gm+lambda, lambda*detJ^2, 0;
lambda*detJ^2, 2*Gm+lambda, 0; 0, 0, Gm-
lambda*(detJ^2-1)/2];
```

La matriz tangente del elemento se calculará de la siguiente forma:

```
K_el = K_el + wg(ig)*wg(jg)*(B_L'*D*B_L + G)*detj;
```

Luego se procede al ensamblado de la matriz.

```
% Grados de libertad del elemento
aux = [2*node'-1  2*node'];
gdle = reshape(aux',[],1);
gdlfe = reshape(gdle*ones(1,8),[],1);
gdlce = reshape(ones(8,1)*gdle',[],1);
K_el = reshape(K_el,[],1);
K = [K; K_el];
gdlf = [gdlf; gdlfe];
gdlc = [gdlc; gdlce];

% Ensamblado de la matriz
K = sparse(gdlf,gdlc,K);
```

4.8.2 Función fint_ld4Nsolid

Datos de entrada: TOP, XYZ, E, v, XYZ_def.

Datos de salida: F_{int}.

Calcula el vector de fuerzas internas de cada elemento y lo ensambla para formar el vector de Fuerza interna global

Se definen los puntos de integración para nuestro cuadrilátero

```
ng = 3;
wg = [5/9 8/9 5/9];
xg = [-sqrt(3/5) 0 sqrt(3/5)];
```

Se definen los nodos de las posiciones deformadas y sin deformar del elemento.

```
% Coordenadas de los nodos del elemento
Xe = XYZ(node,:);

% Posición deformada de los nodos del elemento
xe = XYZ_def(node,:);
```

La matriz XYZ_def es la matriz donde se van almacenando los desplazamientos de los nodos.

Se definen la matriz F y B_L.

```
F = eye(2);
            for k=1:4
                dN_x(:,k) = J\dN(:,k);
                F = F - (xe(k,:)-
Xe(k,:))'*dN_x(:,k)';
            end
            F = inv(F);
```

```
detJ = det(F);
detj = det(J);

for k=1:4
    B_L(1,2*k-1) = dN_x(1,k);
    B_L(1,2*k) = 0;
    B_L(2,2*k-1) = 0;
    B_L(2,2*k) = dN_x(2,k);
    B_L(3,2*k-1) = dN_x(2,k);
    B_L(3,2*k) = dN_x(1,k);
end
```

Se define el tensor de tensiones de Cauchy

```
s = lambda/(2*detJ)*(detJ^2-1)*eye(2) +
Gm/detJ*(F*F'-eye(2));
sv = [s(1,1); s(2,2); s(1,2)];
```

El vector de fuerzas internas del elemento se calculará de la siguiente forma:

```
fel_el = fel_el + wg(ig)*wg(jg)*B_L'*sv*detj ;
```

Luego se procede al ensamblado.

```
% Grados de libertad del elemento
aux = [2*node'-1  2*node'];
gdle = reshape(aux',[],1);
fel = [fel; fel_el];
gdl = [gdl; gdle];

% Ensamblado del vector:
Fint = accumarray(gdl,fel);
```

4.9 Código en cgFEM

El código desarrollado genera un problema que afecta a los puntos de integración, esto se debe a que en el programa de cgFEM la malla no es parte del objeto como en la teoría de elementos finitos.

Es por esto que el código de cgFEM ha tenido que desarrollar nuevas ecuaciones basadas en los MPC explicados en el capítulo 3 con tal de asegurar la continuidad en el campo de desplazamientos.

Al interpolar los nodos en MPC genera errores que dificultan la convergencia del problema.

De cgFEM se utiliza:

- Puntos de integración y coordenadas ξ y η.
- Pesos de integración.
- Procedimiento de ensamblado.

Se modifica

- Calculo de f_{int} y k_t.
- Método de resolución iterativo de Newton-Raphson.

5. Ejemplos y comparación con ANSYS

Para la adaptación al programa de FEAVox se va a usar los siguientes ejemplos:

5.1 Cubo.

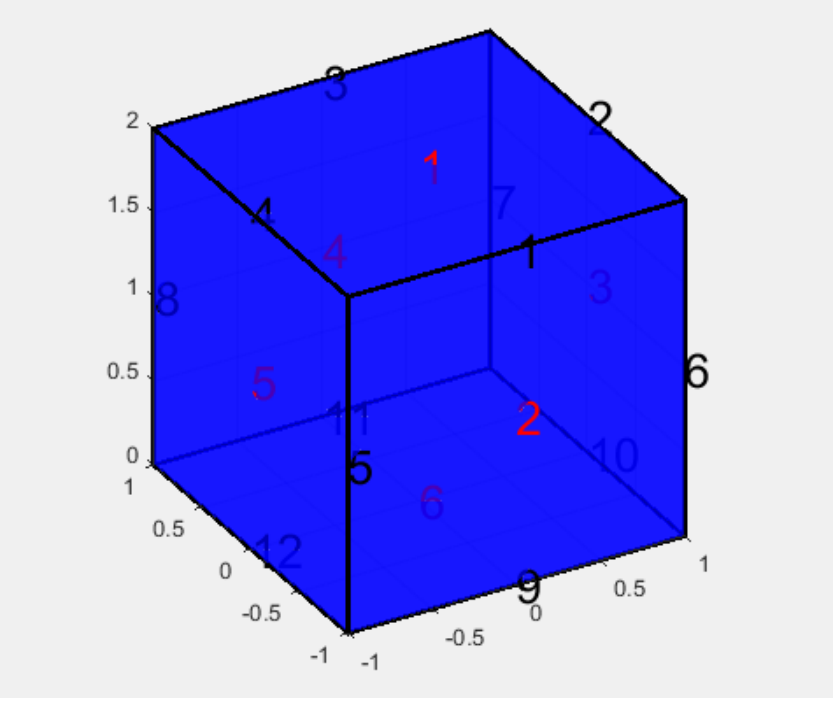


Figura 7.

Las medidas en este caso fueron de 2x2x2 (m^3). Se tomó el módulo de Young en 1000 (Pa) y el módulo de Poisson de 0.3. Para la simulación se lo hizo en 20 pasos de carga.

En la cara 1 se restringen los desplazamientos en dirección z, en la cara 2 en la dirección y, y en la cara 5 en dirección x. Se asignó una presión de tracción de 200 (Pa) en la cara 3 en dirección x como se muestra en la figura.

Saint Venant.

A continuación, se muestran las figuras que representan la solución de elementos finitos en cgFEM en el último paso de carga.

En la figura 8 se observa el desplazamiento en dirección x. El valor máximo es de 0.3194 (m).

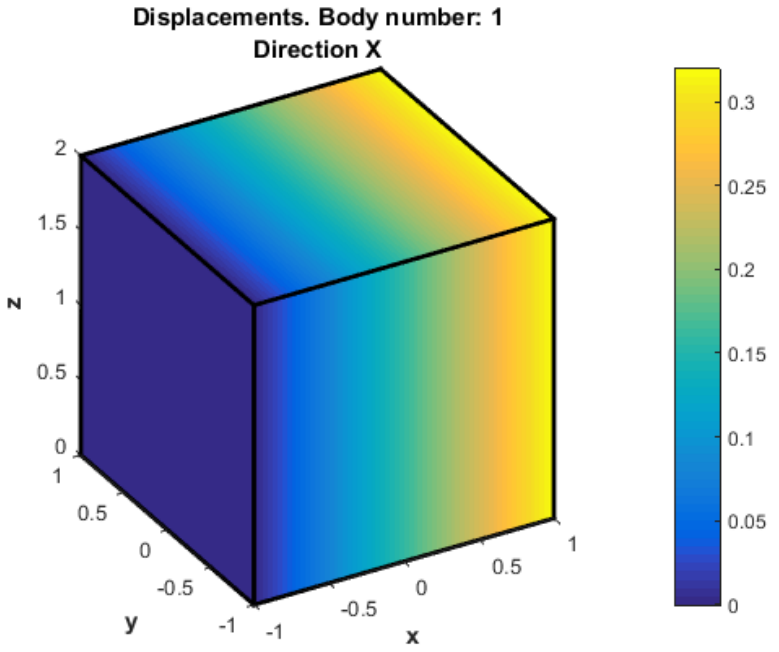


Figura 8.

En la figura 9 se presenta el elemento deformado y sin deformar.

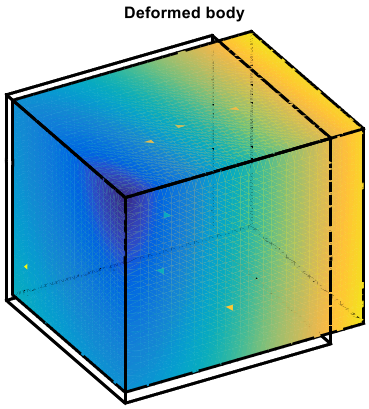


Figura 9.

En la figura 10 la tensión máxima en dirección x es de 223 (Pa).

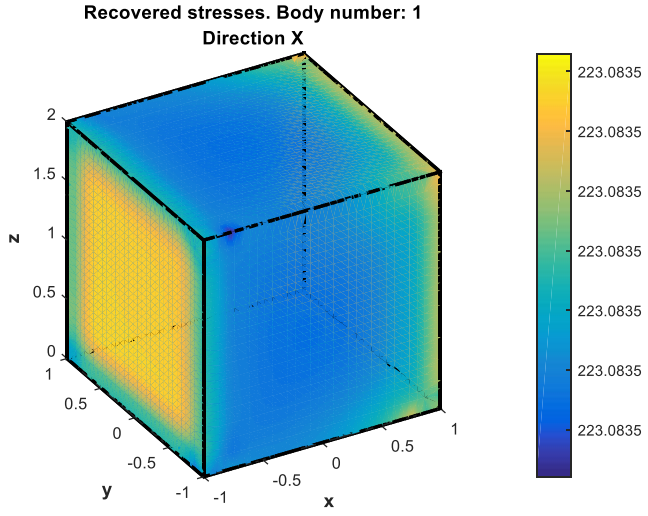


Figura 10.

Neo-Hookean.

En la figura 11 se observa el desplazamiento en dirección x. El valor máximo es de 0.463 (m).

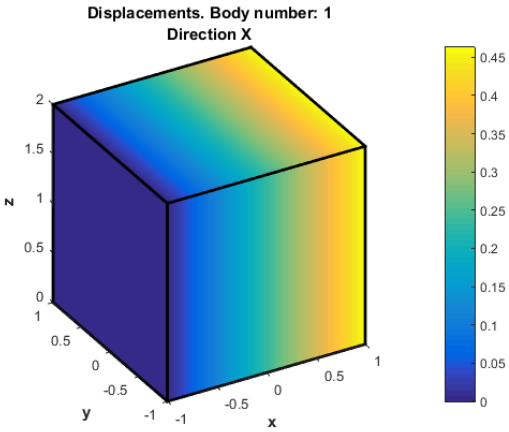


Figura 11.

En la figura 12 se presenta el elemento deformado y sin deformar.

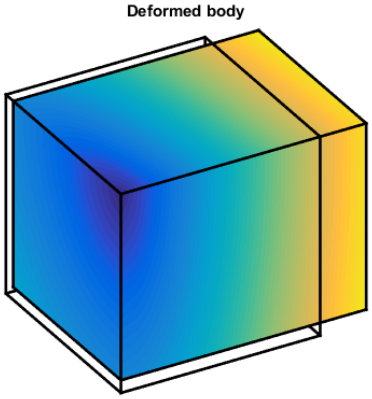


Figura 12.

En la figura 13 la tensión máxima en dirección x es de 320 (Pa).

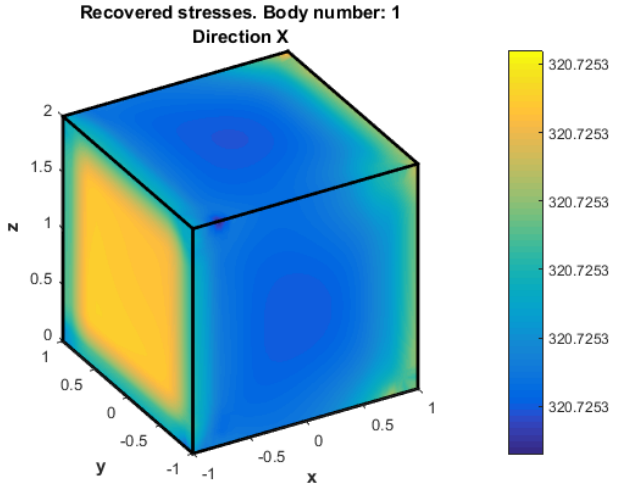


Figura 13.

ANSYS.

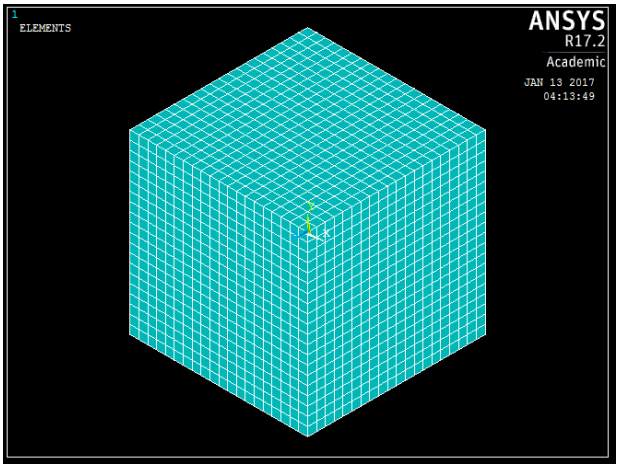


Figura 14.

Para el modelo de Ansys las medidas en este caso fueron de 2x2x2 (m^3). Se usó un tipo de elemento tipo SOLID185. Para el mallado se usó elementos de tamaño 0.1. Se tomó el módulo de Young en 1000 (Pa) y el módulo de Poisson de 0.3.

Se obtuvo un total de 8000 elementos y para la simulación se lo hizo en 20 pasos de carga.

Las restricciones se hicieron sobre la cara lateral izquierda, en dirección X, en la cara inferior en Y, en la cara frontal en Z. Se asignó una presión de tracción de 200 (Pa) en la cara lateral derecha como se muestra en la figura 15.

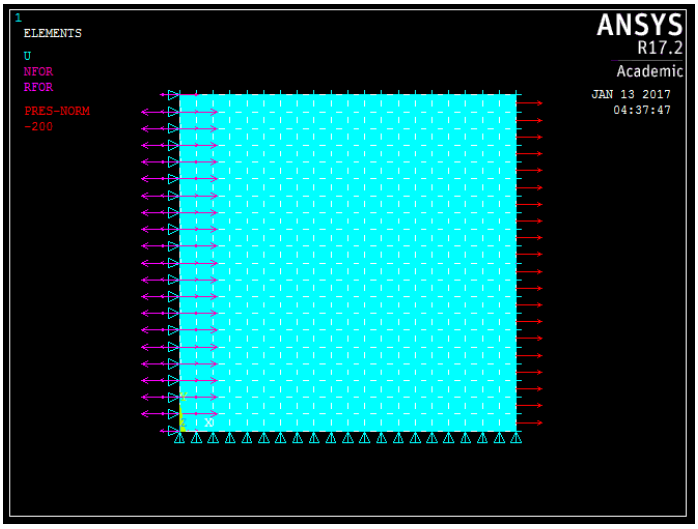


Figura 15.

En la figura 16 se presenta el elemento deformado y sin deformar.

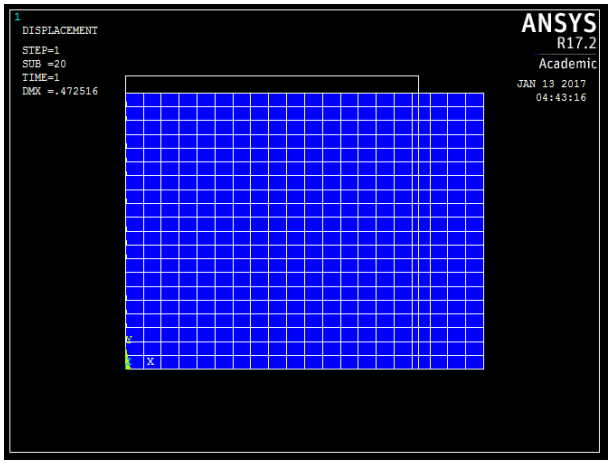


Figura 16.

En la figura 17 se observa el desplazamiento en dirección x. El valor máximo es de 0.443 (m).

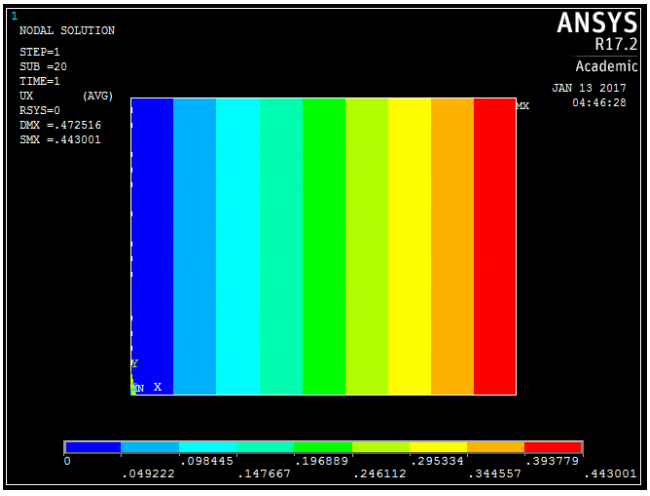


Figura 17.

En la figura 18 la tensión máxima en dirección x es de 200 (Pa).

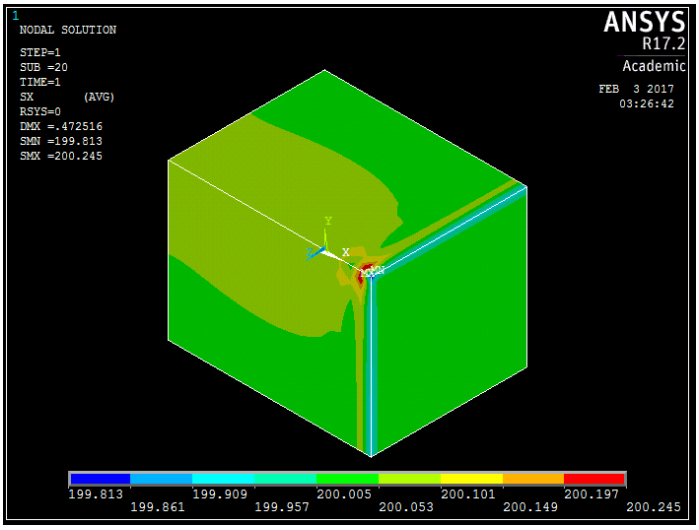


Figura 18.

A continuación, se muestra en forma de grafica una comparación de los resultados obtenidos en cgFEM con los modelos de Saint Venant, Neo-Hookean y Ansys.

En la figura 19 se muestra el desplazamiento en direccion x versus la presión de tracción aplicada. Se puede ver que el modelo de Neo-Hookean y Ansys son los que más se parecen teniendo una diferencia menor del 10%.

Las diferencias son debidas a dos factores:

(i) La utilización de un modelo diferente de comportamiento hiperélastico del material.

(ii) En cgFEM la tensión externa se aplica en la superficie sin deformar mientras que ANSYS lo hace en la superficie deformada. Como en este caso el área se reduce con la deformación, la fuerza total aplicada en ANSYS es un poco menor que en cgFEM.

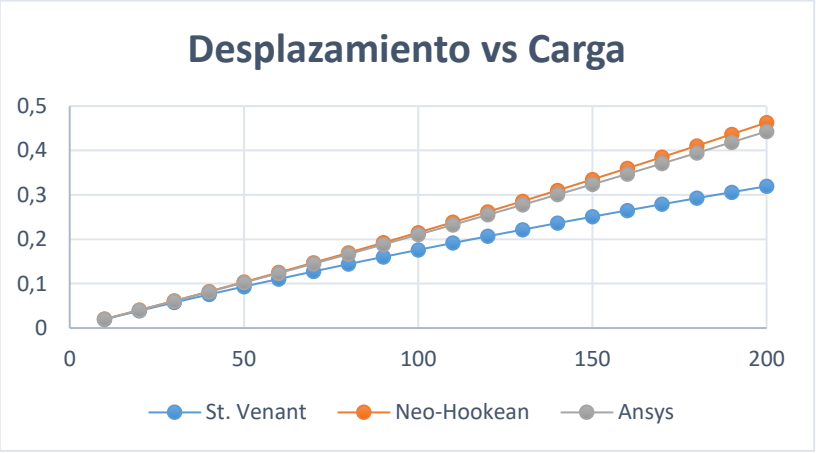


Figura 19.

En la figura 20 la deformación varía entre los 3 modelos, donde el modelo de Neo-Hookean adquiere una mayor magnitud. Las diferencias se deben a que ANSYS utiliza una medida de la deformación logarítmica, y al valor diferente de los desplazamientos.

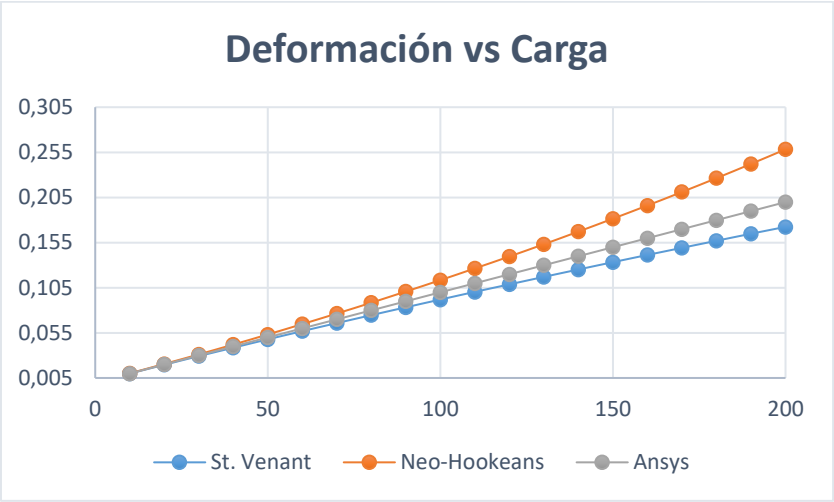


Figura 20.

En la figura 21 se muestra el desplazamiento versus la deformacion y se puede ver que los modelos de Neo-Hookean y Ansys mantienen la misma curva.

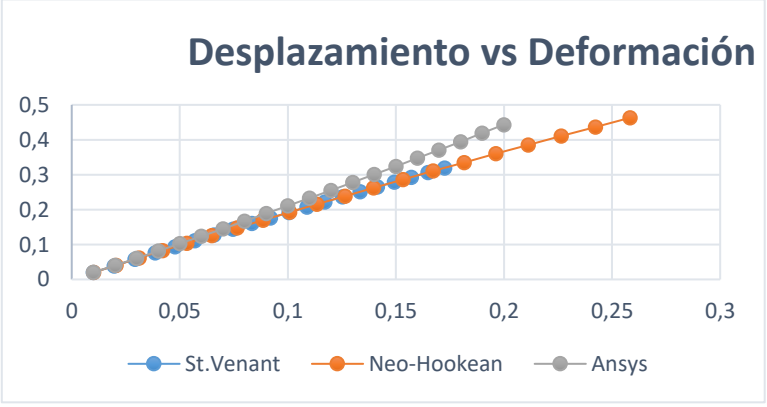


Figura 21.

La figura 22 nos muestra el desplazamiento versus los pasos de carga y podemos apreciar que es muy similar a los resultados de la figura 19.

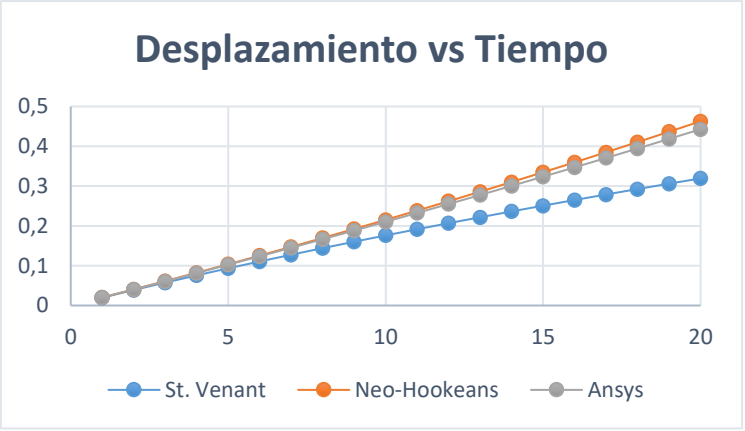


Figura 22.

5.2 Esfera Hueca.

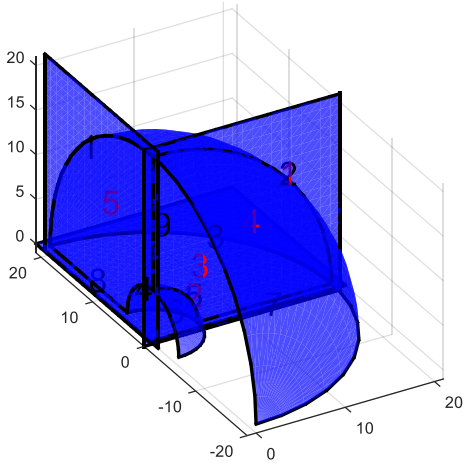


Figura 23.

Las medidas en este caso fueron de radio interno de 5 (m) y radio externo de 20 (m). Se tomó el módulo de Young en 1000 (Pa) y el módulo de Poisson de 0.3. La simulación se lo hizo en 20 pasos de carga.

Se dibujó una octava parte de la esfera y luego se le dio restricciones de simetría en la cara 3 en dirección z, en la cara 4 en dirección x, y en la cara 5 en dirección y. Se asignó una presión interna de 400 (Pa) en la superficie 2 como se muestra en la figura.

Saint Venant

Por problemas de distorsión de los elementos no pudo converger la solución y solo se llegó hasta el paso de carga 8, por lo cual presentaremos los resultados hasta este.

En la figura 24 se observa el desplazamiento en dirección x. El valor máximo es de 0.6025 (m).

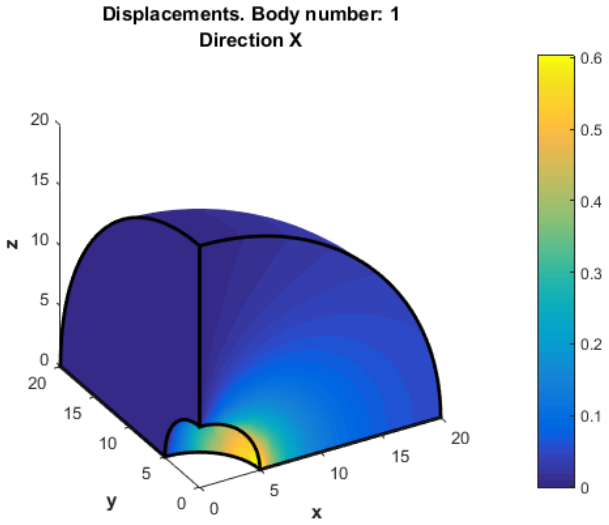


Figura 24.

En la figura 25 se presenta el elemento deformado y sin deformar.

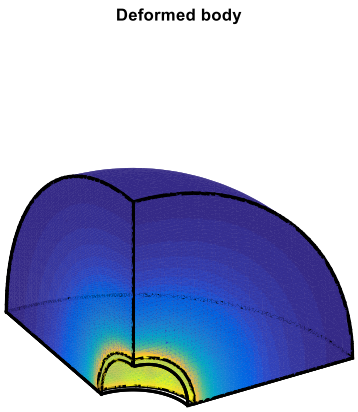


Figura 25.

En la figura 26 la tensión máxima de von Mises es de 200 (Pa).

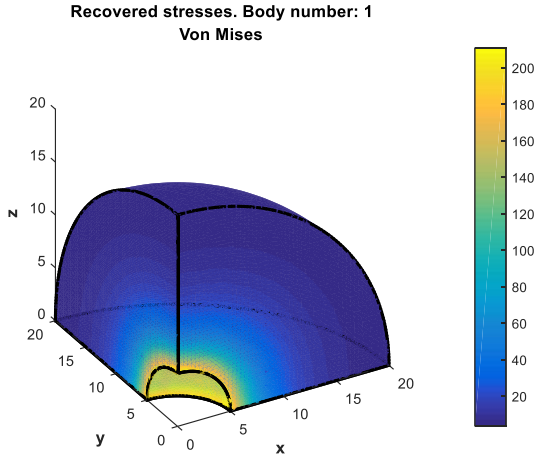


Figura 26.

Neo-Hookean.

En la figura 27 se observa el desplazamiento en dirección x. El valor máximo es de 1.2672 (m).

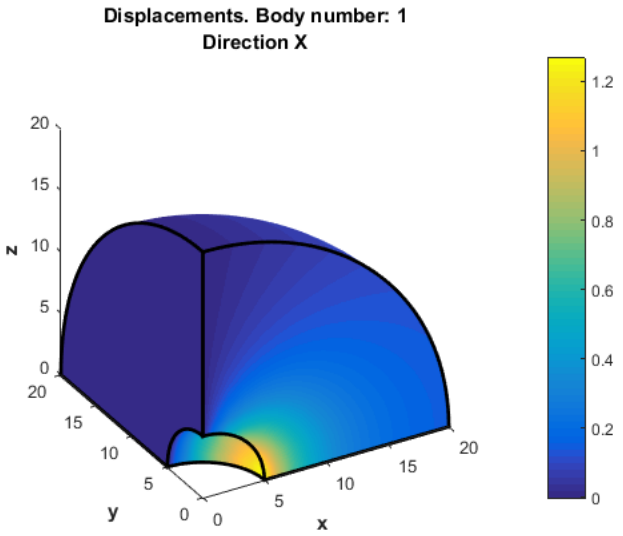


Figura 27.

En la figura 28 se presenta el elemento deformado y sin deformar, y el mapa de colores corresponde al módulo del desplazamiento.

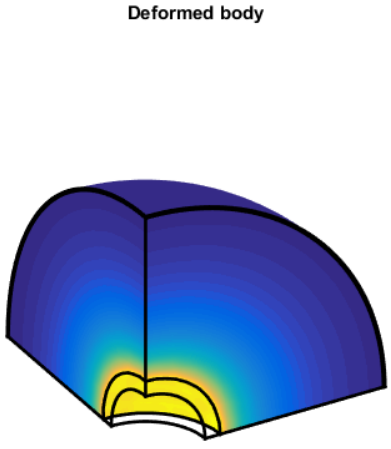


Figura 28.

En la figura 29 la tensión máxima en dirección x es de 300 (Pa).

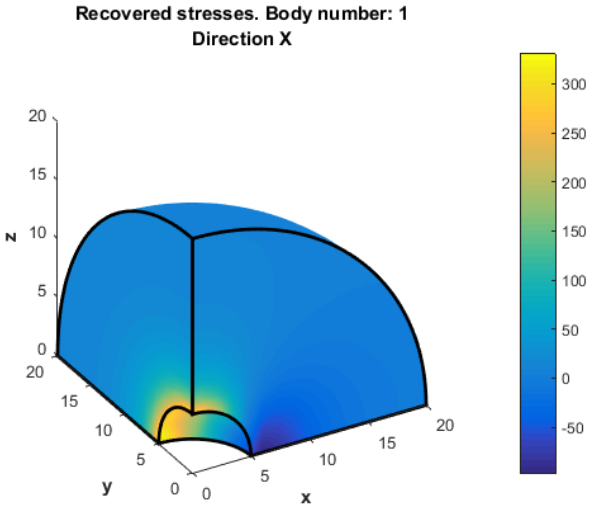


Figura 29.

Ansys.

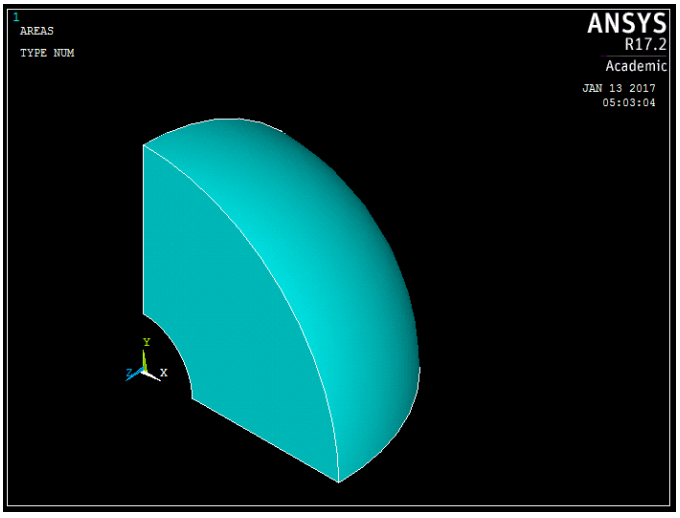


Figura 30.

Las medidas en este caso fueron de radio interno de 5 (m) y radio externo de 20 (m). Se usó un tipo de elemento tipo SOLID185. Para el mallado se usó elementos de tamaño 1. Se tomó el módulo de Young en 1000 (Pa) y el módulo de Poisson de 0.3.

Se obtuvo un total de 11520 elementos y para la simulación se lo hizo en 20 pasos de carga.

Se dibujó una octava parte de la esfera y luego se le dio restricciones de simetría en sus tres caras planas. Se asignó una presión interna de 400 (Pa) como se muestra en la figura 31.

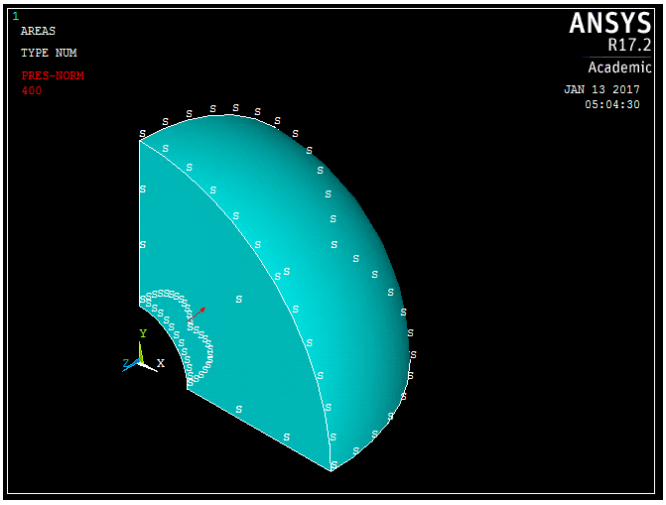


Figura 31.

En la figura 32 se presenta el elemento deformado y sin deformar.

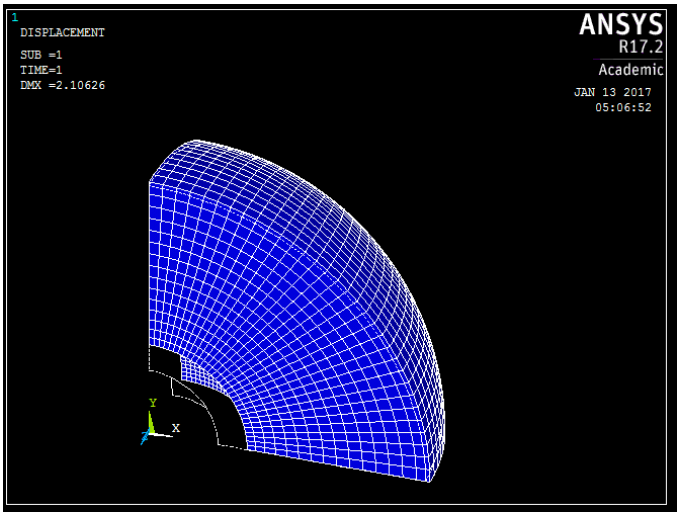


Figura 32.

En la figura 33 se observa el desplazamiento en dirección x. El valor máximo es de 2.1 (m).

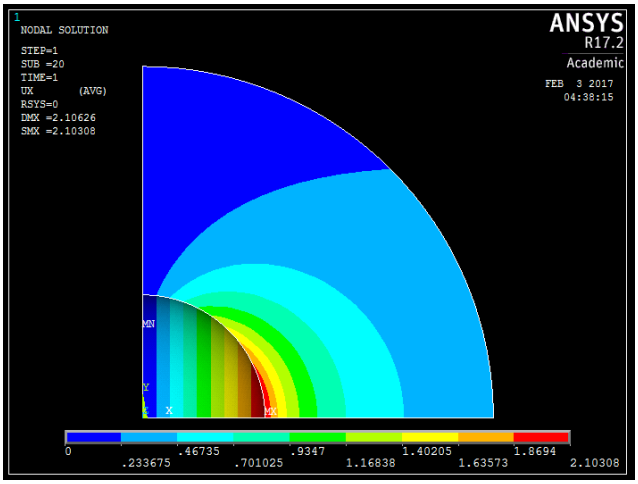


Figura 33.

En la figura 18 la tensión máxima en dirección x es de 373 (Pa).

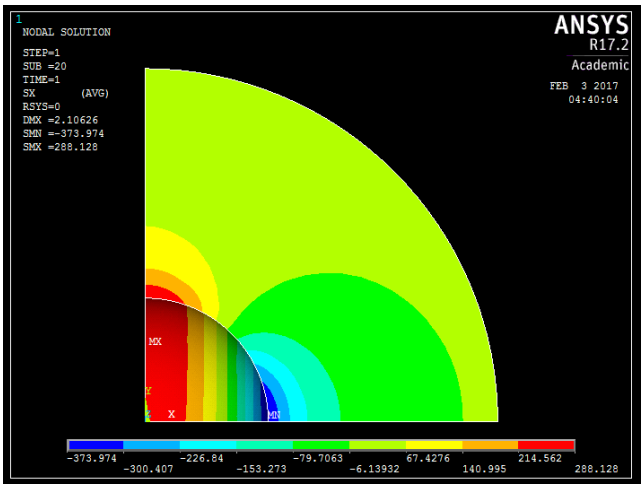


Figura 34.

A continuación, se muestra en forma de grafica una comparación de los resultados obtenidos en cgFEM con los modelos de Saint Venant, Neo-Hookean y Ansys.

En la figura 35 se muestra el desplazamiento del punto de coordenadas (1,0,0) en dirección x versus la presión de tracción aplicada (Este desplazamiento es igual al desplazamiento radial de cualquier punto de la superficie en la solución teórica del problema). Se aprecia un error mayor, ya que con Ansys el desplazamiento máximo en dirección x es de 2,1 (m) y con Neo-Hokean el desplazamiento máximo en x es de 1.2672 (m). Esto se debe a que Ansys calcula los desplazamientos en función del

elemento deformado y en cgFEM se trabaja con el elemento sin deformar. Al coincidir que la cara donde se aplica la presión es la cara donde ocurre el mayor desplazamiento el error es mayor. Ademas se puede ver que el modelo de Saint-Venant no converge para este nivel de deformaciones ya que alcanzan un desplazamiento mayor al 10% y este modelo no es hiperélastico.

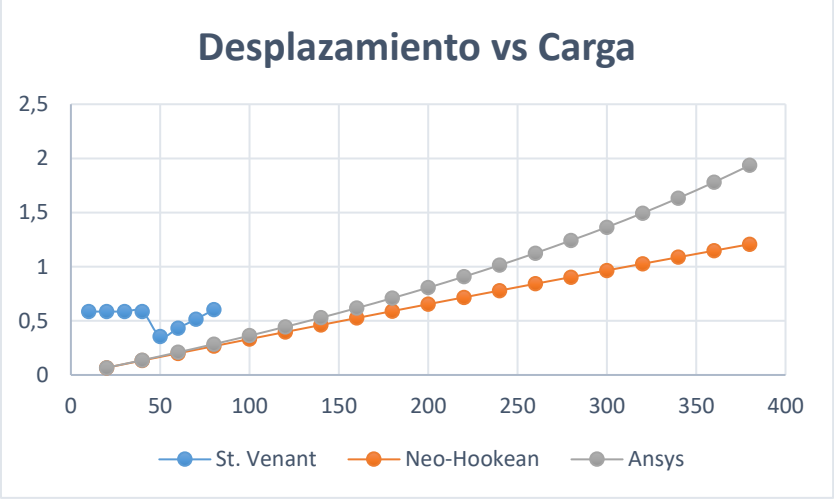


Figura 35.

En la figura 36 la deformación varía entre los 3 modelos, donde el modelo de Ansys adquiere una mayor magnitud. Se observa la no convergencia del modelo de Saint-Venant.

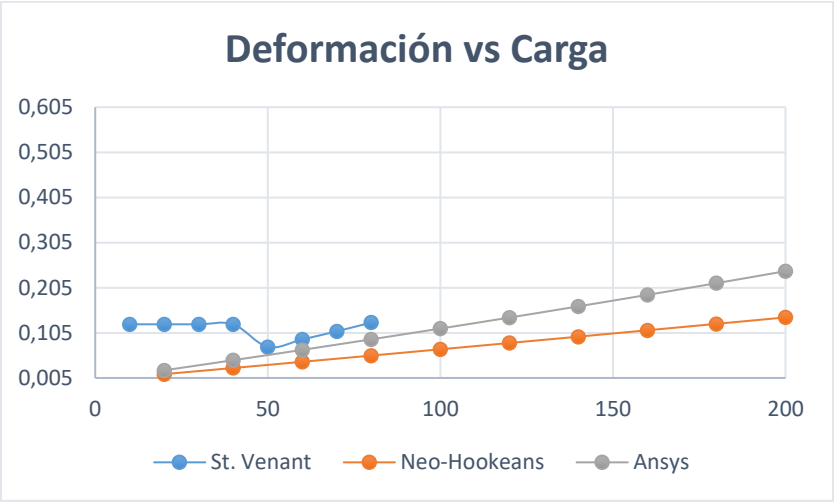


Figura 36.

En la figura 37 se muestra el desplazamiento versus la deformacion. Los resultados varian mucho con el modelo de Ansys por la difencia del analisis entre el elemento deformado y sin deformar. Se observa la no convergencia del modelo de Saint-Venant.

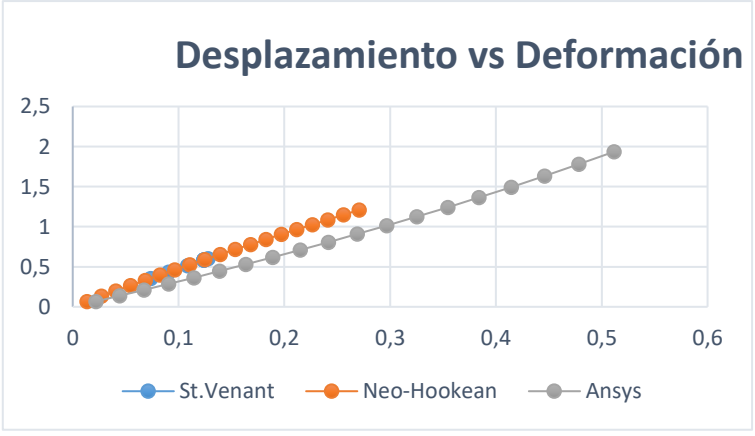


Figura 37.

La figura 38 nos muestra el desplazamiento versus los pasos de carga y podemos apreciar que es muy similar a los resultados de la figura 35. Se observa la no convergencia del modelo de Saint-Venant.

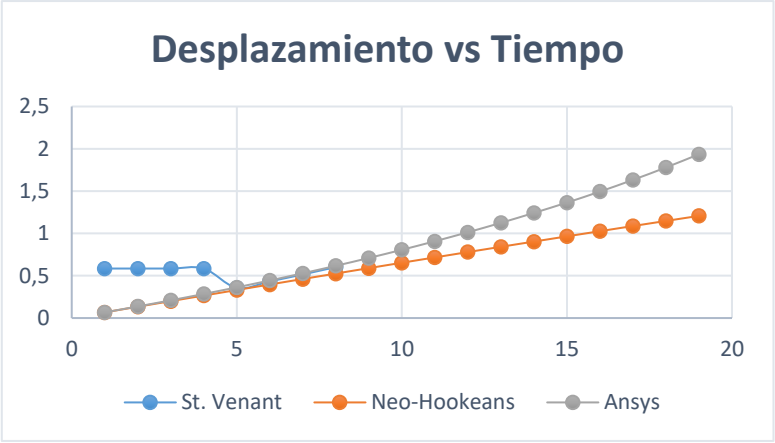


Figura 38.

5.3 Placa con Agujero.

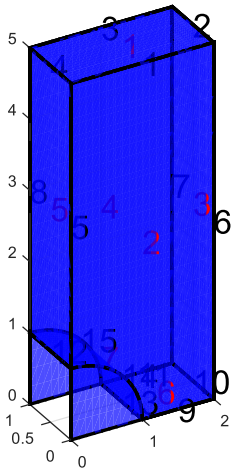


Figura 39.

Las medidas son como se muestran en la figura 39 medidas en metro. Se tomó el módulo de Young en 1000 (Pa) y el módulo de Poisson de 0.3 y para la simulación se lo hizo en 20 pasos de carga.

Se dibujó una octava parte de la de la placa y luego se le dio restricciones de simetría en la cara 2 en dirección y, en la cara 5 en dirección x, y en la cara 6 en dirección z. Se asignó una presión de tracción de 200 (Pa) en la cara 1 como de acuerdo a la figura 40.

Saint Venant.

A continuación, se muestran las figuras del elemento en el último paso de carga.

En la figura 40 se observa el desplazamiento en dirección z. El valor máximo es de 0.9826 (m).

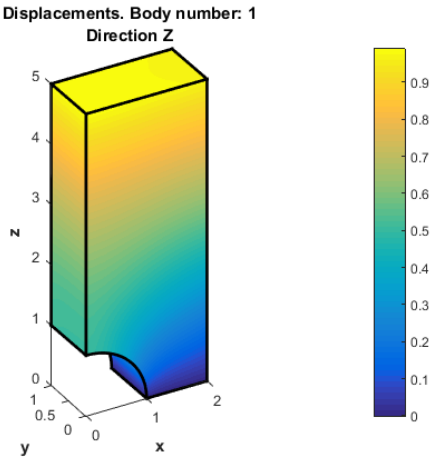


Figura 40.

En la figura 41 se presenta el elemento deformado y sin deformar.

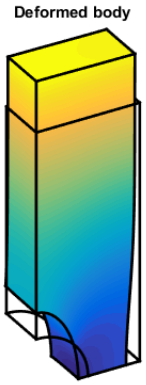


Figura 41.

En la figura 42 la tensión máxima en dirección z es de 1000 (Pa).

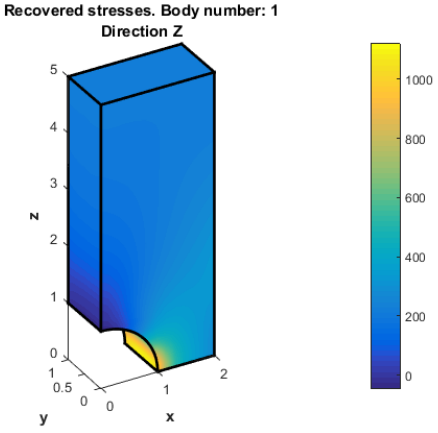


Figura 42.

Neo-Hookean.

En la figura 43 se observa el desplazamiento en dirección z. El valor máximo es de 1.4767 (m).

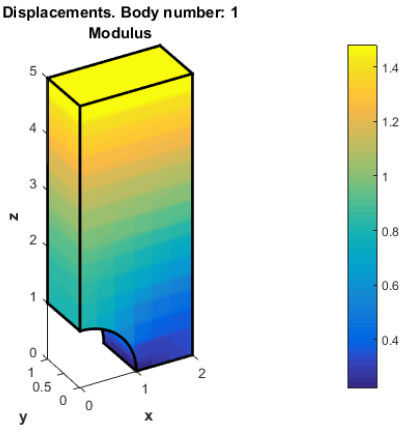


Figura 43.

En la figura 44 se presenta el elemento deformado y sin deformar.

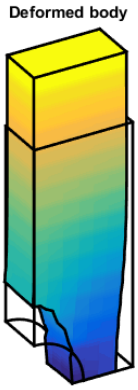


Figura 44.

En la figura 45 la tensión máxima en dirección z es de 1800 (Pa). Hay un mayor esfuerzo está en el punto [1 0 0] que se genera por una mayor reducción del área transversal.

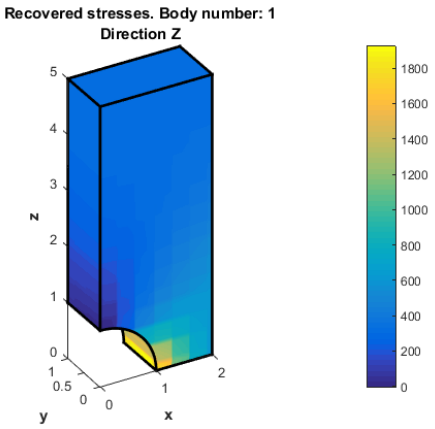


Figura 45.

Ansys.

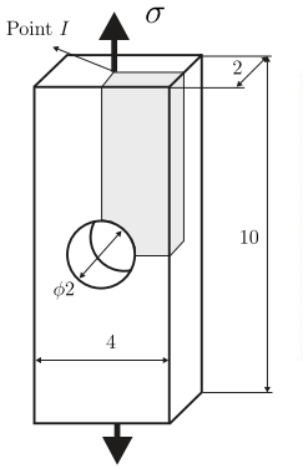


Figura 46.

Las medidas son como se muestran en la figura 46. Se usó un tipo de elemento tipo SOLID185. Para el mallado se usó elementos de tamaño 0.2. Se tomó el módulo de Young en 1000 (Pa) y el módulo de Poisson de 0.3.

Se obtuvo un total de 8665 elementos y para la simulación se lo hizo en 20 pasos de carga.

Se dibujó una octava parte de la de la placa y luego se le dio restricciones de simetría en sus tres caras planas. Se asignó una presión de tracción de 200 (Pa) en la cara como se muestra en la figura 47.

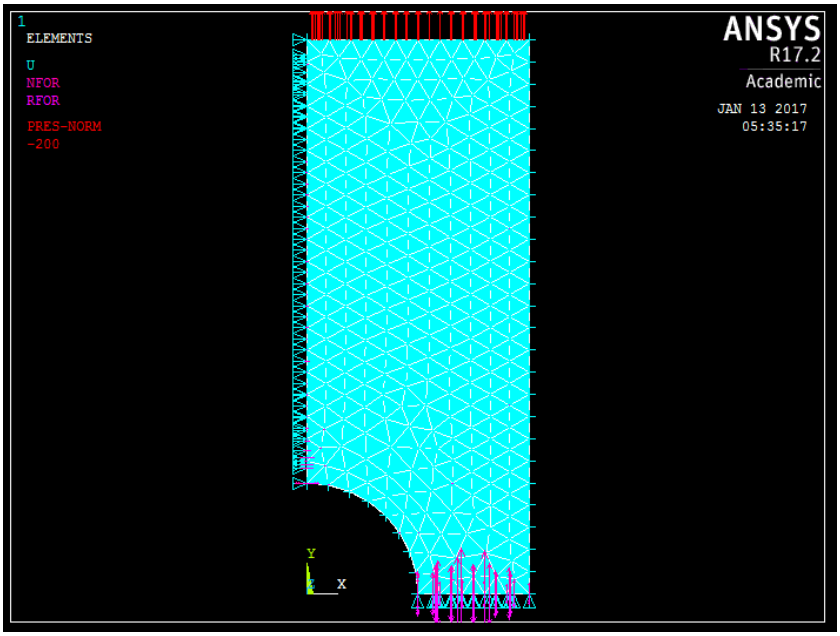


Figura 47.

En la figura 48 se presenta el elemento deformado y sin deformar.

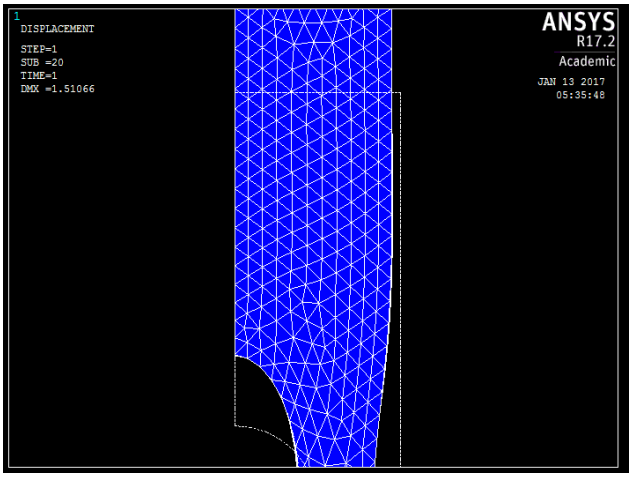


Figura 48.

En la figura 49 se observa el desplazamiento en dirección y. El valor máximo es de 1.5 (m).

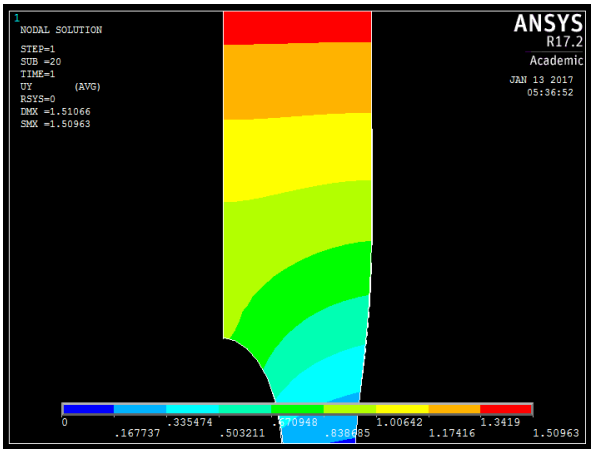


Figura 49.

En la figura 50 la tensión máxima en dirección y es de 706 (Pa).

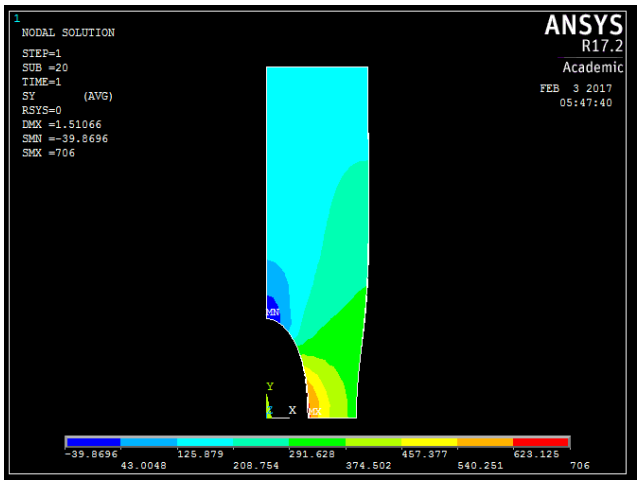


Figura 50.

A continuación, se muestra en forma de grafica una comparación de los resultados obtenidos en cgFEM con los modelos de Saint Venant, Neo-Hookean y Ansys.

En la figura 51 se muestra el desplazamiento versus la presión de tracción aplicada. Se puede ver que el modelo de Neo-Hookean y Ansys son los que más se parecen teniendo una diferencia menor del 10%.

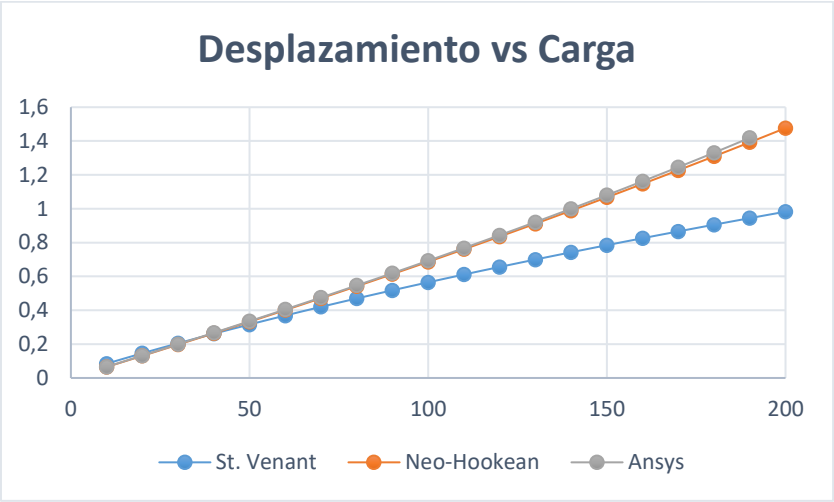


Figura 51.

En la figura 52 la deformación varía entre los 3 modelos, donde el modelo de Neo-Hookean adquiere una mayor magnitud.

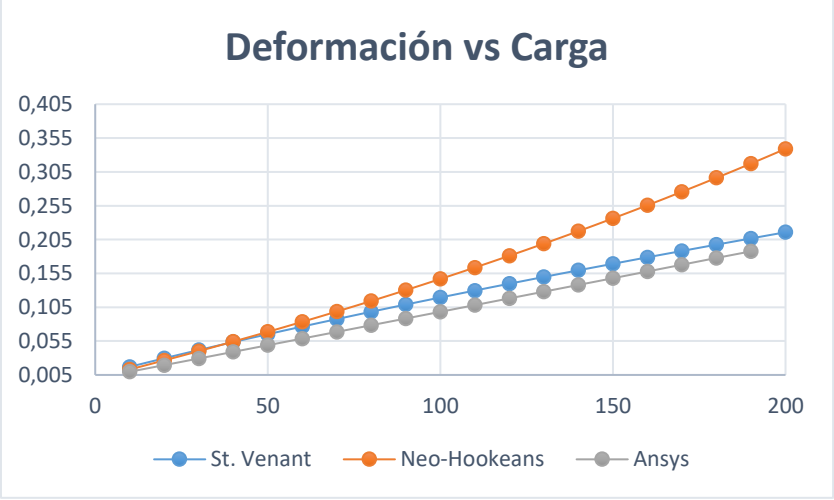


Figura 52.

En la figura 53 se muestra el desplazamiento versus la deformacion y se puede ver que los modelos de Neo-Hookean y Ansys mantienen la misma curva.

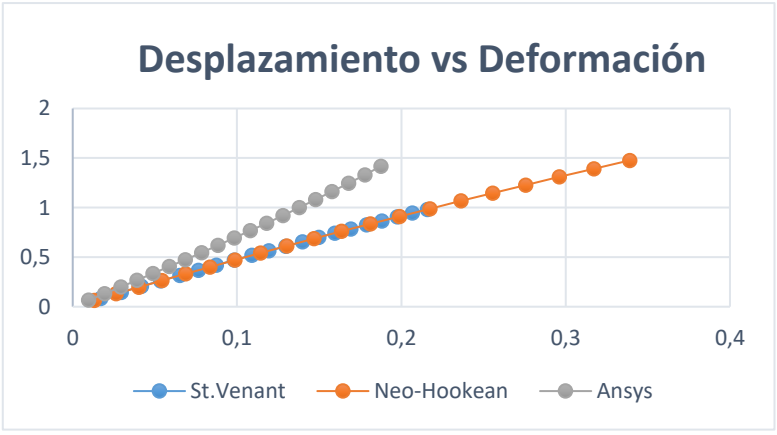


Figura 53.

La figura 54 nos muestra el desplazamiento versus los pasos de carga y podemos apreciar que es muy similar a los resultados de la figura 51.

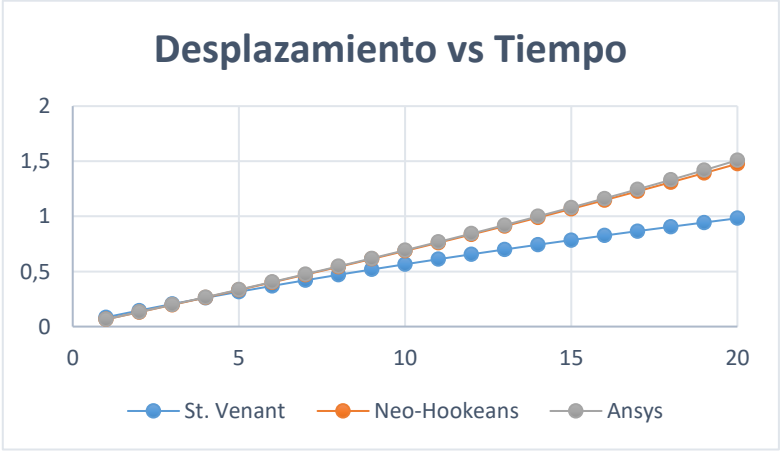


Figura 54.

5.4 Análisis de Resultados.

Resumen de magnitudes máximas.

	cgFEM Saint Venant		
	CUBO	ESFERA HUECA	PLACA AGUJERO
Desplazamiento	0,3197	0,61	0,99
Deformación	0,1724	0,1278	0,2166
Tensión	223	200	1000
Carga	200	400	200
	cgFEM Neo-Hookean		
Desplazamiento	0,463	1,2672	1,4767
Deformación	0,2583	0,2855	0,339
Tensión	320	300	1800
Carga	200	400	200
	ANSYS		
Desplazamiento	0,44	0,61	1,5
Deformación	0,2	0,189	0,1989
Tensión	200	373	706
Carga	200	400	200

Al realizar la comparativa entre cgFEM y ANSYS podemos observar que hay diferencias significativas entre los dos programas. Si bien el modelo de Neo-Hookean se ajusta más a la realidad, con Saint Venant hubo mayor margen de error.

6. Conclusiones.

- La ventaja de trabajar con la deformación de Green-Lagrange se tienen buenos resultados al trabajar en grandes desplazamientos, es decir en grandes deformaciones y rotaciones. Es por eso que se empleó este método en el código de cgFEM. La ventaja de trabajar con la deformación logarítmica es que se tiene mejores resultados cuando se trabaja solamente con grandes desplazamientos es por eso que en estos ejemplos donde no hay grandes rotaciones ANSYS obtiene mejores resultados.
- El problema de la distorsión de los elementos es otro problema que se presentó, al momento de la interpolación, como la malla no coincide con el elemento se dificulta la convergencia, como en el ejemplo de la esfera hueca, en donde no pudimos alcanzar la convergencia en este.
- De acuerdo a las gráficas de desplazamiento vs presión, podemos ver que la no linealidad existe y se ajusta bastante en comparación con las de ANSYS.
- Se puede concluir uno de los principales problemas que se presenta al momento en el programa de cgFEM es la interpolación ya que la malla no coincide con el objeto de análisis.

- En los gráficos de desplazamiento vs presión y desplazamiento vs pasos de carga se puede observar una gran similitud en todos sus tipos de ejemplos, por lo que se puede concluir que la variación de la carga aplicada (gradiente de presiones) es casi lineal, por lo que la no linealidad se marca en los desplazamientos.
- El modelo de Saint Venant es más real cuando se trabaja en deformaciones del 5% al 10%, es por eso que en el ejemplo de la esfera hueca presento errores, y el modelo de Neo-Hookean es el que se ajustó más a lo que nos mostraba ANSYS.
- Neo-Hookean al ser un modelo no lineal presento mejores resultados que Saint Venant.
- En la formulación de grandes desplazamientos y en el desarrollo del código trabaja con la deformación de Green-Lagrange, mientras que Ansys trabaja con la deformación logarítmica. Es por eso que se ve en las deformaciones una mayor variación del resultado final. [7]

BIBLIOGRAFIA.

[1] «Non Linear Finite elements For Continua and Structures,» de *Ted Belytshko, Wing Kam Liu, Brian Moran, Khalil I.Elkhodary*.

[2] DIMM, «Tema 3. Problemas de Grandes desplazamientos.,» de *Departamento de Ingeniería Mecánica y de Materiales.*.

[3] «Computational Modeling of Objects Presented in Images: Fundamentals, Methods, and Applications,» de *Yongjie Jessica Zhang, Joao Manuel R.S. Tavares*, Springer, Septiembre 2014.

[4] «Optimization of a finite element code implemented in MATLABr. On the use of GPUs for High Performance Computing.,» de *José Manuel Navarro Jiménez*, 2014.

[5] «Cartesian grid FEM (cgFEM): High performance h-adaptive FE analysis with efficient error control. Application to structural shape optimization,» de *Enrique Nadal Soriano*, Universidad Politécnica de Valencia.

[6] DIMM, «Tema 3. Interpolación de funciones de continuidad Co,» de *Departamento de Ingeniería Mecánica y de Materiales.*.

[7] ANSYS, «ANSYS Mechanical APDL Theory Reference».

ANEXO

```
function sd2Dsolid
% Programa 2D de cuadriláteros con grandes
desplazamientos y
% comportamiento elástico lineal
% Datos del material
E = 1000;
nu = 0.3;
modK = E/3/(1-2*nu);
lambda = E*nu/(1+nu)/(1-2*nu);
G = E/2/(1+nu);
%-- Grados de libertad por nodo
Ngdl = 2;
%-- Lee la malla de elementos finitos (TOP, XYZ)
load TOP
load XYZ
%-- Número de elementos: filas del fichero TOP
Ne = size(TOP,1);
%-- Número de nodos: filas del fichero XYZ
Nn = size(XYZ,1);
%-- Leer condiciones de Dirichlet
%-- Genera el vector gdlA que contiene los grados
de libertad activos
load DIR
gdlA(1:Ngdl*Nn) = 1;
for i=1:size(DIR,1)
    node = DIR(i,1);
    dir = DIR(i,2);
    gdlA((node-1)*Ngdl + dir ) = 0;
end
gdlA=gdlA>0;
%-- Dibuja posición sin deformar
dibuja_cuad(1,TOP,XYZ,1)
axis equal
%-- Definir vector de fuerzas externas en cada
grado de libertad
% Opción 1: Leer un fichero
% load FEXT
% Fex(2*Nn,1) = 0;
% for i=1:size(FEXT,1)
```

```
%     Fex(FEXT(i,1)*2-1) = FEXT(i,2);
%     Fex(FEXT(i,1)*2) = FEXT(i,3);
% end
% Opción 2: Definirlo manualmente
Fex(Ngdl*Nn,1) = 0;
Nodo = 2;
Gdl=1;
Fex(Ngdl*(Nodo-1)+Gdl) = -10;
Nodo = 3;
Gdl=1;
Fex(Ngdl*(Nodo-1)+Gdl) = 20;
%-- Resolución de Newton-Raphson
%-- Cálculo del vector de fuerzas internas debido a
las deformaciones elásticas
%-- Ya sale ensamblado
U = XYZ*0;
XYZ_def = XYZ;
Fint =
fint_sd4Nsolid(TOP,XYZ,U,E,nu,XYZ_def,lambda,G);
%-- Cálculo del residuo inicial
R = Fint - Fex;
disp('Norma del residuo:')
nr_ini = norm(R);
nr = nr_ini;
nr
Iter = 1;
Q = reshape(XYZ',[],1);
XYZ_def = XYZ;
Uv = reshape(U,[],1);
while ((nr/nr_ini > 1e-8) && (Iter<10))
    %-- Matriz tangente ya ensamblada
    K = K_sd4Nsolid(TOP,XYZ,E,nu,XYZ_def,lambda,G);
    %-- Solución sistema de ecuaciones
    DQ = - K(gdlA,gdlA) \ R(gdlA);
    %-- Desplazamientos
    Uv(gdlA) = DQ;
    U = reshape(Uv,Ngdl,Nn)';
    %-- Actualizar posición
    Q(gdlA) = Q(gdlA) + DQ;
    XYZ_def = reshape(Q,Ngdl,Nn)';
    %-- Cálculo del residuo con la solución
modificada
```

```
    Fint =
fint_sd4Nsolid(TOP,XYZ,U,E,nu,XYZ_def,lambda,G);
    R = Fint - Fex;
    Iter
    disp('     - Norma del residuo:')
    nr = norm(R(gdlA));
    nr/nr_ini
    Iter = Iter + 1;
    dibuja_cuad(1,TOP,XYZ_def,1)
    pause
end
dibuja_cuad(1,TOP,XYZ_def,1)
end
```

```
function N = ShapeFunc4N_loc( xi, eta)
N(1) = (1-xi)*(1-eta) / 4;
N(2) = (1+xi)*(1-eta) / 4;
N(3) = (1+xi)*(1+eta) / 4;
N(4) = (1-xi)*(1+eta) / 4;
end
```

```
function dN = derN(xi,eta)

dN1_1 = -(1-eta)/4;
dN2_1 = (1-eta)/4;
dN3_1 = (1+eta)/4;
dN4_1 = -(1+eta)/4;
dN1_2 = -(1-xi)/4;
dN2_2 = -(1+xi)/4;
dN3_2 = (1+xi)/4;
dN4_2 = (1-xi)/4;
dN = [dN1_1 dN2_1 dN3_1 dN4_1 ; dN1_2 dN2_2 dN3_2
dN4_2];
end
```

```
function [J dN_XY] = jaco(ie,TOP,XYZ,xi,eta)
%-- Derivadas de las funciones de forma en
coordenadas locales
```

```
dN = derN(xi,eta);
X = XYZ(TOP(ie,:),1);
Y = XYZ(TOP(ie,:),2);
J(1,1) = dN(1,:)*X;
J(1,2) = dN(2,:)*X;
J(2,1) = dN(1,:)*Y;
J(2,2) = dN(2,:)*Y;
%-- Derivadas de las funciones de forma respecto de
las coordenadas XYZ
iJ = inv(J);
for k=1:4
    dN_XY(:,k) = iJ*dN(:,k);
end
end
```

```
function dibuja_cuad(nf,TOP,XYZ,op)
%   dibuja_cuad(nf,TOP,XYZ)
%   Datos de entrada:
%     - nf: Número de la figura
%     - TOP: elementos de la malla
%     - XYZ: coordenadas de los nodos
%     - op: 1 - Borra la figura
%           2 - No borra la figura
figure(nf);
if op==1
    clf
end
grid on
hold on
Ne = size(TOP,1);
% Dibuja cada elemento como una línea
for ie=1:Ne
    % Bucle para todos los elementos
    % Coordenadas x de los dos nodos
    xd = [XYZ(TOP(ie,1),1) XYZ(TOP(ie,2),1)
XYZ(TOP(ie,3),1) XYZ(TOP(ie,4),1)
XYZ(TOP(ie,1),1)];
    % Coordenadas y de los dos nodos
    yd = [XYZ(TOP(ie,1),2) XYZ(TOP(ie,2),2)
XYZ(TOP(ie,3),2) XYZ(TOP(ie,4),2)
XYZ(TOP(ie,1),2)];
```

```
    % Dibuja linea
    if op==1
        plot(xd,yd,'b--o','LineWidth',1);
    else
        plot(xd,yd,'b-o','LineWidth',3);
    end
end
% Esto sirve para ajustar los ejes de la figura
ax = gca;
dx = max(XYZ(:,1)) - min(XYZ(:,1));
xmin = min(XYZ(:,1)) - dx*0.1;
xmax = max(XYZ(:,1)) + dx*0.1;
dy = max(max(XYZ(:,2)) - min(XYZ(:,2)), 1);
ymin = min(XYZ(:,2)) - dy*0.1;
ymax = max(XYZ(:,2)) + dy*0.1;
set(ax,'XLim', [xmin xmax], 'YLim', [ymin ymax]);
end
```

SAINT VENANT

```
function K =
K_sd4Nsolid(TOP,XYZ,E,nu,XYZ_def,lambda,G)
% Calcula la matriz tangente de cada elemento y la
ensambla
% para formar la matriz K global
% Define los puntos de integración
ng = 3;
wg = [5/9 8/9 5/9];
xg = [-sqrt(3/5) 0 sqrt(3/5)];
% Matriz del material
D = E/(1-2*nu)/(1+nu)*[1-nu nu 0; nu 1-nu 0; 0 0
(1-2*nu)/2];
K = [];
gdlf = [];
gdlc = [];
for ie=1:size(TOP,1)
    % Bucle para cada elemento
    % Nodos del elemento
    ni = TOP(ie,1);  nj = TOP(ie,2);
    % Coordenadas de los nodos del elemento
    K_el = zeros(8,8);
```

```
    % Matriz tangente del elemento
    for ig=1:ng
        for jg=1:ng
            xi=xg(ig);
            eta=xg(jg);
            % Derivadas de las funciones de forma
en coordenadas locales
            dN = derN(xi,eta);
            %Leer funcion de jacobiana y derivada
de funciones de forma en
            %coordenadas globales
            [Jacobiano,dNXY] =
jaco(ie,TOP,XYZ,xi,eta);
            % Matriz jacobiana de la transformación
            J = Jacobiano;
            % Derivadas de las funciones de forma
en coordenadas globales
            dN_XY = dNXY;
                        %-- Coordendasd deformadas
            x = XYZ_def(TOP(ie,:),1);
            y = XYZ_def(TOP(ie,:),2);
            %-- Gradiente de deformaciones
            F(1,1) = dN_XY(1,:)*x;
            F(1,2) = dN_XY(2,:)*x;
            F(2,1) = dN_XY(1,:)*y;
            F(2,2) = dN_XY(2,:)*y;
            %-- Deformación de Green-Lagrange
            I = eye(2);
            Eg = 1/2*(F'*F - I);
            %-- Tensión de Piola-Kirchhoff PK2
            S = lambda*sum(diag(Eg))*I + 2*G*Eg;
            %-- Vector de fuerzas equivalentes en
nodos:
            %-- Int ( S * deltaEg )
            B_L = [F(1,1)*dN_XY(1,1)
F(2,1)*dN_XY(1,1)  F(1,1)*dN_XY(1,2)
F(2,1)*dN_XY(1,2)  F(1,1)*dN_XY(1,3)
F(2,1)*dN_XY(1,3)  F(1,1)*dN_XY(1,4)
F(2,1)*dN_XY(1,4)  ; ...
                   F(1,2)*dN_XY(2,1)
F(2,2)*dN_XY(2,1)  F(1,2)*dN_XY(2,2)
F(2,2)*dN_XY(2,2)  F(1,2)*dN_XY(2,3)
```

```
F(2,2)*dN_XY(2,3)  F(1,2)*dN_XY(2,4)
F(2,2)*dN_XY(2,4)  ; ...
                   F(1,1)*dN_XY(2,1) +
F(1,2)*dN_XY(1,1)   F(2,1)*dN_XY(2,1) +
F(2,2)*dN_XY(1,1)   F(1,1)*dN_XY(2,2) +
F(1,2)*dN_XY(1,2)   F(2,1)*dN_XY(2,2) +
F(2,2)*dN_XY(1,2) ...
                   F(1,1)*dN_XY(2,3) +
F(1,2)*dN_XY(1,3)   F(2,1)*dN_XY(2,3) +
F(2,2)*dN_XY(1,3)  F(1,1)*dN_XY(2,4) +
F(1,2)*dN_XY(1,4)   F(2,1)*dN_XY(2,4) +
F(2,2)*dN_XY(1,4)];
            %-- Matriz G
            for im=1:4;
                for km=1:4;
                    Gik =
dN_XY(:,im)'*S*dN_XY(:,km);
                    MG(2*im-1:2*im,2*km-
1:2*km)=eye(2)*Gik;
                end
            end
            % Fuerza interna en un elemento
            K_el = K_el +
wg(ig)*wg(jg)*(MG+B_L'*D*B_L)*det(J) ;
        end
    end
    % Información para el ensamblado
    % Grados de libertad del elemento
    gdle = [2*TOP(ie,1)-1 2*TOP(ie,1) 2*TOP(ie,2)-1
2*TOP(ie,2) 2*TOP(ie,3)-1 2*TOP(ie,3) 2*TOP(ie,4)-1
2*TOP(ie,4)]';
    gdlfe = reshape(gdle*ones(1,8),[],1);
    gdlce = reshape(ones(8,1)*gdle',[],1);
    K_el = reshape(K_el,[],1);
    K = [K; K_el];
    gdlf = [gdlf; gdlfe];
    gdlc = [gdlc; gdlce];
end
% Ensamblado de la matriz
K = sparse(gdlf,gdlc,K);
```

```
function Fint =
fint_sd4Nsolid(TOP,XYZ,U,E,nu,XYZ_def,lambda,G)
% Calcula el vector de fuerzas internas de cada
elementos y lo ensambla
% para formar el vector Fint global
% Define los puntos de integración en cada
direccion
ng = 3;
wg = [5/9 8/9 5/9];
xg = [-sqrt(3/5) 0 sqrt(3/5)];
% Matriz del material
D = E*[(1/(1-nu^2)) (nu/(1-nu^2)) 0 ; (nu/(1-nu^2))
(1/(1-nu^2)) 0; 0 0 (1/(2*(1-nu)))];
fel = [];
gdl = [];
for ie=1:size(TOP,1)
    % Nodos del elemento
    % Coordenadas de los nodos del elemento
    %definir U del elemento
 Uel=[U(TOP(ie,1),1) U(TOP(ie,1),2) U(TOP(ie,2),1)
U(TOP(ie,2),2) U(TOP(ie,3),1) U(TOP(ie,3),2)
U(TOP(ie,4),1) U(TOP(ie,4),2)]';
    % VEctor de fuerzas debido a la deformación
elástica
    fel_el = zeros(8,1);
    for ig=1:ng
        for jg=1:ng
            % Derivadas de las funciones de forma
en coordenadas locales
            xi=xg(ig);
            eta=xg(jg);
            dN = derN(xi,eta);
            %Leer funcion de jacobiana y derivada
de funciones de forma en
            %coordenadas globales
            [Jacobiano,dNXY] =
jaco(ie,TOP,XYZ,xi,eta);
            % Matriz jacobiana de la transformación
            J = Jacobiano;
            % Derivadas de las funciones de forma
en coordenadas globales
            dN_XY = dNXY;
            %-- Coordendasd deformadas
```

```
            x = XYZ_def(TOP(ie,:),1);
            y = XYZ_def(TOP(ie,:),2);
            %-- Gradiente de deformaciones
            F(1,1) = dN_XY(1,:)*x;
            F(1,2) = dN_XY(2,:)*x;
            F(2,1) = dN_XY(1,:)*y;
            F(2,2) = dN_XY(2,:)*y;
            %-- Deformación de Green-Lagrange
            I = eye(2);
            Eg = 1/2*(F'*F - I);
            %-- Tensión de Piola-Kirchhoff PK2
            S = lambda*sum(diag(Eg))*I + 2*G*Eg;
            Sv = [S(1,1);S(2,2);S(1,2)];
            %-- Vector de fuerzas equivalentes en
nodos:
            %-- Int ( S * deltaEg )
            B_L = [F(1,1)*dN_XY(1,1)
F(2,1)*dN_XY(1,1)  F(1,1)*dN_XY(1,2)
F(2,1)*dN_XY(1,2)  F(1,1)*dN_XY(1,3)
F(2,1)*dN_XY(1,3)  F(1,1)*dN_XY(1,4)
F(2,1)*dN_XY(1,4)  ; ...
                   F(1,2)*dN_XY(2,1)
F(2,2)*dN_XY(2,1)  F(1,2)*dN_XY(2,2)
F(2,2)*dN_XY(2,2)  F(1,2)*dN_XY(2,3)
F(2,2)*dN_XY(2,3)  F(1,2)*dN_XY(2,4)
F(2,2)*dN_XY(2,4)  ; ...
                   F(1,1)*dN_XY(2,1) +
F(1,2)*dN_XY(1,1)   F(2,1)*dN_XY(2,1) +
F(2,2)*dN_XY(1,1)   F(1,1)*dN_XY(2,2) +
F(1,2)*dN_XY(1,2)   F(2,1)*dN_XY(2,2) +
F(2,2)*dN_XY(1,2) ...
                   F(1,1)*dN_XY(2,3) +
F(1,2)*dN_XY(1,3)   F(2,1)*dN_XY(2,3) +
F(2,2)*dN_XY(1,3)  F(1,1)*dN_XY(2,4) +
F(1,2)*dN_XY(1,4)   F(2,1)*dN_XY(2,4) +
F(2,2)*dN_XY(1,4)];
                % Fuerza interna en un elemento
            fel_el = fel_el +
wg(ig)*wg(jg)*det(J)*B_L'*Sv;
        end
    end
    % Información para ensamblado
    % Grados de libertad del elemento
```

```
    gdle = [2*TOP(ie,1)-1 2*TOP(ie,1) 2*TOP(ie,2)-1
2*TOP(ie,2) 2*TOP(ie,3)-1 2*TOP(ie,3) 2*TOP(ie,4)-1
2*TOP(ie,4)]';
    fel = [fel; fel_el];
    gdl = [gdl; gdle];
end
% Ensamblado del vector:
Fint = accumarray(gdl,fel);
```

NEO-HOOKEANS

```
function K = K_ld4Nsolid(TOP,XYZ,XYZdef,E,nu)
% Calcula la matriz tangente de cada elemento y la
ensambla
% para formar la matriz K global
% Define los puntos de integración
ng = 3;
wg = [5/9 8/9 5/9];
xg = [-sqrt(3/5) 0 sqrt(3/5)];
% Matriz del material
lambda = E*nu/(1+nu)/(1-2*nu);
Gm = E/2/(1+nu);
G = zeros(8,8);
dN_x = zeros(2,4);
B_L = zeros(3,8);
K = [];
gdlf = [];
gdlc = [];
for ie=1:size(TOP,1)
    % Bucle para cada elemento
    % Nodos del elemento
    node = TOP(ie,:);
    % Coordenadas de los nodos del elemento
    Xe = XYZ(node,:);
    % Posición deformada de los nodos del elemento
    xe = XYZdef(node,:);
    K_el = zeros(8,8);
    % Matriz tangente del elemento
    for ig=1:ng
        for jg=1:ng
```

```
            % Derivadas de las funciones de forma
en coordenadas locales
            dN = derShapeFunc4N_loc(xg(ig),xg(jg));
            % Matriz jacobiana de la transformación
            J(1,1) = dN(1,:) * xe(:,1);
            J(1,2) = dN(1,:) * xe(:,2);
            J(2,1) = dN(2,:) * xe(:,1);
            J(2,2) = dN(2,:) * xe(:,2);
            F = eye(2);
            for k=1:4
                dN_x(:,k) = J\dN(:,k);
                F = F - (xe(k,:)-
Xe(k,:))'*dN_x(:,k)';
            end
            F = inv(F);
            detJ = det(F);
            detj = det(J);
            for k=1:4
                B_L(1,2*k-1) = dN_x(1,k);
                B_L(1,2*k) = 0;
                B_L(2,2*k-1) = 0;
                B_L(2,2*k) = dN_x(2,k);
                B_L(3,2*k-1) = dN_x(2,k);
                B_L(3,2*k) = dN_x(1,k);
            end
            % Tensor de tensiones de Cauchy
            s = lambda/(2*detJ)*(detJ^2-1)*eye(2) +
Gm/detJ*(F*F'-eye(2));
            sv = [s(1,1); s(2,2); s(1,2)];
            for k=1:4
                for i=1:4
                    Gik = dN_x(:,i)'*s*dN_x(:,k);
                    G(2*i-1:2*i,2*k-1:2*k) =
Gik*eye(2);
                end
            end
            D = 1/detJ*[2*Gm+lambda, lambda*detJ^2,
0; lambda*detJ^2, 2*Gm+lambda, 0; 0, 0, Gm-
lambda*(detJ^2-1)/2];
            % Fuerza interna en un elemento
            K_el = K_el + wg(ig)*wg(jg)*(B_L'*D*B_L
+ G)*detj;
        end
```

```
    end
    % Información para el ensamblado
    aux = [2*node'-1  2*node'];
    gdle = reshape(aux',[],1);
    gdlfe = reshape(gdle*ones(1,8),[],1);
    gdlce = reshape(ones(8,1)*gdle',[],1);
    K_el = reshape(K_el,[],1);
    K = [K; K_el];
    gdlf = [gdlf; gdlfe];
    gdlc = [gdlc; gdlce];
end
% Ensamblado de la matriz
K = sparse(gdlf,gdlc,K);
```

```
function Fint = fint_ld4Nsolid(TOP,XYZ,XYZdef,E,nu)
% Calcula el vector de fuerzas internas de cada
elementos y lo ensambla
% para formar el vector Fint global
% Define los puntos de integración en cada
direccion
ng = 3;
wg = [5/9 8/9 5/9];
xg = [-sqrt(3/5) 0 sqrt(3/5)];
% Matriz del material
lambda = E*nu/(1+nu)/(1-2*nu);
Gm = E/2/(1+nu);
%
dN_x = zeros(2,8);
B_L = zeros(3,8);
fel = [];
gdl = [];
for ie=1:size(TOP,1)
    % Nodos del elemento
    node = TOP(ie,:);
    % Coordenadas de los nodos del elemento
    Xe = XYZ(node,:);
    % Posición deformada de los nodos del elemento
    xe = XYZdef(node,:);
    % VEctor de fuerzas debido a la deformación
elástica
    fel_el = zeros(8,1);
```

```
    for ig=1:ng
        for jg=1:ng
            % Derivadas de las funciones de forma
en coordenadas locales
            dN = derShapeFunc4N_loc(xg(ig),xg(jg));
            % Matriz jacobiana de la transformación
            J(1,1) = dN(1,:) * xe(:,1);
            J(1,2) = dN(1,:) * xe(:,2);
            J(2,1) = dN(2,:) * xe(:,1);
            J(2,2) = dN(2,:) * xe(:,2);
            F = eye(2);
            for k=1:4
                dN_x(:,k) = J\dN(:,k);
                F = F - (xe(k,:)-
Xe(k,:))'*dN_x(:,k)';
            end
            F = inv(F);
            detJ = det(F);
            detj = det(J);
            for k=1:4
                B_L(1,2*k-1) = dN_x(1,k);
                B_L(1,2*k) = 0;
                B_L(2,2*k-1) = 0;
                B_L(2,2*k) = dN_x(2,k);
                B_L(3,2*k-1) = dN_x(2,k);
                B_L(3,2*k) = dN_x(1,k);
            end
            % Tensor de tensiones de Cauchy
            s = lambda/(2*detJ)*(detJ^2-1)*eye(2) +
Gm/detJ*(F*F'-eye(2));
            sv = [s(1,1); s(2,2); s(1,2)];
            % Fuerza interna en un elemento
            fel_el = fel_el +
wg(ig)*wg(jg)*B_L'*sv*detj ;
        end
    end
    % Información para ensamblado
    % Grados de libertad del elemento
    aux = [2*node'-1  2*node'];
    gdle = reshape(aux',[],1);
    fel = [fel; fel_el];
    gdl = [gdl; gdle];
end
```

```
% Ensamblado del vector:
Fint = accumarray(gdl,fel);
```
