Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Implementación de una nueva formulación de Deformación Plana Generalizada en un código de elementos finitos en MATLAB

Autor: Manuel Seco Ruiz

Tutor: Vladislav Mantič, José Reinoso

Grupo de Elasticidad y Resistencia de Materiales
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Implementación de una nueva formulación de Deformación Plana Generalizada en un código de elementos finitos en MATLAB

Autor:
Manuel Seco Ruiz

Tutor:
Vladislav Mantič
José Reinoso

Grupo de Elasticidad y Resistencia de Materiales
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Trabajo Fin de Grado: Implementación de una nueva formulación de Deformación Plana Generalizada en un código de elementos finitos en MATLAB

Autor: Manuel Seco Ruiz

Tutor: Antonio Blázquez, Vladislav Mantič

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia

Agradecimientos

Quiero mostrar mis agradecimientos en primer lugar a la Escuela Superior de Ingeniería de Sevilla (y, por ende, a la Universidad de Sevilla) por, a pesar de todo, haberme dado las herramientas y las posibilidades de estudiar una carrera universitaria, una ingeniería.

Doy las gracias también a todas las personas que he conocido en estos años, profesores y compañeros de clase, por todo lo que me han aportado. Pero, sobre todo, quiero dar las gracias a "Esiland", a la "Shavalería" y a los "Máquinas", no solo compañeros de clase, sino amigos.

No me quiero olvidar de los que me han acompañado en estos cinco años de caminar universitario, mi familia, los amigos del SARUS, mis experiencias erasmus en Suecia y toda la gente de los Equipos.

Mencionar de manera especial a mis tutores, Antonio Blázquez y Vladislav Mantič, por haberme guiado en este largo trabajo, y a José Reinoso, por su inestimable ayuda cada vez que la necesité. Extiendo este agradecimiento a todo el Grupo de Elasticidad y Resistencia de Materiales, por todos los recursos que nos han ofrecido a los estudiantes que hemos realizado el Trabajo Fin de Grado con ellos.

Resumen

El presente trabajo pretende desarrollar una nueva formulación del Problema Plano Generalizado (PPG) e implementarla en un código de elementos finitos en lenguaje MATLAB.

El PPG es un tipo de configuración específica en la cual la geometría se define por una sección transversal que se extiende de manera cilíndrica a lo largo de un eje y las cargas de dominio o de contorno y restricciones en los desplazamientos son de tal forma que las soluciones de tensiones y deformaciones dependen únicamente de las coordenadas del plano perpendicular al eje del cilindro. Por lo tanto, el problema tridimensional puede ser reducido a dos dimensiones.

Para la implementación de esta formulación se ha utilizado un elemento plano cuadrilátero serendípito de 8 nodos. En el capítulo 5 del trabajo se incluye un pseudocódigo, en el que se va explicando paso a paso las partes del código de elementos finitos. En el capítulo 6 del trabajo se valida la formulación sometiendo a elementos a problemas sencillos y comparando los resultados obtenidos numéricamente con los resultados analíticos.

En el futuro, este trabajo podrá ser continuado para desarrollar dicha formulación en un código de elementos finitos comercial como Abaqus ®, por ejemplo.

Abstract

This thesis tries to develop a new formulation of the Generalized Plane Problem (GPP) and implement it in a finite element code in MATLAB script.

The GPP is a specific configuration where the geometry is defined by a transversal cross section which is extended along an axis in a cylindrical way. The domain loads and contour loads and the boundary conditions are in a way that the stress and strain solutions only depend on the coordinates defining the plane perpendicular to the axis of the cylinder. Therefore, the three-dimensional problem can be expressed by a two-dimensional one.

For the implementation of this formulation, an 8-node serendipity element has been utilized. In chapter 5 a pseudocode is included, where the finite element code is explained step by step. In chapter 6, the formulation is validated by subjecting elements to some simple problems and comparing the numerical results to analytical results.

In the future, this thesis could be continued in order to implement the formulation in a commercial finite element code, such as Abaqus ™.

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

EL trabajo que se presenta consiste en la implementación de la formulación del Problema Plano Generalizado (PPG) en un código de Elementos Finitos. El elemento resultante de este trabajo es un elemento plano con capacidad de sufrir desplazamientos en la dirección normal al plano. Este elemento será útil para estudiar problemas de barras de material compuesto, en los cuales las cargas contenidas en el plano pueden provocar desplazamientos fuera de él debido a la naturaleza anisótropa del material.

El código utilizado para la implementación del PPG ha sido escrito en MATLAB ®. La intención del autor es continuar el desarrollo de este elemento en un futuro TFM, traduciendo el código a lenguaje Fortran e implementándolo en una subrutina para Abaqus ®.

2 ESTRUCTURA DEL TRABAJO

El trabajo se estructurará en tres bloques. En el primer bloque se realizará una presentación de toda la teoría que hay detrás de este trabajo, es decir, la formulación del Problema Plano Generalizado y la formulación del elemento serendípito de 8 nodos.

En el segundo bloque se desarrollará el procedimiento seguido para el cáculo de la matriz de rigidez de un elemento y se explicará el código de ensamblaje para problemas con más de un elemento.

El tercer bloque consiste en la validación de resultados. Se comprobará que la matriz de rigidez de un elemento ha sido correctamente calculada sometiendo el elemento a problemas sencillos, y comparando los resultados obtenidos con la solución analítica de dichos problemas. Además, se realizará un "test de la parcela" o "patch test" para comprobar la convergencia del elemento, es decir, que al aumentar el número de elementos la solución obtenida converge a la solución exacta.

3 EL PROBLEMA PLANO GENERALIZADO

Considérese un cuerpo homogéneo de forma cilíndrica recta y sección transversal constante. Se asume que el cuerpo tiene un comportamiento anisótropo. Sea D el dominio tridimensional completo, ∂D el contorno lateral del cilindro y ∂D^e las dos secciones extremas del cilindro si este es de longitud finita ($e=b$ o t, ya sea la sección extrema inferior o superior, respectivamente). La sección transversal del cilindro D determina un dominio bidimensional Ω, con contorno $\partial\Omega$ determinado por la sección transversal de ∂D (véase Figura 1.).

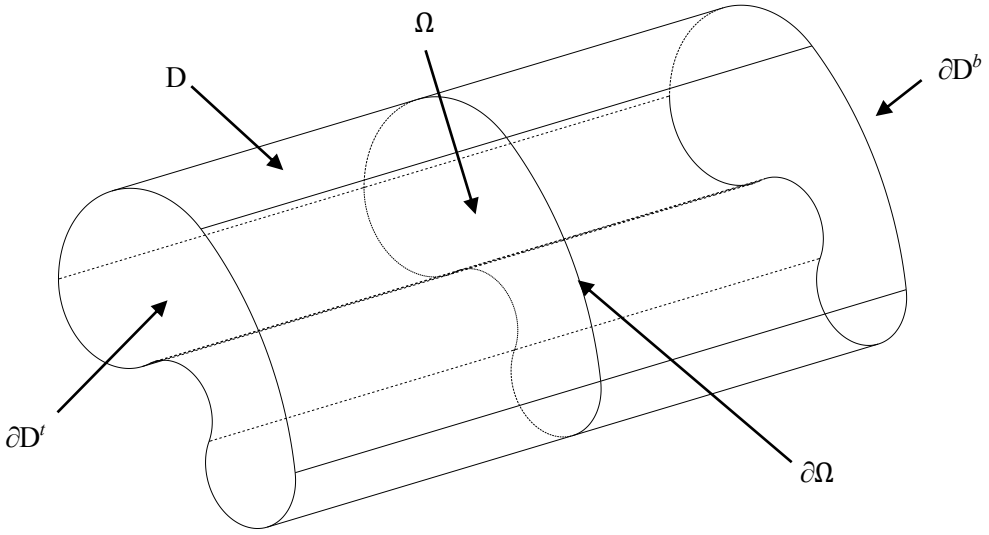

Figura 1. Descripción de la geometría

Se considera el caso en que este cuerpo estará sujeto a cargas en el dominio o en el contorno y restricciones en los desplazamientos de tal forma que las soluciones de tensiones y deformaciones dependan únicamente de las coordenadas del plano perpendicular al eje del cilindro. Si el eje longitudinal del cilindro es paralelo al eje x_3, entonces los tensores de tensiones y deformaciones pueden expresarse como:

$$\varepsilon_{ij} = \varepsilon_{ij}(x_1, x_2) \text{ y } \sigma_{ij} = \sigma_{ij}(x_1, x_2) \qquad i, j = 1,2,3 \tag{1}$$

La situación descrita anteriormente es denominada *Problema Plano Generalizado (PPG)*. Las situaciones de deformación plana, deformación plana generalizada, tensión plana generalizada, así como torsión libre pueden considerarse casos particulares del PPG.

3.1 Formulación del Problema Plano Generalizado (PPG)

Se integra la relación $\varepsilon - u$

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

a lo largo de la coordenada x_3, obteniéndose:

$$\varepsilon_{33} = u_{3,3} \Longrightarrow u_3(x_1, x_2, x_3) = x_3\varepsilon_{33} + \widetilde{U}_3(x_1, x_2)$$
$$\gamma_{13} = u_{1,3} + u_{3,1} \Longrightarrow u_1(x_1, x_2, x_3) = -\frac{x_3^2}{2}\varepsilon_{33,1} + x_3\left(\gamma_{13} - \widetilde{U}_{3,1}\right) + \widetilde{U}_1(x_1, x_2) \quad (3)$$
$$\gamma_{23} = u_{2,3} + u_{3,2} \Longrightarrow u_2(x_1, x_2, x_3) = -\frac{x_3^2}{2}\varepsilon_{33,2} + x_3\left(\gamma_{23} - \widetilde{U}_{3,2}\right) + \widetilde{U}_2(x_1, x_2)$$

donde $\widetilde{U}_i(x_1, x_2)$ $(i = 1,2,3)$ son funciones que sólo dependen de x_1 y x_2, y $\gamma_{ij} = 2\varepsilon_{ij}$ cuando $i \neq j$.

Sustituyendo estas expresiones en las restantes relaciones $\varepsilon - u$, e identificando términos de la misma potencia de x_3, se obtiene:

$$\left.\begin{matrix} \varepsilon_{11} = u_{1,1} \Longrightarrow u_{1,1} = x_3^2 \cdot 0 + x_3 \cdot 0 + \varepsilon_{11} \\ u_{1,1} = -\frac{x_3^2}{2}\varepsilon_{33,11} + x_3\left(\gamma_{13,1} - \widetilde{U}_{3,11}\right) + \widetilde{U}_{1,1}(x_1, x_2) \end{matrix}\right\} \Longrightarrow \begin{cases} \varepsilon_{33,11} = 0 \\ \gamma_{13,1} - \widetilde{U}_{3,11} = 0 \\ \widetilde{U}_{1,1} = \varepsilon_{11} \end{cases} \quad (4)$$

$$\varepsilon_{22} = u_{2,2} \Longrightarrow \begin{cases} \varepsilon_{33,22} = 0 \\ \gamma_{23,2} - \widetilde{U}_{3,22} = 0 \\ \widetilde{U}_{2,2} = \varepsilon_{22} \end{cases} \quad (5)$$

$$\gamma_{12} = u_{1,2} + u_{2,1} \Longrightarrow \begin{cases} \varepsilon_{33,12} = 0 \\ \gamma_{13,2} + \gamma_{23,1} - 2\widetilde{U}_{3,12} = 0 \\ \widetilde{U}_{1,2} + \widetilde{U}_{2,1} = \gamma_{12} \end{cases} \quad (6)$$

A partir de (4), (5) y (6) es fácil deducir que ε_{33} debe ser una función lineal de x_1 y x_2:

$$\varepsilon_{33} = Ax_1 + Bx_2 + C \quad (7)$$

donde A, B y C son constantes. Además, se tiene que:

$$\left(\gamma_{13} - \widetilde{U}_{3,1}\right)_{,1} = 0 \Longrightarrow \gamma_{13} - \widetilde{U}_{3,1} = f(x_2)$$
$$\left(\gamma_{23} - \widetilde{U}_{3,2}\right)_{,2} = 0 \Longrightarrow \gamma_{23} - \widetilde{U}_{3,2} = g(x_2) \quad (8)$$
$$f_{,2}(x_2) + g_{,1}(x_1) = \gamma_{13,2} + \gamma_{23,1} - 2\widetilde{U}_{3,12} = 0 \Longrightarrow f_{,2}(x_2) = -g_{,1}(x_1) = D$$

Sea μ_i el conjunto de movimientos como sólido rígido del problema. Entonces, sustituyendo las ecuaciones (7) y (8) en (3), se obtiene:

$$u_1(x_1, x_2, x_3) = -\frac{A}{2}x_3^2 + Dx_2x_3 + U_1(x_1, x_2) + \mu_1(x_2, x_3)$$
$$u_2(x_1, x_2, x_3) = -\frac{B}{2}x_3^2 - Dx_1x_3 + U_2(x_1, x_2) + \mu_2(x_1, x_3) \quad (9)$$
$$u_3(x_1, x_2, x_3) = (Ax_1 + Bx_2 + C)x_3 + U_3(x_1, x_2) + \mu_3(x_1, x_2)$$

que son las expresiones de la solución general en desplazamientos del PPG. Nótese que la única diferencia entre U_i y $\widetilde{U}_i$ es el movimiento como sólido rígido (μ_i).

Las constantes anteriores están asociadas a desplazamientos relativos entre secciones transversales del cilindro: C representa una elongación axial unitaria (deformación axial); A y B representan, respectivamente, curvaturas asociadas a los ejes x_2 y x_1 (flectores); y D representa un incremento en el giro de la sección respecto al eje x_3 (deformación torsional). Definiendo u_i^a como los desplazamientos en (9) gobernados por estas constantes, la expresión (9) se puede escribir en una forma más compacta:

$$u_i = u_i^a + U_i + \mu_i \quad (10)$$

Las funciones $U_1(x_1, x_2)$ y $U_2(x_1, x_2)$ representan los desplazamientos dentro del plano de la sección transversal que no varían a lo largo del eje x_3. $U_3(x_1, x_2)$ es la función de alabeo. [1]

4 EL ELEMENTO SERENDÍPITO DE 8 NODOS

El tipo de elemento finito que se va a utilizar en este proyecto es un elemento plano cuadrilátero serendípito de 8 nodos. Además, el elemento posee 9 puntos de integración o puntos de Gauss, situados en dominio del elemento como se describe en la Tabla 1:

Gauss point	1	2	3	4	5	6	7	8	9
Peso w_i	25/81	25/81	25/81	25/81	64/81	25/81	25/81	25/81	25/81
ξ_1	$-\sqrt{3/5}$	0	$\sqrt{3/5}$	$-\sqrt{3/5}$	0	$\sqrt{3/5}$	$-\sqrt{3/5}$	0	$\sqrt{3/5}$
ξ_2	$-\sqrt{3/5}$	$-\sqrt{3/5}$	$-\sqrt{3/5}$	0	0	0	$\sqrt{3/5}$	$\sqrt{3/5}$	$\sqrt{3/5}$

Tabla 1. Coordenadas y pesos de los 9 puntos de Gauss del elemento

En la Figura 2 se describe la geometría del elemento y su configuración en el espacio isoparamétrico, formado por las coordenadas naturales ξ_1 y ξ_2.

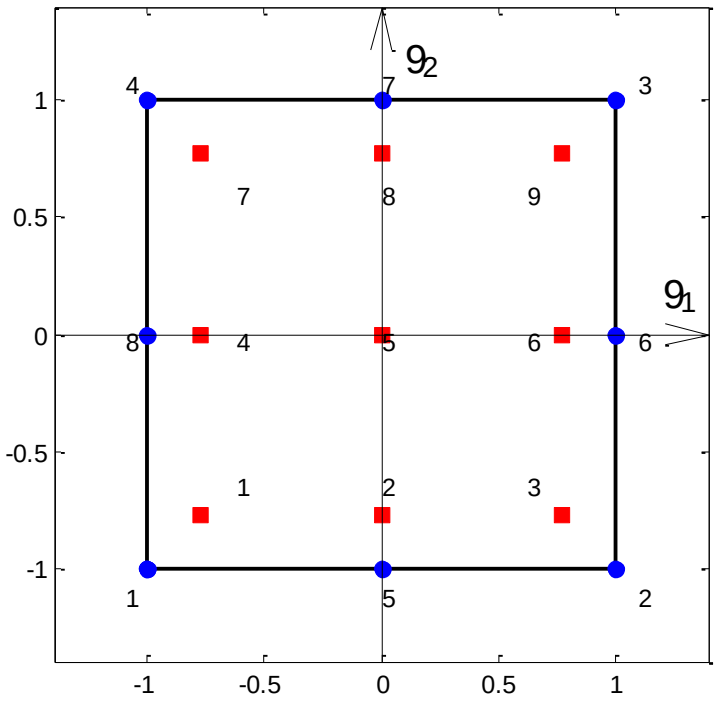

Figura 2. Elemento serendípito de 8 nodos (●) y 9 puntos de integración (■) en el espacio natural

Este elemento es isoparamétrico, es decir, que utiliza las mismas funciones de forma para interpolar la geometría que para interpolar el campo de variables primarias (en este caso, los desplazamientos). Dichas funciones de forma son construidas atendiendo al criterio de que una función de forma debe valer 1 en su nodo asociado y 0 en el resto [2].

Las funciones de forma de los nodos situados en los vértices son las expresadas en la ecuación (11):

$$
\begin{aligned}
N_1(\xi_1,\xi_2) &= -\frac{1}{4}(1-\xi_1)(1-\xi_2)(1+\xi_1+\xi_2)\\
N_2(\xi_1,\xi_2) &= -\frac{1}{4}(1+\xi_1)(1-\xi_2)(1-\xi_1+\xi_2)\\
N_3(\xi_1,\xi_2) &= -\frac{1}{4}(1+\xi_1)(1+\xi_2)(1-\xi_1-\xi_2)\\
N_4(\xi_1,\xi_2) &= -\frac{1}{4}(1-\xi_1)(1+\xi_2)(1+\xi_1-\xi_2)
\end{aligned}
\qquad (11)
$$

Por otro lado, las funciones de forma asociadas a los nodos situados en el punto medio de los lados del elemento son las indicadas a continuación en la ecuación (12):

$$
\begin{aligned}
N_5(\xi_1,\xi_2) &= \frac{1}{2}(1-\xi_1)(1+\xi_1)(1-\xi_2)\\
N_6(\xi_1,\xi_2) &= \frac{1}{2}(1+\xi_1)(1+\xi_2)(1-\xi_2)\\
N_7(\xi_1,\xi_2) &= \frac{1}{2}(1-\xi_1)(1+\xi_1)(1+\xi_2)\\
N_8(\xi_1,\xi_2) &= \frac{1}{2}(1-\xi_1)(1+\xi_2)(1-\xi_2)
\end{aligned}
\qquad (12)
$$

Una representación gráfica de las funciones de forma puede ser apreciada en la Figura 3.

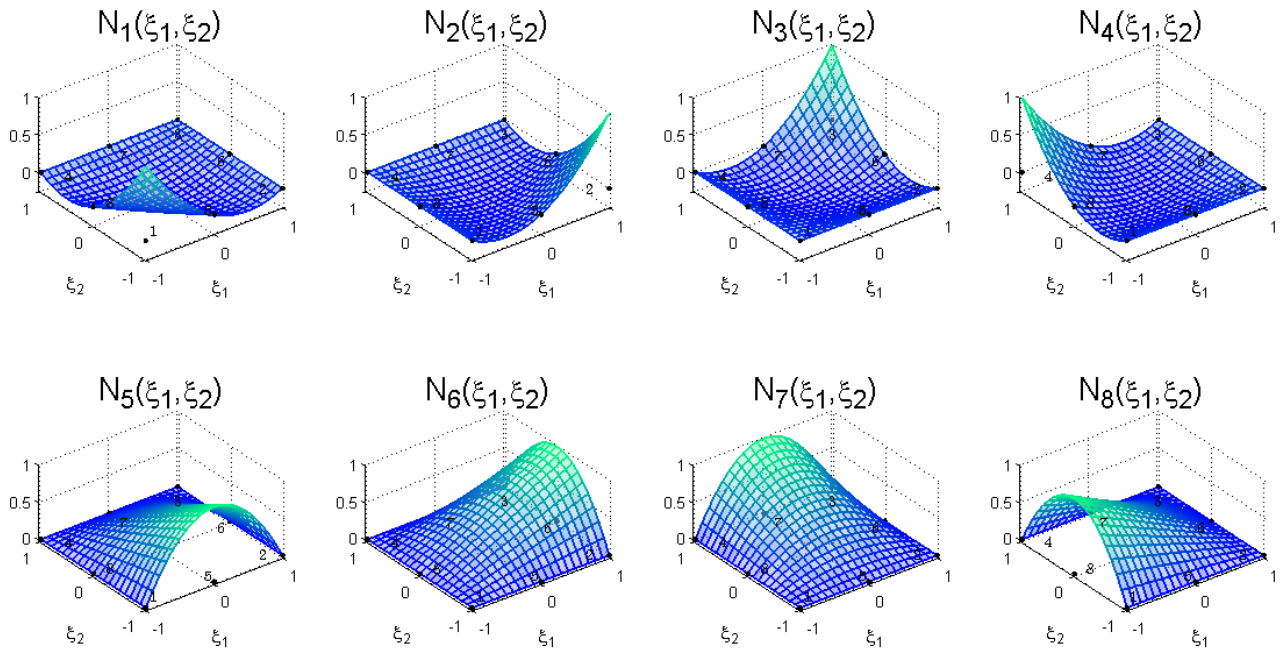

Figura 3. Funciones de forma del elemento representadas en el espacio natural

5 PSEUDOCÓDIGO

5.1 Cálculo de la matriz de rigidez del elemento

El cálculo de la matriz de rigidez de un elemento con comportamiento isótropo es realizado por una función llamada `StiffnessMatrixISO(nodes,elements,iel)`. Los argumentos de entrada son:

- `nodes`, es una matriz que contiene, por cada nodo, su número identificativo y sus coordenadas en el sistema de coordenadas globales.
- `elements`, es una matriz que contiene, por cada elemento, su número identificativo y su conectividad (qué nodos definen ese elemento).
- `iel`, es un contador que indica el elemento del cual se va a calcular la matriz de rigidez.

Como argumento de salida, la función devuelve `[Kelement]`, que es la matriz de rigidez del elemento `iel`.

5.1.1 Inicialización de variables

Las primeras líneas de esta función constituyen un bloque en el cual se inicializan algunas variables que serán de utilidad a lo largo del código:

```
nnode=8; %número de nodos
ndim =2; %dimensión
ndof=3; %número de grados de libertad
h=1.0; %espesor del elemento
density=0;
fy=zeros(1,nnode); % Vector de fuerzas asociado a cargas de
volumen
ngauss=3; %número de puntos de integración por dirección
Kelement=zeros(28);
```

5.1.2 Cálculo de las coordenadas y pesos de los puntos de Gauss

Posteriormente, se llama a la función `[posgp,weightgp]=kgauss(ngauss)`, que devuelve los pesos y las coordenadas en el espacio natural de cada punto de Gauss. Se abren dos bucles que irán recorriendo estos 9 puntos de integración; en cada paso del bucle se calculará la aportación que se realizará a la matriz de rigidez, y se irá acumulando como se detallará más adelante.

```
%----------  BUCLE SOBRE LOS PUNTOS DE INTEGRACIÓN
      kip = 0;                                  %contador

      for i = 1:ngauss                    %bucle sobre xi_1 (xi)
        xi =  posgp(i);
        wxi =  weightgp(i);

        for  j = 1:ngauss                 %bucle sobre xi_2 (eta)
            eta =  posgp(j);
            weta =  weightgp(j);


                kip = kip + 1; %Contador del punto de integración
```

5.1.3 Evaluación de las funciones de forma y sus derivadas parciales

El primer paso a seguir una vez se saben las coordenadas del punto de integración actual (`xi` y `eta` para `kip`) es evaluar las funciones de forma y sus derivadas parciales en dicho punto, puesto que será necesario utilizarlas a lo largo del código. La función `kshapefunctions2` contiene las funciones de forma detalladas en las ecuaciones (11) y (12), y devuelve los valores de todas las funciones de forma y sus derivadas parciales en cualquier punto del elemento. Nótese que, de momento, todos estos cálculos se realizan en el espacio natural.

```
%Evalúa las funciones de forma y sus derivadas
[shapef,dshape] =kshapefunctions2(xi,eta,ndim,nnode);
%-------------------------------------------------
```

5.1.4 Cálculo del jacobiano

A continuación, se calcula el jacobiano en el punto de integración. El jacobiano (y su inversa) será necesario para resolver las integrales del cálculo de la matriz de rigidez mediante cuadratura de Gauss (para realizar la transformación de coordenadas naturales a coordenadas reales) y para poder aplicar la regla de la cadena a la hora de expresar las derivadas de las funciones de forma en el espacio real. La función `kJacobian2D` calcula la matriz jacobiana en el punto donde se han evaluado previamente las funciones de forma:

```
%-------------------------------------------------
% Calcula el Jacobiano y su inversa
[xjac,xjacinv,detxjac]=kJacobian2D(ndim,nnode,shapef,dshape,nodes
_coord);
%-------------------------------------------------
```

El jacobiano es una matriz cuyos componentes son las derivadas de las coordenadas cartesianas respecto a las coordenadas naturales:

$$J(\xi_1,\xi_2)=\frac{\partial(x,y)}{\partial(\xi_1,\xi_2)}=\begin{bmatrix}\frac{\partial x}{\partial \xi_1} & \frac{\partial y}{\partial \xi_1}\\ \frac{\partial x}{\partial \xi_2} & \frac{\partial y}{\partial \xi_2}\end{bmatrix} \qquad (13)$$

Para el caso de los elementos cuadriláteros, el operador de compatibilidad B (como se verá más adelante), no admite una forma algebraica concreta y debe ser evaluado a partir de las funciones de forma. De igual manera se ha de actuar con la matriz jacobiana, ya que esta se empleará en el cálculo de las componentes de la matriz B. Así, haciendo uso del concepto de discretización, los términos del jacobiano pueden expresarse:

$$x\approx\sum_{A=1}^{n_n}N_A(\xi_1,\xi_2)x_A\,;\; y\approx\sum_{A=1}^{n_n}N_A(\xi_1,\xi_2)y_A \qquad (14)$$

$$\begin{aligned}\frac{\partial x}{\partial \xi_i}&\approx\sum_{A=1}^{n_n}\frac{\partial N_A(\xi_1,\xi_2)}{\partial \xi_i}x_A\,;i=1{,}2\\ \frac{\partial y}{\partial \xi_i}&\approx\sum_{A=1}^{n_n}\frac{\partial N_A(\xi_1,\xi_2)}{\partial \xi_i}y_A\,;i=1{,}2\end{aligned} \qquad (15)$$

Siendo n_n el número de nodos (8) del elemento, N_A la función de forma del nodo A, y (x_A, y_A) las coordenadas del nodo A en el espacio real. Lo expresado en las ecuaciones (13) y (15) se implementa en las líneas de

código de la función `kJacobian2D` que se muestran a continuación:

```
function
[xjac,xjacinv,detxjac]=kJacobian2D(ndim,nnode,shapef,dshape,xref)

%Fase de inicialización
       for i=1:ndim
        for j=1:ndim
       xjac(i,j)=0.0;
       xjacinv(i,j)=0.0;
        end
       end

      % xref son las coordenadas de los nodos en el espacio real
      for i=1:nnode
      xjac(1,1) = xjac(1,1) + dshape(1,i)*xref(1,i);
      xjac(1,2) = xjac(1,2) + dshape(1,i)*xref(2,i);

      xjac(2,1) = xjac(2,1) + dshape(2,i)*xref(1,i);
      xjac(2,2) = xjac(2,2) + dshape(2,i)*xref(2,i);
      end

      %Calcula el determinante del jacobiano
      detxjac=det(xjac);

      if(detxjac > 0.0)  %Puede formularse la inversa
        %Calcula la inversa
        xjacinv = inv(xjac);
      else
      end
end
```

5.1.5 Cálculo del operador de compatibilidad B

Una vez se han obtenido los valores de las funciones de forma (en coordenadas naturales) y la matriz jacobiana (que permite expresar estas coordenadas naturales en coordenadas reales), se procede al cálculo de la matriz B u operador de compatibilidad. En este caso, la matriz B adopta la siguiente forma:

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{\partial N_1}{\partial x} & 0 & 0 & \dots & \frac{\partial N_8}{\partial x} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\partial N_1}{\partial y} & 0 & \dots & 0 & \frac{\partial N_8}{\partial y} & 0 \\ x & y & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & -x & 0 & 0 & \frac{\partial N_1}{\partial y} & \dots & 0 & 0 & \frac{\partial N_8}{\partial y} \\ 0 & 0 & 0 & y & 0 & 0 & \frac{\partial N_1}{\partial x} & \dots & 0 & 0 & \frac{\partial N_8}{\partial x} \\ 0 & 0 & 0 & 0 & \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & 0 & \dots & \frac{\partial N_8}{\partial y} & \frac{\partial N_8}{\partial x} & 0 \end{bmatrix} \quad (16)$$

Siendo (x, y) las coordenadas del punto de Gauss en el que se está evaluando la matriz. Por tanto, hay que transformar las coordenadas (ξ_1, ξ_2) en (x, y) a través de la ecuación (14). Las derivadas parciales de las funciones de forma en la matriz B son respecto a las coordenadas reales, mientras que las ya conocidas y calculadas en la matriz `dshape` son las derivadas respecto a las coordenadas naturales. Para realizar la transformación, se aplica la regla de la cadena:

$$\frac{\partial N_i}{\partial x} = \frac{\partial N_i}{\partial \xi_1}\frac{\partial \xi_1}{\partial x} + \frac{\partial N_i}{\partial \xi_2}\frac{\partial \xi_2}{\partial x}$$
$$\frac{\partial N_i}{\partial y} = \frac{\partial N_i}{\partial \xi_1}\frac{\partial \xi_1}{\partial y} + \frac{\partial N_i}{\partial \xi_2}\frac{\partial \xi_2}{\partial y} \quad (17)$$

Los términos $\frac{\partial N_i}{\partial \xi_j}$ están almacenados en la matriz `dshape`, mientras que los términos $\frac{\partial \xi_1}{\partial x}$ son las componentes de la matriz jacobiana inversa ($\boldsymbol{J}^{-1}$), almacenadas en la matriz `xjacinv`. A continuación se muestra el código de la función `kBmat2DLin`, en el cual se implementa todo el cálculo desarrollado en este subapartado:

```
function [Bmat2] =
kBmat2DLin(ndim,nnode,ngauss,shapef,dshape,xjac,xjacinv,xi,eta,xr
ef)

% xi y eta son coordenadas naturales, hay que transformarlas al
espacio
% real: x1_real=N1(xi,eta)*x1_nodo1+N2(xi,eta)*x1_nodo2...
x1_real=0;
x2_real=0;
for i=1:8
    x1_real=x1_real+shapef(i)*xref(1,i);
    x2_real=x2_real+shapef(i)*xref(2,i);
end

    %inicialización
      Bmat=zeros(6,ndim*nnode);

       % El operador lineal B tiene el siguiente aspecto
       % 0   0   0 0 | Nk,x     0       0|
       % 0   0   0 0 |   0     Nk,y    0|
       % x1  x2  1 0 |   0      0      0|

       % 0 0 0 -x1 |   0       0    Nk,x|
       % 0 0 0  x2 |   0       0    Nk,y|
       % 0 0 0  0  | Nk,y    Nk,x     0|

       % dN     dN  dxi     dN  deta
       %---- = ----*---- + ----*----
       % dx     dxi  dx    deta  dx

      kint = 1;
      for i=1:nnode
       Bmat(1,kint)   = dshape(1,i)*xjacinv(1,1)+ ...
                        dshape(2,i)*xjacinv(1,2);

       Bmat(2,kint+1)= dshape(1,i)*xjacinv(2,1)+...
                        dshape(2,i)*xjacinv(2,2);

       % 0    0     Nk,2|
       Bmat(4,kint+2)    = Bmat(2,kint+1);

       %   0    0   Nk,1|
       Bmat(5,kint+2) = Bmat(1,kint);

       % Nk,2    Nk,1  0|
```

```
        Bmat(6,kint)   = Bmat(2,kint+1);
        Bmat(6,kint+1) = Bmat(1,kint);


          kint = kint + 3;
        end

        A=zeros(6,4);
        A(3,1)=x1_real;
        A(3,2)=x2_real;
        A(3,3)=1;
        A(4,4)=-x1_real;
        A(5,4)=x2_real;

Bmat2=[A Bmat];
```

5.1.6 Cálculo de la matriz de rigidez

La matriz de rigidez es el resultado del siguiente cálculo:

$$k^e = \int_{\Omega^e} \boldsymbol{B^T C B} d\Omega \tag{18}$$

Ω^e es el dominio del elemento. La matriz $\boldsymbol{B}$ contiene las derivadas de las funciones de forma y su cálculo se ha detallado en el apartado 5.1.5. La matriz $\boldsymbol{C}$ es la matriz constitutiva del material, y contiene las constantes elásticas que caracterizan el comportamiento del material. En un primer caso, se va a tomar un material isótropo por simplicidad, por lo que en la matriz $\boldsymbol{C}$ solo entrarán en juego dos constantes elásticas (E y ν). La matriz constitutiva será de la siguiente manera:

$$\boldsymbol{C} = \begin{bmatrix} \frac{1}{E} & \frac{-\nu}{E} & \frac{-\nu}{E} & 0 & 0 & 0 \\ \frac{-\nu}{E} & \frac{1}{E} & \frac{-\nu}{E} & 0 & 0 & 0 \\ \frac{-\nu}{E} & \frac{-\nu}{E} & \frac{1}{E} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{2(1+\nu)}{E} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{2(1+\nu)}{E} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{2(1+\nu)}{E} \end{bmatrix}^{-1} \tag{19}$$

5.1.6.1 Cuadratura de Gauss

Para realizar el cálculo de la integral en la ecuación (18) de una manera computacionalmente eficiente y con un excelente grado de aproximación, se utiliza la técnica de la cuadratura de Gauss, que a continuación se detalla:

$$\int_{-1}^{1}\int_{-1}^{1} F(\xi_1,\xi_2)d\xi_1 d\xi_2 = \sum_{i}^{n_{pg_1}} \sum_{j}^{n_{pg_2}} w_i w_j F\left(\xi_{1_i},\xi_{2_j}\right) \tag{20}$$

Como se aprecia en la ecuación (20), la técnica de la cuadratura de Gauss para realizar integrales consiste

en calcular dicha integral mediante la suma de una serie de productos de uno pesos asignados a los puntos de Gauss ($w_i w_j$) por el valor de la función en dicho punto de Gauss ($F\left(\xi_{1_i}, \xi_{2_j}\right)$). La suma de este producto en cada punto de integración da como resultado la integral de F en el dominio del elemento.

La integral que da como resultado la matriz de rigidez se resuelve en el espacio isoparamétrico (ξ_1, ξ_2), ya que en él los límites de integración son muy sencillos (en el espacio isoparamétrico todos los elementos son cuadrados de lado 2, y los límites de integración van de -1 a 1, mientras que en el espacio real los límites de integración serían los contornos parabólicos del elemento real). Para expresar esta integral en el espacio isoparamétrico, hay que transformar el diferencial de área de elemento $d\Omega$, siguiendo la siguiente relación:

$$d\Omega = dxdy = Jd\xi_1 d\xi_2 \qquad (21)$$

Siendo J el determinante de la matriz jacobiana. El cálculo de la matriz de rigidez se realiza en las siguientes líneas de código:

```
%-------- CmatVoigt -------------
E=210e9; %acero
nu=0.3; %acero
CmatVoigt=inv([1/E -nu/E -nu/E 0 0 0
               -nu/E 1/E -nu/E 0 0 0
               -nu/E -nu/E 1/E 0 0 0
                0 0 0 2*(1+nu)/E 0 0
                0 0 0 0 2*(1+nu)/E 0
                0 0 0 0 0 2*(1+nu)/E]);
%---------------------------------
factor=wxi*weta*h*detxjac; %wi*wj*|J|
Kloop=BmatLin'*CmatVoigt*BmatLin*factor; % aportación a la matriz
de rigidez del punto de integración "kip"
Kelement=Kelement+Kloop; % acumulación de la matriz de rigidez
del elemento
```

`Kelement` es la matriz del elemento calculada por la función `StiffnessMatrixISO` y es el argumento de salida. Una vez que se ha implementado la rutina que calcula la matriz de rigidez de un elemento, se procede a la programación de una rutina de ensamblaje, para resolver problemas con más de un elemento.

5.2 Rutina de ensamblaje

La rutina de ensamblaje tiene como objetivo calcular la matriz de rigidez global de una malla de elementos. Esta se realiza calculando la matriz de rigidez de cada elemento y añadiendo la aportación de esta matriz elemental a la matriz global en las posiciones que correspondan a los grados de libertad del elemento en cuestión.

Se comienza declarando dos variables globales, ya que serán utilizadas en esta rutina y en varias subrutinas. Estas dos variables almacenarán una matriz de nodos (número de nodo y sus coordenadas) y otra matriz de elementos (número de elemento y nodos que lo forman, la conectividad), como se muestra a continuación:

```
clear all
close all
global elements
global nodes
```

A continuación se procede a la generación de las matrices `elements` ($n_e \times 9$) y `nodes` ($n_n \times 4$). Esto se realiza importando los datos desde un fichero .inp de Abaqus®.

Es decir, previamente se ha debido crear un malla de nodos y elementos utilizando el software de elementos finitos Abaqus®. Una vez creada, se genera el archivo .inp correspondiente que contiene, entre otras cosas, las matrices deseadas, `elements` y `nodes`. La lectura del archivo .inp y la extracción de dichas matrices es realizada por la función `[nodes,elements] =readInp('')`. El código de esta subrutina puede ser encontrado en el Anexo 1. Una vez obtenidas las matrices anteriores, se continúa con la generación de una serie de variables que serán necesarias a lo largo del código:

```
[nodes,elements]=readInp('Job-2.inp.txt'); % Función que lee un
inp de Abaqus y devuelve los nodos y la conectividad de los
elementos
numnode=length(nodes(:,1)); %número de nodos
numelem=length(elements(:,1)); %número de elementos
ndof=3; %número de grados de libertad de cada nodo
%total number of unknowns = number of nodes of the model * number
of dof per node
totalUnknown = numnode*ndof+4; % +4 para A,B,C,D
desp=zeros(totalUnknown,1);
F_ext = zeros(totalUnknown,1); %vector de fuerzas externas
F_int = zeros(totalUnknown,1); %vector de fuerzas internas
Kglobal = zeros(totalUnknown,totalUnknown); % Kglobal = K;
%matriz de rigidez global
Fglobal=zeros(totalUnknown,1);
```

Una vez generadas las variables anteriores, se entra en un bucle que recorre todos los elementos uno por uno. En cada pasada se calculan las posiciones que corresponden a ese elemento dentro de la matriz de rigidez global, se calcula la matriz de rigidez de ese elemento y se asignan los valores correspondientes a cada posición dentro de la matriz de rigidez global. Se muestra el código:

```
% bucle sobre los elementos
for iel = 1 : numelem %contador de elementos
    sctrB = assemblyPHN(iel); %función que devuelve los índices
de la matriz global en función de la conectividad de los
elementos
    Kedd = zeros(length(sctrB),length(sctrB));
    Fedd = zeros(length(sctrB),1);

    [Kedd]=StiffnessMatrixISO(nodes,elements,iel); % Devuelve
Kedd (matriz de rigidez del elemento "iel") incluyendo A,B,C,D


 %-------------- Assembly ----------------------
    Kglobal(sctrB,sctrB) = Kglobal(sctrB,sctrB) + Kedd;
    F_ext(sctrB,1) = Fglobal(sctrB,1) + Fedd;
end
```

`iel` es el contador de elementos, que va desde el elemento 1 hasta el `numelem`. La variable `sctrB` es un vector que contiene las posiciones que corresponden al elemento `iel` dentro de la matriz de rigidez global. Este vector se calcula en la función `assemblyPHN(iel,elements)`, que se detalla a continuación:

```
function [sctrB] = assemblyPHN(e)

global elements % esta variable global "elements" contiene el
número del element y su conectividad

sctr = elements(e,2:9);
nn = length(sctr);
```

```
% sctr contiene los nodos que conforman el elemento e. A
continuación se va
% a construir un vector a partir de estos nodos, cuyas
componentes
% representan las posiciones de los 3 grados de libertad de
dichos nodos en
% la matriz global. Estas posiciones serán 3*nodo+4(A,B,C,D) y -
2(si es el
% grado de libertad x), -1(si es el grado de libertad y) o 0 (si
es elgrado de libertad z)
for k = 1 : nn
    sctrBfem(3*k-2) = 3*sctr(k)-2+4; %3 is the number of degrees
of freedom per node
    sctrBfem(3*k-1) = 3*sctr(k)-1+4;
    sctrBfem(3*k) = 3*sctr(k)+4;
end
sctrB=[1 2 3 4 sctrBfem]; % a los grados de libertad de los
nodos, se añaden los 4 grados de libertad del nodo "maestro"
% de tal forma que se acumulan en las cuatro primeras posiciones
las aportaciónes de cada elemento a los grados
% de libertad A, B, C y D.
```

Una vez conocidas las posiciones dentro de la matriz de rigidez global, se calcula la matriz de rigidez del elemento `iel` y se introducen en la matriz de rigidez global en las posiciones que corresponden:

```
[Kedd]=StiffnessMatrixISO(nodes,elements,iel);

Kglobal(sctrB,sctrB) = Kglobal(sctrB,sctrB) + Kedd;
```

6 VALIDACIÓN DE RESULTADOS

En este apartado se va a proceder a la validación de la matriz de rigidez de un elemento para comprobar que está correctamente calculada. Esto se va a realizar mediante varios problemas "test" con comportamiento isótropo, resolviéndolos a mano y con el elemento, y comprobando que el resultado es el mismo. El elemento tiene unas dimensiones de 1 x 1 y unas constantes elásticas de $E = 210e9\ Pa$ y $\nu = 0.3$.

6.1 Ensayo de tracción

Se aplica una tensión de tracción de $5\ GPa$ a un elemento de dimensiones $1m \times 1m$ (Figura 5) en su lado superior, teniendo apoyo de bolas en el lado inferior y estando impedidos los movimientos como sólido rígido.

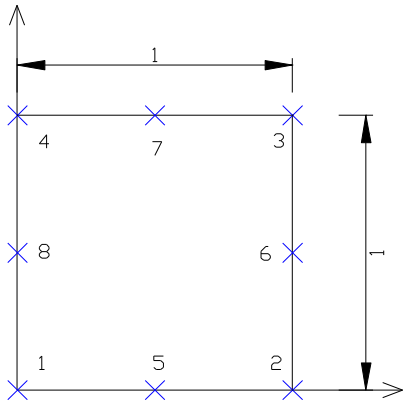

Figura 5. Elemento sometido a ensayo de tracción

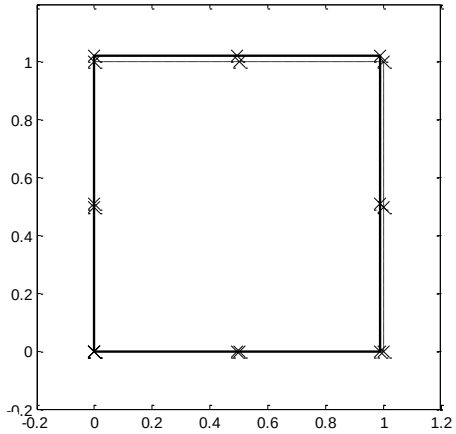

Figura 4.Situación deformada (−) e indeformada (- - -)

La solución analítica es la siguiente:

$$\sigma_{ij} = \begin{pmatrix} 0 & 0 \\ 0 & 5 \cdot 10^9 \end{pmatrix} \Longrightarrow \varepsilon_{ij} = \begin{pmatrix} -9.2857 \cdot 10^{-3} & 0 \\ 0 & 0.021667 \end{pmatrix} \Longrightarrow \begin{cases} u_x = -9.2857 \cdot 10^{-3}\, x \\ u_y = 0.021667\, y \end{cases} \quad (22)$$

Los resultados en desplazamientos (en metros) son:

	Solución numérica		Solución analítica		Error relativo	
Nodo	u_x	u_y	u_x	u_y	u_x	u_y
1	0	0	0	0	0	0
2	-0,00928571	0	-0,00928571	0	5,60E-16	0
3	-0,00928571	0,02166667	-0,00928571	0,02166667	3,74E-16	4,80E-16
4	0	0,02166667	0	0,02166667	0	6,41E-16
5	-0,00464286	0	-0,00464286	0	9,34E-17	0
6	-0,00928571	0,01083333	-0,00928571	0,01083333	0	4,00E-16

7	-0,00464286	0,02166667	-0,00464286	0,02166667	5,60E-16	3,20E-16
8	-6,26E-18	0,01083333	0	0,01083333	6,74E-16	1,60E-16

Tabla 2. Comparación de resultados del ensayo 6.1

6.2 Bitracción

A un elemento de las mismas dimensiones que el del problema 6.1 se le aplica el siguiente campo de desplazamientos en los nodos:

$$\begin{cases} u_x = x \\ u_y = y \\ u_z = 0 \end{cases} \Longrightarrow \varepsilon_{ij} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \Longrightarrow \sigma_{ij} = \begin{pmatrix} 4.0385 \cdot 10^{11} & 0 & 0 \\ 0 & 4.0385 \cdot 10^{11} & 0 \\ 0 & 0 & 2.4231 \cdot 10^{11} \end{pmatrix} \quad (23)$$

A coontinuación se muestra la comparación de resultados analíticos y numéricos:

Nodo		1	2	3	4	5	6	7	8
σ_{xx}	Analítica	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11
	Numérica	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11
	Diferencia	1.5113e-16	1.5113e-16	1.5113e-16	1.5113e-16	1.5113e-16	1.5113e-16	1.5113e-16	1.5113e-16
σ_{yy}	Analítica	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11
	Numérica	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11	4.0385e+11
	Diferencia	1.5113e-16	1.5113e-16	1.5113e-16	1.5113e-16	1.5113e-16	1.5113e-16	1.5113e-16	1.5113e-16
σ_{zz}	Analítica	2.4231e+11	2.4231e+11	2.4231e+11	2.4231e+11	2.4231e+11	2.4231e+11	2.4231e+11	2.4231e+11
	Numérica	2.4231e+11	2.4231e+11	2.4231e+11	2.4231e+11	2.4231e+11	2.4231e+11	2.4231e+11	2.4231e+11
	Diferencia	1.5113e-16	1.5113e-16	1.5113e-16	1.5113e-16	1.5113e-16	1.5113e-16	1.5113e-16	1.5113e-16
σ_{xy}	Analítica	0	0	0	0	0	0	0	0
	Numérica	0	0	0	0	0	0	0	0
	Diferencia	0	0	0	0	0	0	0	0
σ_{xz}	Analítica	0	0	0	0	0	0	0	0
	Numérica	0	0	0	0	0	0	0	0
	Diferencia	0	0	0	0	0	0	0	0
σ_{yz}	Analítica	0	0	0	0	0	0	0	0
	Numérica	2.6902e-05	2.6902e-05	2.6902e-05	2.6902e-05	2.6902e-05	2.6902e-05	2.6902e-05	2.6902e-05
	Diferencia	6.6613e-17	6.6613e-17	6.6613e-17	6.6613e-17	6.6613e-17	6.6613e-17	6.6613e-17	6.6613e-17

Tabla 3. Comparación entre resultados analíticos y numéricos del problema 6.2

6.3 Ensayo de cortadura

Se impone el siguiente campo de desplazamientos a un elemento similar al de los problemas anteriores:

$$\begin{cases} u_x = y \\ u_y = x \\ u_z = 0 \end{cases} \Longrightarrow \varepsilon_{ij} = \begin{pmatrix} 0 & 2 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \Longrightarrow \sigma_{ij} = \begin{pmatrix} 0 & 1.6154 \cdot 10^{11} & 0 \\ 1.6154 \cdot 10^{11} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (24)$$

Nodo		1	2	3	4	5	6	7	8
σ_{xx}	Analítica	0	0	0	0	0	0	0	0
	Numérica	7.3979e-05	7.3979e-05	7.3979e-05	7.3979e-05	7.3979e-05	7.3979e-05	7.3979e-05	7.3979e-05
	Diferencia	4.5797e-16	4.5797e-16	4.5797e-16	4.5797e-16	4.5797e-16	4.5797e-16	4.5797e-16	4.5797e-16
σ_{yy}	Analítica	0	0	0	0	0	0	0	0
	Numérica	-6.7254e-06	-6.7254e-06	-6.7254e-06	-6.7254e-06	-6.7254e-06	-6.7254e-06	-6.7254e-06	-6.7254e-06
	Diferencia	4.1633e-17	4.1633e-17	4.1633e-17	4.1633e-17	4.1633e-17	4.1633e-17	4.1633e-17	4.1633e-17
σ_{zz}	Analítica	0	0	0	0	0	0	0	0
	Numérica	2.0176e-05	2.0176e-05	2.0176e-05	2.0176e-05	2.0176e-05	2.0176e-05	2.0176e-05	2.0176e-05
	Diferencia	1.249e-16	1.249e-16	1.249e-16	1.249e-16	1.249e-16	1.249e-16	1.249e-16	1.249e-16
σ_{xz}	Analítica	0	0	0	0	0	0	0	0
	Numérica	0	0	0	0	0	0	0	0
	Diferencia	0	0	0	0	0	0	0	0
σ_{yz}	Analítica	0	0	0	0	0	0	0	0
	Numérica	0	0	0	0	0	0	0	0
	Diferencia	0	0	0	0	0	0	0	0
σ_{xy}	Analítica	1.6154e+11	1.6154e+11	1.6154e+11	1.6154e+11	1.6154e+11	1.6154e+11	1.6154e+11	1.6154e+11
	Numérica	1.6154e+11	1.6154e+11	1.6154e+11	1.6154e+11	1.6154e+11	1.6154e+11	1.6154e+11	1.6154e+11
	Diferencia	1.8892e-16	1.8892e-16	1.8892e-16	1.8892e-16	1.8892e-16	1.8892e-16	1.8892e-16	1.8892e-16

Tabla 4. Comparación entre resultados numéricos y analíticos del problema 6.3

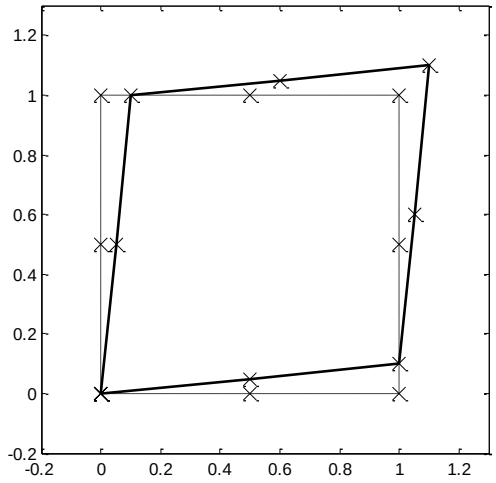

Figura 6. Configuración deformada e indeformada del problema 6.3

6.4 Campo de desplazamientos parabólico

En esta ocasión el campo de desplazamientos que se impone en los nodos del elemento es parabólico:

$$\begin{cases} u_x = x^2 \\ u_y = 0 \\ u_z = 0 \end{cases} \Longrightarrow \varepsilon_{ij} = \begin{pmatrix} 2x & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \Longrightarrow \sigma_{ij} = \begin{pmatrix} 5.6538 \cdot 10^{11} x & 0 & 0 \\ 0 & 2.4231 \cdot 10^{11} x & 0 \\ 0 & 0 & 2.4231 \cdot 10^{11} x \end{pmatrix} \qquad (25)$$

Nodo		1	2	3	4	5	6	7	8
σ_{xx}	Analítica	0	5,6538E+11	5,6538E+11	0	2,8269E+11	5,6538E+11	2,8269E+11	0
	Numérica	0	5,6538E+11	5,6538E+11	0	2,8269E+11	5,6538E+11	2,8269E+11	0
	Diferencia	0	0	0	0	0	0	0	0
σ_{yy}	Analítica	0	2,4231E+11	2,4231E+11	0	1,2115E+11	2,4231E+11	1,2115E+11	0
	Numérica	0	2,4231E+11	2,4231E+11	0	1,2115E+11	2,4231E+11	1,2115E+11	0
	Diferencia	0	0	0	0	0	0	0	0
σ_{zz}	Analítica	0	2,4231E+11	2,4231E+11	0	1,2115E+11	2,4231E+11	1,2115E+11	0
	Numérica	0	2,4231E+11	2,4231E+11	0	1,2115E+11	2,4231E+11	1,2115E+11	0
	Diferencia	0	0	0	0	0	0	0	0
σ_{xy}	Analítica	0	0	0	0	0	0	0	0
	Numérica	0	0	0	0	0	0	0	0
	Diferencia	0	0	0	0	0	0	0	0
σ_{xz}	Analítica	0	0	0	0	0	0	0	0
	Numérica	0	0	0	0	0	0	0	0

	Diferencia	0	0	0	0	0	0	0	0
σ_{yz}	Analítica	0	0	0	0	0	0	0	0
	Numérica	0	0	0	0	0	0	0	0
	Diferencia	0	0	0	0	0	0	0	0

Tabla 5. Comparación de resultados del problema 6.4

Nótese que los errores relativos son cero. Esto se debe a que los valores de los desplazamientos, las funciones de forma y sus derivadas, el jacobiano, etc. en los nodos de este problema son redondos y no acumulan errores de redondeo.

6.5 Flexión respecto al eje y ($A = 1$)

En este ensayo, todos los grados de libertad adquieren un valor nulo, excepto el grado de libertad A (flexión unitaria según el eje y), que adquiere un valor unidad. El campo de desplazamientos es el siguiente:

$$\begin{cases} u_x = -\frac{z^2}{2}A \\ u_y = 0 \\ u_z = xzA \end{cases} \Longrightarrow \varepsilon_{ij} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & xA \end{pmatrix} \Longrightarrow \sigma_{ij} = \begin{pmatrix} 1.2115 \cdot 10^{11}Ax & 0 & 0 \\ 0 & 1.2115 \cdot 10^{11}Ax & 0 \\ 0 & 0 & 2.8269 \cdot 10^{11}Ax \end{pmatrix} \quad (26)$$

Se calcula la tensión σ_z de forma analítica y numérica en los puntos de Gauss del elemento, y se muestra la comparación de dichos resultados en la Tabla 6.

Punto de integración	σ_z analítica	σ_z numérica	Diferencia
1	3,186E+10	3,186E+10	0
2	1,4135E+11	1,4135E+11	0
3	2,5083E+11	2,5083E+11	0
4	3,186E+10	3,186E+10	0
5	1,4135E+11	1,4135E+11	0
6	2,5083E+11	2,5083E+11	0
7	3,186E+10	3,186E+10	0
8	1,4135E+11	1,4135E+11	0
9	2,5083E+11	2,5083E+11	0

Tabla 6. Comparación de resultados del problema 6.5

6.6 Flexión respecto al eje x ($B = 1$)

Se realiza el mismo ensayo que en el problema 6.5, pero esta vez todos los grados de libertad adquieren un valor nulo excepto B.

$$\begin{cases} u_x = 0 \\ u_y = -\dfrac{z^2}{2}B \\ u_z = yzB \end{cases} \Longrightarrow \varepsilon_{ij} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & yB \end{pmatrix} \Longrightarrow \sigma_{ij} = \begin{pmatrix} 1.2115 \cdot 10^{11}By & 0 & 0 \\ 0 & 1.2115 \cdot 10^{11}By & 0 \\ 0 & 0 & 2.8269 \cdot 10^{11}By \end{pmatrix} \quad (27)$$

Punto de integración	σ_z analítica	σ_z numérica	Diferencia
1	3,186E+10	3,186E+10	0
2	3,186E+10	3,186E+10	0
3	3,186E+10	3,186E+10	0
4	1,4135E+11	1,4135E+11	0
5	1,4135E+11	1,4135E+11	0
6	1,4135E+11	1,4135E+11	0
7	2,5083E+11	2,5083E+11	0
8	2,5083E+11	2,5083E+11	0
9	2,5083E+11	2,5083E+11	0

Tabla 7. Comparación de resultados analíticos y numéricos del problema 6.6

6.7 Alargamiento unitario ($C = 1$)

En este ensayo, se impone un campo de desplazamientos similar al de los problemas 6.5 y 6.6, pero en este caso el grado de libertad distinto de cero es C (alargamiento unitario según la dirección longitudinal z):

$$\begin{cases} u_x = 0 \\ u_y = 0 \\ u_z = zC \end{cases} \Longrightarrow \varepsilon_{ij} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & C \end{pmatrix} \Longrightarrow \sigma_{ij} = \begin{pmatrix} 1.2115 \cdot 10^{11}C & 0 & 0 \\ 0 & 1.2115 \cdot 10^{11}C & 0 \\ 0 & 0 & 2.8269 \cdot 10^{11}C \end{pmatrix} \quad (28)$$

Punto de integración	σ_z analítica	σ_z numérica	Diferencia
1	2,8269E+11	2,8269E+11	0
2	2,8269E+11	2,8269E+11	0
3	2,8269E+11	2,8269E+11	0
4	2,8269E+11	2,8269E+11	0
5	2,8269E+11	2,8269E+11	0

6	2,8269E+11	2,8269E+11	0
7	2,8269E+11	2,8269E+11	0
8	2,8269E+11	2,8269E+11	0
9	2,8269E+11	2,8269E+11	0

Tabla 8. Comparación de resultados analíticos y numéricos del problema 6.7

6.8 Torsión unitaria ($D = 1$)

Se impone el siguiente campo de desplazamientos en los nodos:

$$\begin{cases} u_x = yzD \\ u_y = -xzD \\ u_z = 0 \end{cases} \Rightarrow \varepsilon_{ij} = \begin{pmatrix} 0 & 0 & 2\frac{yD}{2} \\ 0 & 0 & -2\frac{xD}{2} \\ 2\frac{yD}{2} & -2\frac{xD}{2} & 0 \end{pmatrix} \Rightarrow \sigma_{ij} = \begin{pmatrix} 0 & 0 & 8.0769 \cdot 10^{10} yD \\ 0 & 0 & -8.0769 \cdot 10^{10} xD \\ 8.0769 \cdot 10^{10} yD & -8.0769 \cdot 10^{10} xD & 0 \end{pmatrix} \quad (29)$$

Punto de integración	σ_{xz} analítica	σ_{xz} numérica	Diferencia	σ_{yz} analítica	σ_{yz} numérica	Diferencia
1	9,1028E+09	9,1028E+09	0	-9,1028E+09	-9,1028E+09	0
2	9,1028E+09	9,1028E+09	0	-4,0385E+10	-4,0385E+10	0
3	9,1028E+09	9,1028E+09	0	-7,1666E+10	-7,1666E+10	0
4	4,0385E+10	4,0385E+10	0	-9,1028E+09	-9,1028E+09	0
5	4,0385E+10	4,0385E+10	0	-4,0385E+10	-4,0385E+10	0
6	4,0385E+10	4,0385E+10	0	-7,1666E+10	-7,1666E+10	0
7	7,1666E+10	7,1666E+10	0	-9,1028E+09	-9,1028E+09	0
8	7,1666E+10	7,1666E+10	0	-4,0385E+10	-4,0385E+10	0
9	7,1666E+10	7,1666E+10	0	-7,1666E+10	-7,1666E+10	0

Tabla 9. Comparación de resultados analíticos y numéricos del problema 6.8

6.9 Test de la parcela 1

En este problema de ensayo, se tiene un cuadrado de dimensiones $1m \times 1m$ formado por 5 elementos, como se indica en la Figura 8. A este cuadrado se le somete a una tracción igual que al elemento del problema 6.1, por lo que se deberán obtener los mismos desplazamientos en los nodos del contorno.

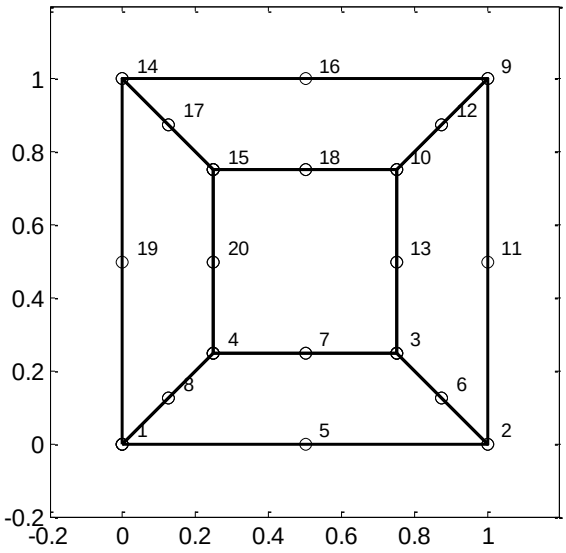

Figura 8. Malla de nodos y elementos del problema 6.9

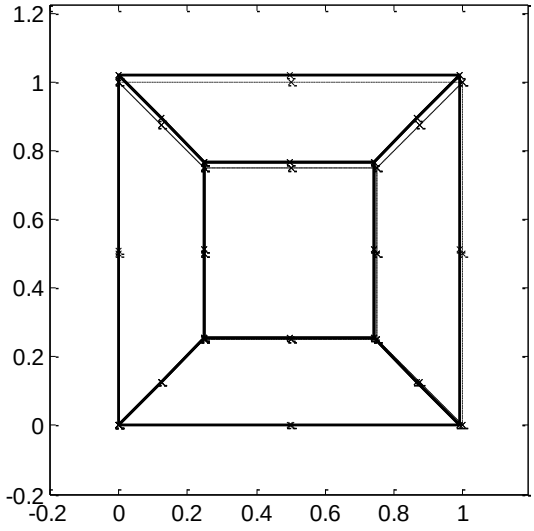

Figura 7. Situación deformada del problema 6.9

Problema 6.1			Problema 6.9		
Nodo	u_x	u_y	Nodo	u_x	u_y
1	0	0	1	0	0
2	-0,00928571	0	2	-0,00928571	0
3	-0,00928571	0,02166667	9	-0,00928571	0,02166667
4	0	0,02166667	14	0	0,02166667
5	-0,00464286	0	5	-0,00464286	0
6	-0,00928571	0,01083333	11	-0,00928571	0,01083333
7	-0,00464286	0,02166667	16	-0,00464286	0,02166667
8	-6,26E-18	0,01083333	19	-9,48E-18	0,01083333

Tabla 10. Comparación de resultados en desplazamientos de los nodos del contorno de los problemas 6.1 y 6.9

6.10 Test de la parcela 2

En este caso, se tiene un problema muy similar al anterior, pero con algunas diferencias en las coordenadas de los nodos. Este test de la parcela está inspirado en el que se propone en el libro de Zienkiewicz [3]. Las coordenadas de los nodos son:

Nodos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
x	0	2	2	0	0,4	1,4	1,5	0,3	1	1,7	0,9	0,2	2	1,75	1,45	1	0,15	0,9	0	0,35
y	0	0	3	2	0,4	0,6	2	1,6	0	0,3	0,5	0,2	1,5	2,5	1,3	2,5	1,8	1,8	1	1

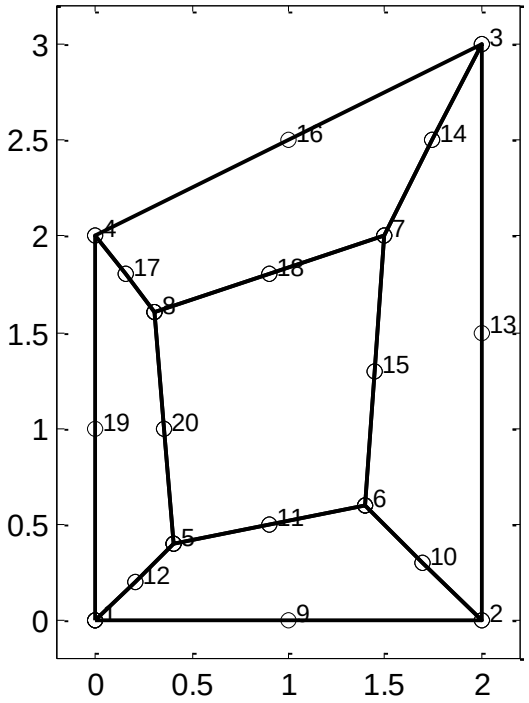

Figura 10. Malla de nodos y elementos del problema 6.10

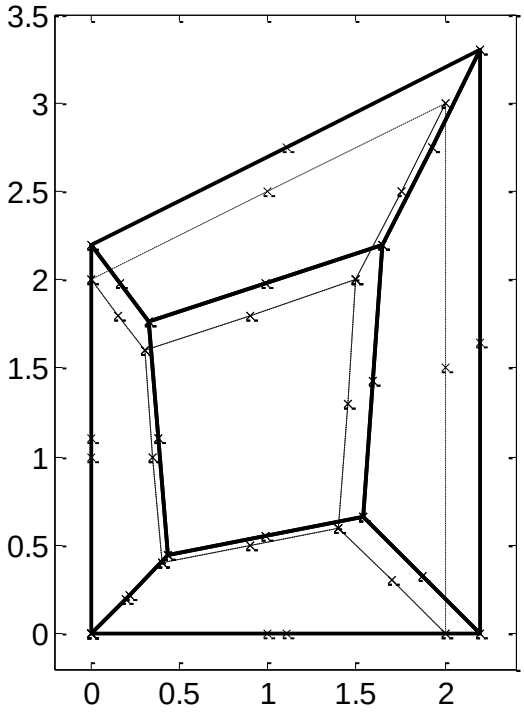

Figura 9. Situación deformada e indeformada del problema 6.10

Esta malla de nodos y elementos es sometida al mismo campo de desplazamientos que el problema 6.2. Posteriormente, se comparan los resultados en tensiones de la solución analítica y numérica en cada punto de Gauss, y se calcula la diferencia entre ellas, que se muestra en la Tabla 11:

Elemento	Punto de Gauss	σ_x analítica	σ_x numérica	Diferencia
1	1	4,0385E+11	4,0385E+11	-6,10E-05
1	2	4,0385E+11	4,0385E+11	0,00012207
1	3	4,0385E+11	4,0385E+11	-6,10E-05
1	4	4,0385E+11	4,0385E+11	0
1	5	4,0385E+11	4,0385E+11	0
1	6	4,0385E+11	4,0385E+11	-0,00024414
1	7	4,0385E+11	4,0385E+11	0
1	8	4,0385E+11	4,0385E+11	0
1	9	4,0385E+11	4,0385E+11	-0,00024414
2	1	4,0385E+11	4,0385E+11	-0,00036621

2	2	4,0385E+11	4,0385E+11	0,00054932
2	3	4,0385E+11	4,0385E+11	0,00146484
2	4	4,0385E+11	4,0385E+11	0,00231934
2	5	4,0385E+11	4,0385E+11	-0,00085449
2	6	4,0385E+11	4,0385E+11	-0,00286865
2	7	4,0385E+11	4,0385E+11	0,00128174
2	8	4,0385E+11	4,0385E+11	0,00079346
2	9	4,0385E+11	4,0385E+11	-0,00109863
3	1	4,0385E+11	4,0385E+11	-0,00061035
3	2	4,0385E+11	4,0385E+11	-0,00158691
3	3	4,0385E+11	4,0385E+11	0,00024414
3	4	4,0385E+11	4,0385E+11	-0,00042725
3	5	4,0385E+11	4,0385E+11	-0,00073242
3	6	4,0385E+11	4,0385E+11	-0,00018311
3	7	4,0385E+11	4,0385E+11	-0,00018311
3	8	4,0385E+11	4,0385E+11	0,00018311
3	9	4,0385E+11	4,0385E+11	0
4	1	4,0385E+11	4,0385E+11	0
4	2	4,0385E+11	4,0385E+11	0
4	3	4,0385E+11	4,0385E+11	0,00012207
4	4	4,0385E+11	4,0385E+11	6,10E-05
4	5	4,0385E+11	4,0385E+11	0
4	6	4,0385E+11	4,0385E+11	6,10E-05
4	7	4,0385E+11	4,0385E+11	-0,00036621
4	8	4,0385E+11	4,0385E+11	0
4	9	4,0385E+11	4,0385E+11	0,00012207
5	1	4,0385E+11	4,0385E+11	-6,10E-05
5	2	4,0385E+11	4,0385E+11	0,00018311
5	3	4,0385E+11	4,0385E+11	-0,00012207
5	4	4,0385E+11	4,0385E+11	0,00012207
5	5	4,0385E+11	4,0385E+11	-6,10E-05
5	6	4,0385E+11	4,0385E+11	0

5	7	4,0385E+11	4,0385E+11	-6,10E-05
5	8	4,0385E+11	4,0385E+11	0,00024414
5	9	4,0385E+11	4,0385E+11	0,00048828

Tabla 11. Comparación de resultados analíticos y numéricos del problema 6.10

Se puede apreciar que las diferencias en el cálculo numérico de las tensiones con respecto a los cálculos analíticos son muy pequeñas en comparación al valor de estas. Se ha elegido mostrar solo los valores de σ_x por ser ésta la mayor tensión y para simplificar la tabla.

6.11 Test de la parcela 3 (ensayo de cortadura)

En este problema, la misma malla utilizada en el problema 6.10 es sometida al campo de desplazamientos del problema 6.3. De nuevo, se comparan los resultados en tensiones analítico y numérico en los puntos de integración y se muestra la diferencia entre ellos. De esta forma se pone en evidencia que la malla de elementos realiza el cálculo numérico de una forma fiable y precisa.

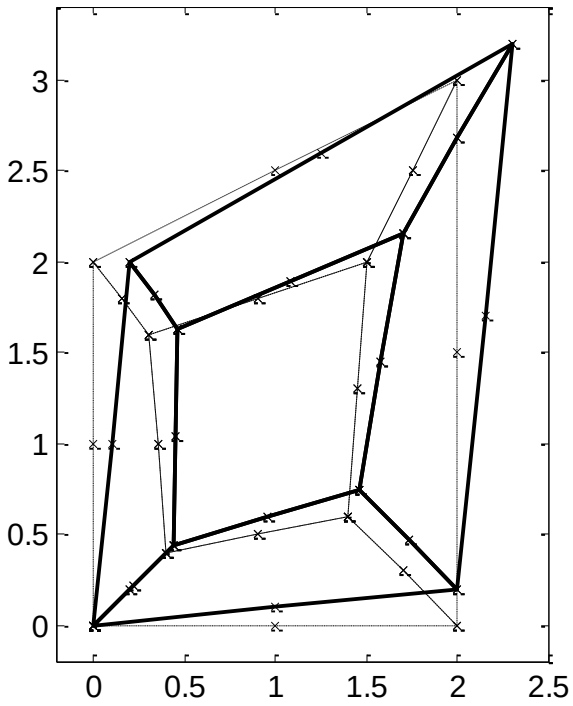

Figura 12. Situación deformada e indeformada del problema 6.11

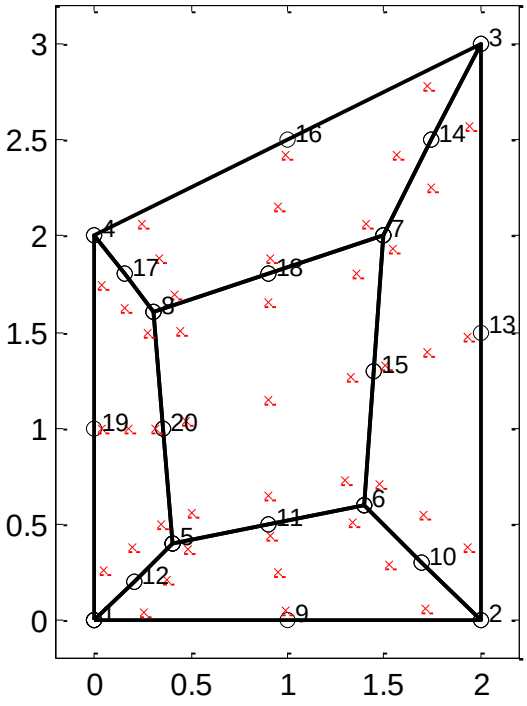

Figura 11. Puntos de Gauss

Elemento	Punto de Gauss	σ_{xy} analítica	σ_{xy} numérica	Diferencia
1	1	1,6154E+11	1,6154E+11	0
1	2	1,6154E+11	1,6154E+11	0
1	3	1,6154E+11	1,6154E+11	0,00012207
1	4	1,6154E+11	1,6154E+11	-3,05E-05

1	5	1,6154E+11	1,6154E+11	0
1	6	1,6154E+11	1,6154E+11	3,05E-05
1	7	1,6154E+11	1,6154E+11	3,05E-05
1	8	1,6154E+11	1,6154E+11	6,10E-05
1	9	1,6154E+11	1,6154E+11	-0,00018311
2	1	1,6154E+11	1,6154E+11	-0,00015259
2	2	1,6154E+11	1,6154E+11	0,00033569
2	3	1,6154E+11	1,6154E+11	0,00015259
2	4	1,6154E+11	1,6154E+11	0,00112915
2	5	1,6154E+11	1,6154E+11	-0,00012207
2	6	1,6154E+11	1,6154E+11	-0,00045776
2	7	1,6154E+11	1,6154E+11	0,00036621
2	8	1,6154E+11	1,6154E+11	-0,00021362
2	9	1,6154E+11	1,6154E+11	-0,00079346
3	1	1,6154E+11	1,6154E+11	-0,00036621
3	2	1,6154E+11	1,6154E+11	-0,00021362
3	3	1,6154E+11	1,6154E+11	-0,00079346
3	4	1,6154E+11	1,6154E+11	-0,00033569
3	5	1,6154E+11	1,6154E+11	-0,00045776
3	6	1,6154E+11	1,6154E+11	-0,00048828
3	7	1,6154E+11	1,6154E+11	-6,10E-05
3	8	1,6154E+11	1,6154E+11	-6,10E-05
3	9	1,6154E+11	1,6154E+11	0,00021362
4	1	1,6154E+11	1,6154E+11	-3,05E-05
4	2	1,6154E+11	1,6154E+11	3,05E-05
4	3	1,6154E+11	1,6154E+11	0
4	4	1,6154E+11	1,6154E+11	9,16E-05
4	5	1,6154E+11	1,6154E+11	6,10E-05
4	6	1,6154E+11	1,6154E+11	3,05E-05
4	7	1,6154E+11	1,6154E+11	-6,10E-05
4	8	1,6154E+11	1,6154E+11	-6,10E-05
4	9	1,6154E+11	1,6154E+11	0,00018311

5	1	1,6154E+11	1,6154E+11	0
5	2	1,6154E+11	1,6154E+11	0,00012207
5	3	1,6154E+11	1,6154E+11	-6,10E-05
5	4	1,6154E+11	1,6154E+11	-3,05E-05
5	5	1,6154E+11	1,6154E+11	-3,05E-05
5	6	1,6154E+11	1,6154E+11	0,00018311
5	7	1,6154E+11	1,6154E+11	-3,05E-05
5	8	1,6154E+11	1,6154E+11	-3,05E-05
5	9	1,6154E+11	1,6154E+11	0,00021362

Tabla 12. Comparación de resultados del test de la parcela del problema 6.11

6.12 Test de la parcela 4 (campo de desplazamientos parabólico)

En este problema, la misma malla utilizada en el problema 6.10 es sometida al campo de desplazamientos del problema 6.4. De nuevo, se comparan los resultados en tensiones analítico y numérico en los puntos de Gauss y se muestra la diferencia.

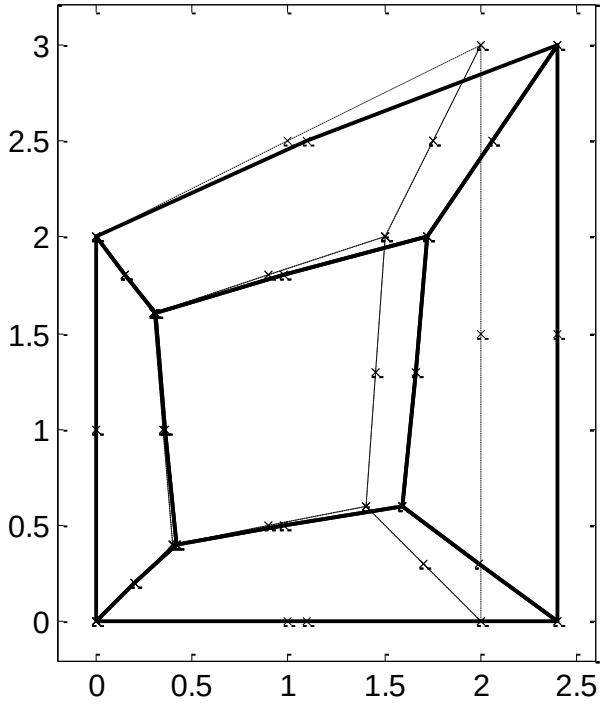

Figura 13. Situación deformada e indeformada del problema 6.12

Elemento	Punto de Gauss	σ_x analítica	σ_x numérica	Diferencia	Error relativo
1	1	1,4575E+11	1,3467E+11	1,1073E+10	0,01008333
1	2	5,5901E+11	5,6032E+11	-1304497524	-0,00118786
1	3	9,7228E+11	9,8421E+11	-1,1935E+10	-0,0108679
1	4	2,0866E+11	1,7043E+11	3,8228E+10	0,0348102
1	5	5,3712E+11	5,3712E+11	6,10E-05	5,56E-17
1	6	8,6557E+11	9,0013E+11	-3,4552E+10	-0,0314623
1	7	2,7157E+11	2,4022E+11	3,1343E+10	0,02854072
1	8	5,1522E+11	4,983E+11	1,6922E+10	0,01540871
1	9	7,5887E+11	7,7088E+11	-1,2015E+10	-0,01094043
2	1	8,3542E+11	8,3497E+11	452599768	0,00041213
2	2	9,6434E+11	9,6417E+11	174552759	0,00015895
2	3	1,0933E+12	1,0936E+12	-315681947	-0,00028746
2	4	8,5485E+11	8,5384E+11	1016074801	0,00092523
2	5	9,7529E+11	9,7529E+11	-0,00012207	-1,11E-16
2	6	1,0957E+12	1,0967E+12	-996778397	-0,00090765
2	7	8,7428E+11	8,7357E+11	714776970	0,00065087
2	8	9,8624E+11	9,8582E+11	414017976	0,000377
2	9	1,0982E+12	1,0985E+12	-305746621	-0,00027841
3	1	7,9773E+11	8,1713E+11	-1,9401E+10	-0,01766658
3	2	8,8747E+11	9,1458E+11	-2,7111E+10	-0,02468688
3	3	9,7721E+11	9,7721E+11	-0,00024414	-2,22E-16
3	4	5,1522E+11	5,3435E+11	-1,9132E+10	-0,01742103
3	5	5,3712E+11	5,3712E+11	-6,10E-05	-5,56E-17
3	6	5,5901E+11	5,3275E+11	2,6267E+10	0,02391808
3	7	2,3271E+11	2,3271E+11	0	0
3	8	1,8676E+11	1,6846E+11	1,8304E+10	0,01666723
3	9	1,4081E+11	1,2279E+11	1,8021E+10	0,01640966
4	1	2,477E+10	2,43E+10	469707104	0,00042771
4	2	1,0989E+11	1,1016E+11	-266080183	-0,00024229
4	3	1,9501E+11	1,9567E+11	-662278539	-0,00060306

4	4	2,2302E+10	2,0738E+10	1564089428	0,00142424
4	5	9,8942E+10	9,8942E+10	0	0
4	6	1,7558E+11	1,7715E+11	-1564089428	-0,00142424
4	7	1,9834E+10	1,9241E+10	592835188	0,00053983
4	8	8,7994E+10	8,8345E+10	-351505093	-0,00032008
4	9	1,5615E+11	1,5709E+11	-936538437	-0,0008528
5	1	2,8494E+11	2,8425E+11	691127671	0,00062933
5	2	5,0885E+11	5,0921E+11	-366584882	-0,00033381
5	3	7,3275E+11	7,3376E+11	-1007298339	-0,00091723
5	4	2,6798E+11	2,6602E+11	1956845201	0,00178188
5	5	5,0885E+11	5,0885E+11	6,10E-05	5,56E-17
5	6	7,4972E+11	7,5174E+11	-2021664461	-0,0018409
5	7	2,5101E+11	2,5006E+11	954174209	0,00086886
5	8	5,0885E+11	5,0831E+11	539971424	0,00049169
5	9	7,6668E+11	7,6723E+11	-549952385	-0,00050078

La exactitud de los cálculos en este problema disminuye debido a la distrosión de la malla. Al ser esta malla más irregular, el campo de desplazamientos parabólico no se adapta tan bien a las funciones de forma parabólicas del espacio isoparamétrico. Aun así, el máximo error relativo cometido, tomando como referencia la máxima tensión σ_x analítica (1098191204282,88 Pa) es del 3.48 %, lo cual es bastante aceptable.

7 CONCLUSIONES Y DESARROLLOS FUTUROS

Una vez validados los resultados en el capítulo 6, se puede decir que la formulación del elemento está correctamente implementada y funciona. Como ya se ha visto, todo el código ha sido escrito en MATLAB ® por la facilidad que supone la depuración e identificación de errores en esta herramienta. Sin embargo, MATLAB ® no es eficiente a la hora de definir las condiciones de contorno, cargas, geometría, malla de nodos y elementos del problema. Por tanto, el siguiente paso en el futuro desarrollo de este proyecto sería realizar una traducción del código a lenguaje de programación Fortran para su implementación en un código comercial de elementos finitos como ABAQUS ®.

REFERENCIAS

[1] A. Blázquez, V. Mantič and F. París, Application of BEM to generalized plane problems for anisotropic elastic materials in presence of contact, Engineering Analysis with Boundary Elements, vol. 30, pp. 489-502, 2006.

[2] J. Reinoso, Elementos Finitos multi-dimensionales: formulaciones 2D y 3D para sólidos elásticos lineales, Sevilla: Universidad de Sevilla, 2016.

[3] O. C. Zienkiewicz, El método de los elementos finitos, Vol. 1, Madrid: McGraw-Hill, 1994.

ANEXO 1

En este anexo se presenta la función `readInp`, que lee el .inp de Abaqus que contiene toda la información de la malla de elementos y devuelve las matrices `nodes` y `elements`.

```
% LECTURA INP DE ABAQUS
function [nodes,elements] = readInp(archivoLectura)
 format short g
% Extraer coordenadas de un archivo inp de Abaqus (sin part y
assembly)
% Devuelve un cell cuyos elementos son las coordenadas de cada pieza
% en el sistema global

texto = fileread(archivoLectura); % Leer inp

% Parte el texto en numero de piezas + 1 trozos.
% El primer trozo es el heading
nodes = strsplit(texto,'*Node');
nodes = nodes(2:end); % Quitar el heading

% Coger el trozo entre *Node y *Element
nodes = cellfun(@(x) strsplit(x,'*Element'), nodes, 'UniformOutput',
0);

% Hay que quedarse con el primer trozo de cada celda excepto
% de la primera
nodes = cellfun(@(x) x{1}, nodes, 'UniformOutput', 0);

% Convertir el string a número
nodes = cellfun(@str2num, nodes, 'UniformOutput', 0);
nodes=nodes{1,1};
nodes=[nodes,zeros(length(nodes(:,1)),1)]; %convert cell into matrix

% OBTENER CONECTIVIDAD ELEMENTOS
elements = strsplit(texto,'type=CPS8'); %Esto puede variar según el
elemento que se haya usado en Abaqus
elements = elements(2:end); % Quitar el heading

% Coger el trozo entre 'type=CPS8' y '*Nset'
elements = cellfun(@(x) strsplit(x,'*Nset'), elements,
'UniformOutput', 0);

% Hay que quedarse con el primer trozo de cada celda excepto
% de la primera
elements = cellfun(@(x) x{1}, elements, 'UniformOutput', 0);

% Convertir el string a número
elements = cellfun(@str2num, elements, 'UniformOutput', 0);
elements=elements{1,1}; % convert cell into matrix
```

ANEXO 2

En este anexo se adjunta la función `StiffnessMatrixISO`, que calcula la matriz de rigidez de un elemento. Esta función emplea otras funciones dentro de ella, como `kgauss`, `kshapefunctions2`, `kJacobian2D` y `kBmat2DLin`, que se adjuntan en el Anexo 3.

```
function [Kelement]=StiffnessMatrixISO(nodes,elements,iel)

nnode=8; %number of nodes
ndim =2; %dimension
ndof=3; %number of degrees of freedom
h=1.0; % espesor del elemento
density=0;
fy=zeros(1,nnode); % Vector de fuerzas asociado al peso propio
ngauss=3; %number of integration points per direction
Kelement=zeros(28);

nodes_coord=zeros(2,8);
jj=1;
for ii=elements(iel,2:9)
    nodes_coord(1,jj)=nodes(ii,2);
    nodes_coord(2,jj)=nodes(ii,3);
    jj=jj+1;
end
%------- position and weights  ------------------------
[posgp,weightgp]= kgauss(ngauss); %call to the function to calculate
the position and weights of gauss points

%----------  LOOP OVER THE INTEGRATION POINTS
      kip = 0;                                %counter

      for i = 1:ngauss                        %loop over xi_1
        xi =  posgp(i);
        wxi =  weightgp(i);

        for  j = 1:ngauss                      %loop over xi2
            eta =  posgp(j);
            weta =  weightgp(j);


                kip = kip + 1; %Integration point counter

         %-------------------------------------------------
         %Evaluate shape functions and derivatives
        [shapef,dshape] =kshapefunctions2(xi,eta,ndim,nnode);
         %-------------------------------------------------

      %-------------------------------------------------
      % Compute Jacobian and Inverse

[xjac,xjacinv,detxjac]=kJacobian2D(ndim,nnode,shapef,dshape,nodes_coor
d);
      %-------------------------------------------------
```

```
      factor=wxi*weta*h*detxjac; % h = espesor

      %-----------------------------------------------
      % Compute B-operator
      [BmatLin] =
kBmat2DLin(ndim,nnode,ngauss,shapef,dshape,xjac,xjacinv,xi,eta,nodes_c
oord);
      %-----------------------------------------------

            %-------- CmatVoigt -------------
            E=210e9; %acero
            nu=0.3; %acero
            CmatVoigt=inv([1/E -nu/E -nu/E 0 0 0
                        -nu/E 1/E -nu/E 0 0 0
                        -nu/E -nu/E 1/E 0 0 0
                        0 0 0 2*(1+nu)/E 0 0
                        0 0 0 0 2*(1+nu)/E 0
                        0 0 0 0 0 2*(1+nu)/E]);
            %---------------------------------
            factor=wxi*weta*h*detxjac; %repetido
            Kloop=BmatLin'*CmatVoigt*BmatLin*factor; % aportación a la
matriz de rigidez del punto de integración "kip"

          %    coord.reales*()*coord.reales*(pesos puntos de gauss en
NATURALES y jacobiano (reales))

            Kelement=Kelement+Kloop; % acumulación de la matriz de
rigidez del elemento

            fy=fy+shapef*density*factor; %Vector de fuerzas asociadas
al peso propio

        end %end loop over xi^1
      end %end loop over xi^2
```

ANEXO 3

Este anexo muestra las funciones auxiliares que se emplean en la función principal `StiffnessMatrixISO`.

La función `kgauss` calcula las posiciones y los pesos de los puntos de Gauss:

```
function [posgp,weightgp]= kgauss(ngauss)

 if (ngauss == 3)

        posgp(1) = -0.774596669241483;
        weightgp(1) = 5/9;

        posgp(2) = 0;
        weightgp(2) = 8/9;

        posgp(3) = 0.774596669241483;
        weightgp(3) = 5/9;
 end
```

La función `kshapefunctions2` evalúa las funciones de forma y sus derivadas en el punto que se le indique:

```
% FUNCIONES DE FORMA CORRECTAS

function [shapef,dshape] =kshapefunctions2(X,Y,ndim,nnode)

%Initialization
for i = 1:nnode
    shapef(i) = 0.0;
    for j = 1:ndim
        dshape(j,i) = 0.0;
    end
end

    shapef(1) = -1/4*(1-X)*(1-Y)*(1+X+Y);

    dshape(1,1) = -1/4*(-1+Y)*(2*X+Y);
    dshape(2,1) = -1/4*(-1+X)*(X+2*Y);

    shapef(2) = -1/4*(1+X)*(1-Y)*(1-X+Y);

    dshape(1,2) = 1/4*(-1+Y)*(-2*X+Y);
    dshape(2,2) = 1/4*(1+X)*(-X+2*Y);

    shapef(3) = -1/4*(1+X)*(1+Y)*(1-X-Y);

    dshape(1,3) =  1/4*(1+Y)*(2*X+Y);
    dshape(2,3) =  1/4*(1+X)*(X+2*Y);

    shapef(4) = -1/4*(1-X)*(1+Y)*(1+X-Y);

    dshape(1,4) = -1/4*(1+Y)*(-2*X+Y);
    dshape(2,4) = -1/4*(-1+X)*(-X+2*Y);
```

```
    shapef(5) = 1/2*(1-X)*(1+X)*(1-Y);

    dshape(1,5) = X*(-1+Y);
    dshape(2,5) = 1/2*(1+X)*(-1+X);

    shapef(6) = 1/2*(1+X)*(1+Y)*(1-Y);

    dshape(1,6) = -1/2*(1+Y)*(-1+Y);
    dshape(2,6) = -Y*(1+X);

    shapef(7) = 1/2*(1-X)*(1+X)*(1+Y);

    dshape(1,7) = -X*(1+Y);
    dshape(2,7) = -1/2*(1+X)*(-1+X);

    shapef(8) = 1/2*(1-X)*(1+Y)*(1-Y);

    dshape(1,8) = 1/2*(1+Y)*(-1+Y);
    dshape(2,8) = Y*(-1+X);
end
```

La función `kJacobian2D` calcula el jacobiano de las deformaciones:

```
%============================================================

function
[xjac,xjacinv,detxjac]=kJacobian2D(ndim,nnode,shapef,dshape,xref)

%initialization phase
       for i=1:ndim
        for j=1:ndim
       xjac(i,j) = 0.0;
       xjacinv(i,j) =0.0;
        end
       end

      % xref son las coordenadas de los nodos del elemento en el
espacio real
      for i=1:nnode
      xjac(1,1) = xjac(1,1) + dshape(1,i)*xref(1,i);
      xjac(1,2) = xjac(1,2) + dshape(1,i)*xref(2,i);

      xjac(2,1) = xjac(2,1) + dshape(2,i)*xref(1,i);
      xjac(2,2) = xjac(2,2) + dshape(2,i)*xref(2,i);
      end

      %COMPUTE DETERMINANT OF THE JACOBIAN
      detxjac=det(xjac);

      if(detxjac > 0.0)  %It can be formulated the inverse
        %COMPUTE THE INVERSE
        xjacinv = inv(xjac);
      else
      end
```

```
end
%=========================================================
```

La función `kBmat2DLin` calcula el operador de compatibilidad ***B***:

```
function [Bmat2] =
kBmat2DLin(ndim,nnode,ngauss,shapef,dshape,xjac,xjacinv,xi,eta,xref)

% xi y eta son coordenadas naturales, hay que transformarlas al
espacio
% real: x1_real=N1(xi,eta)*x1_nodo1+N2(xi,eta)*x1_nodo2...
x1_real=0;
x2_real=0;
for i=1:8
    x1_real=x1_real+shapef(i)*xref(1,i);
    x2_real=x2_real+shapef(i)*xref(2,i);
end

    %initialization
      Bmat=zeros(6,ndim*nnode);

       % Linear B-operator looks like that
       % 0   0   0 0 | Nk,x     0       0|
       % 0   0   0 0 |   0     Nk,y     0|
       % x1  x2  1 0 |   0      0       0|

       % 0 0 0 -x1 |   0       0   Nk,x|
       % 0 0 0  x2 |   0       0   Nk,y|
       % 0 0 0  0  | Nk,y    Nk,x     0|

       % dN     dN  dxi     dN  deta
       %---- = ----*---- + ----*----
       % dx     dxi  dx    deta  dx

     kint = 1;
     for i=1:nnode
      Bmat(1,kint)   = dshape(1,i)*xjacinv(1,1)+ ...
                       dshape(2,i)*xjacinv(1,2);

      Bmat(2,kint+1)= dshape(1,i)*xjacinv(2,1)+...
                       dshape(2,i)*xjacinv(2,2);

      % 0    0     Nk,2|
      Bmat(4,kint+2)    = Bmat(2,kint+1);

      %   0    0   Nk,1|
      Bmat(5,kint+2) = Bmat(1,kint);

      % Nk,2    Nk,1  0|
      Bmat(6,kint)    = Bmat(2,kint+1);
      Bmat(6,kint+1) = Bmat(1,kint);


       kint = kint + 3;
     end
```

```
    A=zeros(6,4);
    A(3,1)=x1_real;
    A(3,2)=x2_real;
    A(3,3)=1;
    A(4,4)=-x1_real;
    A(5,4)=x2_real;

Bmat2=[A Bmat];
%=========================================================
```