F. Zárate, E. Oñate
CIMNE 2006

Índice:

Introducción.

Como cualquier otro método numérico el Método de los Elementos Finitos se encuentra asociado a la programación de los algoritmos que le definen. De hecho, su ventaja radica en la iteración de los cálculos sobre una gran cantidad de elementos. Históricamente el primer lenguaje de programación utilizado para trabajar con el Método de los Elementos Finitos fue el FORTRAN; a partir de entonces muchas rutinas, algoritmos y programas asociados al método han sido programados en este lenguaje. Con el desarrollo de los ordenadores, nuevos lenguajes han aparecido, cada uno con capacidades y herramientas específicas para diversos campos de aplicación, todo ello con la idea de simplificar la codificación de los algoritmos y optimizar los recursos del ordenador.

Aunque FORTRAN continúa siendo un lenguaje de referencia en el uso del Método, los nuevos lenguajes y herramientas de programación permiten la simplificación en la escritura de los algoritmos al tiempo que hacen uso de bibliotecas de recursos específicos que optimizan los recursos de tiempo y memoria del ordenador. Éste es el caso concreto de MATLAB que además de ser una herramienta de investigación, nos permite escribir códigos que interpreta en el momento de la ejecución. Aunque desde el punto de vista de una programación optima, los lenguajes interpretados son muy lentos, MATLAB nos permite hacer uso de toda la librería matricial implementada lo que permite optimizar los cálculos hasta el punto de competir eficientemente con otros lenguajes compilados.

MATLAB es una herramienta diseñada para trabajar con matrices, facilitando todas las operaciones del álgebra matricial, desde el punto de vista numérico y de almacenamiento en memoria, haciendo totalmente transparente el uso de rutinas y algoritmos que en FORTRAN resultarían demasiado complejos de implementar.

El contar con un programa de cálculo eficiente no es el único requisito para poder trabajar de manera adecuada con el método de los Elementos Finitos. Es necesario contar con una interfaz adecuada para preparar los datos del problema, generar mallas adecuadas al tipo de problema a resolver y presentar los resultados de manera que la interpretación de estos sea el principal objetivo. MATLAB es un programa muy eficiente en el tratamiento de matrices pero muy pobre en sus capacidades graficas. El complemento ideal es el pre / postprocesador GiD.

GiD es una herramienta diseñada para tratar adecuadamente las entidades geométricas, realizar la asignación de las propiedades de los materiales y definir las condiciones de contorno del problema. Esfuerzos como la discretización y la escritura de datos en un formato establecido se convierte en una tarea prácticamente transparente para el usuario.

La interpretación de datos mediante GiD resulta una tarea sencilla, en donde la visualización de los resultados obtenidos permite concentrarse en la interpretación de los resultados.

MAT-fem ha sido escrito pensando en la interacción conjunta de GiD con MATLAB. GiD permite manipular las geometrías y discretizaciones, escribiendo el fichero de

entrada de datos correspondiente. A través de MATLAB se ejecuta el programa de cálculo, sin perder las ventajas de uso de MATLAB. Finalmente GiD recoge los ficheros de datos para su visualización grafica e interpretación.

Este esquema permite conocer en detalle la ejecución de un programa de Elementos Finitos, siguiendo, si se desea, paso a paso cada una de las líneas del código, al tiempo que es posible realzar ejemplos que por sus dimensiones caerían fuera de cualquier programa con fines educativos.

En los siguientes apartados se describe el programa en detalle, comenzado por el fichero de entrada de datos, que aunque se transcribe de forma automática por GiD, contiene información importante para entender el funcionamiento de MAT-fem.

Finalmente se describe la interfaz de usuario implementa en GiD para interactuar con MATLAB, presentándose un ejemplo de aplicación.

Fichero de datos.

Antes de describir el programa es obvio decir que de alguna manera es necesario alimentarlo con la información correspondiente a las coordenadas nodales, la discretización en elementos finitos y las condiciones de frontera. Por ello, se describe primeramente este fichero a fin de familiarizarnos con las variables utilizadas y la programación en MATLAB. Como comentamos MATLAB es un intérprete y utilizaremos esta propiedad para definir la lectura de datos. Es decir, que el fichero de entrada de datos sea en realidad una subrutina del programa en donde se asignan directamente a las variables los valores correspondientes al problema a resolver.

Ello nos evita el tener que definir una sintaxis de lectura especial para el programa y al tiempo nos evita también hacer uso de una interfaz de I/O, concentrándonos directamente en la programación del método.

Así pues, el fichero de entrada de datos utilizará la sintaxis de MATLAB y en el se definen directamente las variables utilizadas por el programa para describir el problema a resolver. No esta de más decir que el nombre del fichero de datos deberá llevar la extensión .m propia de los scripts de MATLAB

El fichero de datos podemos distinguir tres grupos de variables: las asociadas al material; las que definen la topología del problema y las que definen las condiciones de contorno asociadas.
Con el objeto de simplificar la codificación, se ha decidido utilizar un material elástico lineal isótropo único para todo el dominio, por lo que la sección de material solo aparece una vez en el fichero de datos.

La figura 1 muestra las variables asociadas al material: la variable `pstrs` indica si se trata de un problema de tensión plana (`pstrs = 1`) o uno de deformación plana (`pstrs = 0`). `young` contiene el modulo de elasticidad y `poiss` el coeficiente de Poisson. `thick` y `denss` definen el espesor del dominio y la densidad de éste. Es obvio que en un problema de Deformación Plana el valor del espesor debe ser unitario.

```
%
% Material Properties
%
 pstrs  = 1;
 young  = 1000.0;
 poiss  = 0.2;
 thick  = 0.1;
 denss  = 1.0;
```

Figura 1: Fichero de entrada de datos, definición del material.

Es importante observar en este momento que el programa se encuentra libre de mecanismos de validación de datos, por lo que no comprobara detalles como que el coeficiente de Poisson se encuentre en el rango de `0 <= poiss < 0.5` u otro tipo de detalles que ocultarían el algoritmo mismo. Se deja al usuario este tipo de comprobaciones que resultan obvias.

El grupo de variables que definen la topología del problema se definen con el atributo de variables globales para ser accesibles en el código y sus subrutinas. La figura 2 muestra la definición de las coordenadas mediante la variable `coordinates` y las conectividades nodales con la variable `elements`.

```
%
% Coordinates
%
global coordinates
coordinates = [
  0.00 , 0.00;
  0.50 , 0.00;

  2.00 , 1.00;
   2.50 , 1.00  ];
%
% Elements
%
global elements
elements = [
    1,   2,   7 ;
    2,   3,   8 ;

   17,  16,  11 ;
   18,  17,  12 ];
```

Figura 2: Fichero de entrada de datos, definición de la topología

`coordinates` es una matriz con tantas filas como numero de nodos y columnas como dimensiones del problema, es decir de dimensión (`npnod x 2`) y describe las coordenadas de todos los nodos utilizados en la discretización. El número de nodo corresponde a la posición que guardan sus coordenadas en la matriz definida.

La variable `elements` define el número de elementos y sus conectividades. Tantas filas como número de elementos y columnas como nodos tiene cada elemento (`nelem x nnode`). En el caso de definir elementos triangulares serán tres el número de columnas y cuatro en el caso de utilizar elementos cuadriláteros. De manera similar a la variable

anterior, el número de elemento se corresponde con el número de fila que guarda en la matriz definida.

El último grupo de variables definen las condiciones de contorno del problema, como se muestra en la figura 3.

```
%
% Fixed Nodes
%
fixnodes = [
    1, 1, 0.0 ;
    1, 2, 0.0 ;

   13, 1, 0.0 ;
   13, 2, 0.0 ];
%
% Point loads
%
pointload = [
              6, 2, -1.0 ;

             18, 2, -1.0 ];
%
% Side loads
%
sideload = [
             11,12, 2.0, 3.0;

             14,15, 2.0, 3.0 ];
```

Figura 3: Fichero de entrada de datos, definición de las condiciones de contorno

la variable `fixnodes` corresponde a los grados de libertad restringidos de acuerdo con el problema a resolver. `fixnodes` es una matriz donde el número de filas corresponde al número de grados de libertad prescritos, mientras que el número de columnas describen en el siguiente orden el nodo restringido, el grado de libertad vinculado (1 en x y 2 en y) y el valor del grado de libertad conocido. De ésta manera si un nodo se encuentra prescrito en sus dos direcciones se necesitarán dos renglones para definir dicha condición.

La variable `pointload` se usa para definir las cargas puntuales. Al igual que las variables anteriores, se trata de una matriz donde número de filas es el número de cargas definidas en el problema y el número de columnas corresponden al numero del nodo sobre cuál actuara carga, la dirección en la que actúa y el valor de la carga. Las cargas se encuentran referidas en el sistema global de coordenadas. En el caso de no existir cargas puntuales `pointload` se define como una matriz vacía mediante el comando `pointload = [ ] ;`

Finalmente se define la variable `sideload` que contiene la información sobre las cargas uniformemente repartidas a lo largo de los lados de los elementos. La definición de las cargas es en el sistema de ejes globales del problema. `sideload` es una matriz donde el numero de filas es el número de lados con carga y las dos primeras columnas definen los nodos de dicho lado mientras las columnas 3 y 4 corresponden al valor de la carga distribuida por unidad de longitud en dirección x e y respectivamente. De no existir este tipo de cargas, `sideload` debe ser definida como una matriz vacía mediante el comando `sideload = [ ] ;`

El nombre del fichero de datos es potestad del usuario, sin embargo, la extensión debe ser .m a fin de que MATLAB pueda reconocerlo como un script valido.

MAT-fem

MAT-fem es un programa de ejecución TOP-DOWN. El esquema de bloques del programa se muestra en la figura 4. La primera parte, la lectura de datos, se realiza en el fichero mismo de datos que es en realidad una subrutina del programa y que ha sido descrito en el apartado anterior.
A continuación se arma para todos los elementos, la matriz de rigidez y el vector de cargas nodales elemental por peso propio, procediendo directamente al ensamblaje de estos.

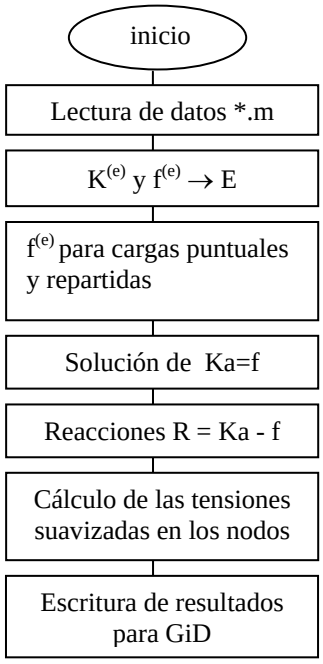

Figura 4: Diagrama de flujo del programa MAT-fem

Se termina de aplicar las condiciones de carga al vector de fuerzas nodales antes de eliminar las ecuaciones correspondientes a los grados de libertad conocidos y poder resolver el sistema de ecuaciones.

Conocidos los desplazamientos se calculan las reacciones y las tensiones del sistema finalizando con la escritura de datos para poder visualizarlos en GiD.

inicio

MAT-fem comienza limpiando todas las variables de la memoria con la instrucción `clear` seguidamente pregunta al usuario el nombre de el fichero de datos que se va a utilizar, sin incluir la extensión `*.m.` . La figura 5 muestra las primeras líneas de código del programa y que corresponden a la inicialización de variables así como la puesta a punto de reloj del programa, almacenando en la variable `ttim` el tiempo total de

ejecución.

```
%% MAT-fem
%
% Clear memory and variables.
  clear

  file_name = input('Enter the file name :','s');

  tic;                    % Start clock
  ttim = 0;               % Initialize time counter
  eval (file_name);       % Read input file

% Finds basics dimentions
  npnod  = size(coordinates,1);        % Number of nodes
  nndof  = 2*npnod;                    % Number of total DOF
  nelem  = size(elements,1);           % Number of elements
  nnode  = size(elements,2);           % Number of nodes per element
  neleq  = nnode*2;                    % Number of DOF per element

  ttim = timing('Time needed to read the input file',ttim);
```

Figura 5: Inicialización del programa y lectura de datos

La lectura de datos, como se comentado, no es otra cosa más que una asignación directa a las variables dentro del programa. A partir de dichas matrices es posible extraer las dimensiones básicas del problema tales como el número de puntos nodales `npnod` que corresponde al número de renglones que contiene la variable `coordinantes`. El número de grados de libertad total del programa `nndof` será dos veces la cantidad anterior (`2*npnod`). `nelem` corresponde al número de elementos leídos y es el número de renglones contenido en la matriz `elements` mientras que el número de nodos que contienen cada elementos es el número de columnas de dicha matriz; de esta manera se identifican los elementos triangulares si el número de columnas es tres mientras que para los elementos cuadriláteros el número de columnas será de cuatro.

El número total de ecuaciones por elemento será el número de nodos por elemento `nnode` multiplicado por el número de grados de libertad de cada nodo: dos en el caso de problemas bidimensionales.

Es interesante reflexionar que estas variables no son necesarias puesto que se encuentran definidas en la estructura de datos, sin embargo, su definición permite que la interpretación de las líneas del código sea muchísimo más sencilla.

A lo largo del programa se utiliza la rutina `timing` para calcular el tiempo de ejecución entre dos puntos del código de esta manera el usuario puede observar cuáles son las partes del programa que requieren más esfuerzo computacional. En dicha rutina se hace uso de las llamadas `tic` y `toc` de MATLAB.

Matriz de rigidez y ensamblaje

El código mostrado en la figura 6 se define a la matriz de rigidez global del sistema y al vector de fuerzas nodales equivalentes como matriz y vector `sparse` respectivamente.

MAT-fem utiliza matrices dispersas con el objeto de optimizar el uso de memoria haciendo uso de haciendo uso de las herramientas de MATLAB. De esta forma y sin esfuerzo adicional MAT-fem hace uso de algoritmos muy potentes sin perder la sencillez de escritura.

```
% Dimension the global matrices.
  StifMat = sparse ( nndof , nndof );  % The global stiffness matrix
  force   = sparse ( nndof , 1 );      % The global force vector

%  Material properties (Constant over the domain).
  dmat = constt(young,poiss,pstrs);
```

Figura 6: Inicialización de las matrices globales

Dado que este programa tiene la finalidad principal de demostrar la implementación del método se hace uso de distintas simplificaciones una de ellas corresponde a que el material a utilizar dentro del dominio será constante y único, de esta manera la matriz constitutiva no variará de elemento a elemento, por lo que se evalúa antes de iniciar calculo de la matriz de rigidez.

La subrutina `constt` tiene como entrada de datos el módulo de Young, el coeficiente de Poisson y la bandera indicativa que nos permite distinguir entre un problema de tensión plana y otro de deformación plana. La matriz constitutiva es almacenada en `dmat` como se observa en la figura 6. En la figura 7 se muestra la rutina `constt` en donde se arma explícitamente la matriz constitutiva de tensión o deformación plana para un material elástico lineal isótropo.

```
function D = constt (young,poiss,pstrs)

%  Plane Sress
   if (pstrs==1)
       aux1 = young/(1-poiss^2);
       aux2 = poiss*aux1;
       aux3 = young/2/(1+poiss);
%  Plane Strain
   else
       aux1 = young*(1-poiss)/(1+poiss)/(1-2*poiss);
       aux2 = aux1*poiss/(1-poiss);
       aux3 = young/2/(1+poiss);
       thick= 1.0;
  end

   D = [aux1,aux2,0;aux2,aux1,0;0,0,aux3];
```

Figura 7: Definición de la matriz constitutiva.

En este programa se ha decidido optar por recalcular valores en lugar de almacenarlos temporalmente. La velocidad de los ordenadores permite esta posibilidad sin menoscabo de en la eficiencia del programa. Además, permite disponer de más memoria para poder atacar problemas mayores.

La figura 8 muestra el ciclo elemental donde se calcula y ensambla las matrices de rigidez y el vector de cargas nodales equivalente para cada elemento. El ciclo comienza recuperando las propiedades geométricas de cada elemento. En el vector `lnods` se almacenan las conectividades nodales del elemento a tratar y en la matriz `coord` se guardan las coordenadas de dichos nodos.

```
%  Element cycle.
  for ielem = 1 : nelem

% Recover element properties
    lnods = elements(ielem,:);                          % conectivities
    coord(1:nnode,:) = coordinates(lnods(1:nnode),:); % coordinates

% Evaluates the elemental stiffnes matrix and mass force vector.
    if (nnode == 3)
      [ElemMat,ElemFor] = TrStif(coord,dmat ,thick,denss); % Triangle
    else
      [ElemMat,ElemFor] = QdStif(coord,dmat ,thick,denss); % Quad.
    end

% Finds the equation number list for the i-th element
    eqnum = [];                                     % Clear the list
    for i =1 : nnode                                % Node cicle
      eqnum = [eqnum,lnods(i)*2-1,lnods(i)*2];   % Build the equation
    end                                             % number list

% Assamble the force vector and the stiffnes matrix
    for i = 1 : neleq
      force(eqnum(i)) = force(eqnum(i)) + ElemFor(i);
      for j = 1 : neleq
         StifMat(eqnum(i),eqnum(j)) = StifMat(eqnum(i),eqnum(j)) + ...
                                      ElemMat(i,j);
      end
    end

  end  % End element cicle
```

Figura 8: Cálculo y ensamblaje de la matriz de rigidez elemental.

El siguiente paso es calcular la matriz de rigidez elemental. Dependiendo del elemento se llamara a la rutina `TrStif` para los elementos triangulares y a `QdStif` para los elementos cuadriláteros. Cabe hacer mención que la misma rutina evalúa la matriz de rigidez y el vector de cargas nodales equivalentes asociados al peso propio. El uso de la misma integración de área permite esta simplificación. El cálculo de las matrices de rigidez elementales requiere un estudio mas detallado y por lo tanto se realiza en el siguiente apartado.

Antes de realizar el ensamblaje se define el vector `eqnum` que contiene el número de ecuaciones en numeración global correspondiente a cada una de las ecuaciones de la matriz de rigidez elemental. La conversión es simple pues para cada nodo corresponden dos ecuaciones (una por cada grado de libertad).

El ensamblaje se realiza mediante dos ciclos de `1` a `neleq` (numero de ecuaciones en cada elemento) lo que permite ensamblar en el primer ciclo el vector de cargas nodales equivalente y dentro del segundo la matriz de rigidez elemental termino a termino. Este esquema evita almacenar temporalmente las matrices de rigidez elementales.

Detalles sobre la matriz de rigidez

En este programa se utilizan dos elementos uno triangular cuya matriz de rigidez se calcula de forma explícita y otro cuadrilátero cuya matriz de rigidez se calcula mediante integración numérica. Ambas rutinas requieren exactamente los mismos datos de

entrada y regresan también las mismas variables, estas son la matriz de rigidez `ElemMat` y el vector de cargas nodales equivalentes `ElemFor` para la condición de peso propio.

La rutina `TrStif` definida para elementos triangulares se muestra en la figura 9. En ella se calculan directamente las derivadas cartesianas de las funciones de forma del triángulo lineal de tres nodos. Dichas derivadas son constantes en todo el elemento por lo que su cálculo es trivial al igual que el cálculo del área del elemento. De esta manera la matriz `bmat` se forma simplemente colocando cada una de las derivadas cartesianas (`b(i)/area2` y `c(i)/area2`) en la posición correspondiente dentro de la matriz de deformaciones, sin olvidar que es factor común a las expresiones de las derivadas el termino uno sobre dos veces el área del elemento.

La matriz de rigidez se calcula entonces mediante la clásica expresión $\mathbf{B}^T\mathbf{D}\mathbf{B}\ dA$. En esta línea se observa una de las ventajas fundamentales de MATLAB donde la multiplicación de matrices es directa sin necesidad de escribir engorrosos ciclos de cálculo.

El vector de cargas nodales equivalentes se evalúa repartiendo equitativamente el peso del elemento en cada uno de los nodos (`force`), tomando en cuenta que dicha carga representa una fuerza negativa en dirección `y`.

```
function [M,F] = TrStif ( nodes,dmat,thick,denss)

  b(1) = nodes(2,2) - nodes(3,2);    % bi = yj - yk
  b(2) = nodes(3,2) - nodes(1,2);
  b(3) = nodes(1,2) - nodes(2,2);

  c(1) = nodes(3,1) - nodes(2,1);    % ci = xk - xj
  c(2) = nodes(1,1) - nodes(3,1);
  c(3) = nodes(2,1) - nodes(1,1);

  area2 = abs(b(1)*c(2) - b(2)*c(1));
  area = area2 / 2;

  bmat = [b(1),  0 ,b(2),  0 ,b(3),  0 ;      % Explicit form for matrix B
            0 ,c(1),  0 ,c(2),  0 ,c(3);
          c(1),b(1),c(2),b(2),c(3),b(3)];

  bmat = bmat / area2;

  M = (transpose(bmat)*dmat*bmat)*area*thick;  % Elemental stiffnes matrix

  force = area*denss*thick/3;
  F = [0,-force,0,-force,0,-force];   % Elemental force vector
```

Figura 9: Cálculo de la matriz de rigidez elemental del elemento triangular.

Para el caso de la matriz de rigidez del elemento cuadrilátero se necesita realizar una integración numérica, para ello se define primeramente las funciones de forma del elemento (`fform`) y sus derivadas en coordenadas naturales (`deriv`). Dicha definición se realiza mediante una función intrínseca. Esta facilidad permitida en MATLAB evita el uso de subrutinas adicionales.

En este nivel también se definen los valores de las coordenadas (`pospg`) y los pesos (`pespg`) correspondientes a la regla de integración de `2x2` de Gauss-Legendre, también

se realiza la inicialización de la matriz `M` de rigidez elemental y el vector de cargas nodales equivalentes `fy`. En la figura 10 se observa con detalle esta rutina.

Una vez inicializadas las variables se procede con dos bucles para realizar una integración en cada uno de los ejes del elemento. El vector `lcffm` contiene los valores de las funciones de forma evaluadas en el punto integración `i,j`. y la matriz `lcder` almacena los valores de las derivadas naturales de dichas funciones en el mismo punto integración. La matriz Jacobiana (`xcjacm`) se calcula mediante la simple multiplicación de la derivadas naturales de las funciones de forma por las coordenadas nodales del elemento. El valor de las derivadas cartesiana (`ctder`) en el mismo punto se obtiene al multiplicar la inversa de la matriz Jacobiana por las derivadas naturales de las funciones de forma. El diferencial de área no es otra cosa más que el determinante de la matriz Jacobiana por las funciones de peso del punto integración sin olvidar el espesor del elemento.

```
function [M,F] = QdStif ( nodes,dmat,thick,denss)

   fform = @(s,t)[(1-s-t+s*t)/4,(1+s-t-s*t)/4,(1+s+t+s*t)/4,(1-s+t-s*t)/4];
   deriv = @(s,t)[(-1+t)/4,( 1-t)/4,( 1+t)/4,(-1-t)/4 ;
                  (-1+s)/4,(-1-s)/4,( 1+s)/4,( 1-s)/4 ];

   pospg = [ -0.577350269189626E+00 , 0.577350269189626E+00 ];
   pespg = [  1.0E+00 , 1.0E+00];
   M = zeros(8,8);
   fy = zeros(1,4);

   for i=1 : 2
      for j=1 : 2
         lcffm = fform(pospg(i),pospg(j)) ;    % FF at gauss point
         lcder = deriv(pospg(i),pospg(j)) ;    % FF Local derivatives
         xjacm = lcder*nodes ;                 % Jacobian matrix
         ctder = xjacm\lcder ;                 % FF Cartesian derivates
         darea = det(xjacm)*pespg(i)*pespg(j)*thick;

         bmat = [];
         for inode = 1 : 4
           bmat = [ bmat , [ctder(1,inode),            0 ;
                                        0 ,ctder(2,inode);
                            ctder(2,inode),ctder(1,inode) ] ] ;
         end

         M = M + (transpose(bmat)*dmat*bmat)*darea;

         fy = fy + lcffm*denss*darea;

      end
   end

   F = [ 0, -fy(1), 0, -fy(2), 0, -fy(3), 0, -fy(4)];
```

Figura 10: Cálculo de la matriz de rigidez elemental del elemento cuadrilátero.

La matriz de deformaciones se obtiene colocando adecuadamente las derivadas cartesianas de dicha funciones de forma en un arreglo matricial, de manera que sea sencillo obtener mediante la clásica expresión la matriz de rigidez del elemento. No hay que olvidar que para realizar una integración numérica es necesario evaluar la sumatoria de la evaluación de las funciones a integrar en cada uno de los puntos de integración. Para ello se realiza la sumatoria en la matriz `M`

El vector de fuerzas nodales equivalentes también requiere una integración sobre el área del elemento pero en este caso del producto de las funciones de forma por la densidad del elemento.

Para finalizar la rutina se arregla convenientemente en el vector `F` las fuerzas nodales encontradas para cada uno de los nodos tomando en cuenta que sólo existe la componente negativa en dirección `y`.

Estas dos rutinas pretenden ejemplificar dos mecanismos a la par el cálculo de la matriz de rigidez el primero de forma explícita y el segundo mediante integración numérica. Resulta conveniente agrupar en la misma rutina las variables que son utilizadas de forma local y evitar la llamada a múltiples subrutinas pues ello sólo conlleva a complicar y dificultar la depuración del código.

Cargas exteriores.

Además las cargas de peso propio es necesario considerar cargas uniformemente distribuidas sobre los lados de los elementos y cargas puntuales en los nodos.

Dado que ambos elementos implementados tienen funciones de forma lineales, el cálculo de la contribución nodal de las cargas uniformemente repartidas será exactamente el mismo para ambos casos. El cálculo se realiza a continuación del ensamblaje de las matrices elementales, dentro de la rutina principal del programa MAT-fem. Dicho código se presenta en la figura 11 donde hay un ciclo por el número de cargas definidas por la variables `sideload`. Por cada carga se calcula la distancia entre ambos nodos y se reparte de manera equitativa el producto de la carga por la longitud del lado (ya que las funciones de forma utilizadas son lineales). La carga nodal evaluada se acumula en el vector de cargas globales equivalentes con ayuda de la posición encontrada en `ieqn`.

Dado que las cargas se encuentran definidas en el sistema de ejes global no es necesario realizar ninguna rotación o transformación de éstas.

```
%  Add side forces to the force vector
  for i = 1 : size(sideload,1)
     x=coordinates(sideload(i,1),:)-coordinates(sideload(i,2),:);
     l = sqrt(x*transpose(x));       % Finds the lenght of the side
     ieqn = sideload(i,1)*2;         % Finds eq. number for the first node
     force(ieqn-1) = force(ieqn-1) + l*sideload(i,3)/2;   % add x force
     force(ieqn  ) = force(ieqn  ) + l*sideload(i,4)/2;   % add y force

     ieqn = sideload(i,2)*2;         % Finds eq. number for the second node
     force(ieqn-1) = force(ieqn-1) + l*sideload(i,3)/2;   % add x force
     force(ieqn  ) = force(ieqn  ) + l*sideload(i,4)/2;   % add y force
  end
```

Figura 11: Cálculo de las fuerzas nodales equivalentes por carga uniformemente repartida.

Para el caso de las cargas concentradas el cálculo se realiza simplemente añadiendo el valor de la carga concentradas al vector global de fuerzas nodales equivalentes. Para

ello es necesario definir un ciclo con el número de cargas contenidas en la variable `poinload` y encontrar para cada una de ellas el número de ecuación asociado necesario para agregar el valor de la carga al vector `force`.

```
%  Add point loads conditions to the force vector
  for i = 1 : size(pointload,1)
    ieqn = (pointload(i,1)-1)*2 + pointload(i,2);       % Finds eq. number
    force(ieqn) = force(ieqn) + pointload(i,3);         % add the force
  end
```

Figura 12: Cálculo de las fuerzas nodales equivalentes por carga concentrada.

Desplazamientos prescritos

Inicialmente se define el vector u que contendrá los valores de los grados de libertad del sistema. En la figura 13 se puede ver el ciclo sobre los grados de libertad prescritos y como sus valores definidos en la matriz `fixnodes` son asignados al vector u. Igualmente se define un vector `fix` que contiene el número de las ecuaciones de aquellos grados de libertad que se encuentran restringidos.
Finalmente se actualiza el vector `force` de cargas nodales equivalentes con el producto de la matriz de rigidez global `StifMat` por el vector u que en este momento contiene exclusivamente los valores de aquellos grados de libertad que han sido restringidos.

```
%  Applies the Dirichlet conditions and adjust the right hand side.
  u = sparse ( nndof, 1 );
  for i = 1 : size(fixnodes,1)
    ieqn = (fixnodes(i,1)-1)*2 + fixnodes(i,2);  %Finds the equation number
    u(ieqn) = fixnodes(i,3);                      %and store the solution in u
    fix(i) = ieqn;                             % and mark the eq as a fix value
  end
  force = force - StifMat * u;       % adjust the rhs with the known values
```

Figura 13: Actualización del vector de fuerzas debido a los grados de libertad prescitos.

Solución del sistema de ecuaciones

la estrategia utilizada en MAT-fem consiste en solucionar el sistema de ecuaciones global sin tener en cuenta aquellos grados de libertad cuyos valores son conocidos y que han sido identificados, como se ha visto en el apartado anterior. El vector `FreeNodes` contiene la lista de aquellas ecuaciones de hay que resolver.

El vector `FreeNodes` es utilizado como un índice de los grados de libertad desconocidos y nos permite escribir de manera sencilla la solución al sistema de ecuaciones. MATLAB se encargan de escoger el algoritmo más adecuado para la solución del problema que se plantea, siendo totalmente transparente para el usuario la solución del sistema de ecuaciones. Las rutinas de solución de ecuaciones implementadas en MATLAB compiten en velocidad y optimización de memoria con los mejores algoritmos existentes hoy en día, por lo que toda la potencia de cálculo disponible queda descrita en una simple línea.

```
%  Compute the solution by solving StifMat * u = force for the
%  remaining unknown values of u.
  FreeNodes = setdiff ( 1:nndof, fix ); % Finds the free node list and
                                        % solve for it.
  u(FreeNodes) = StifMat(FreeNodes,FreeNodes) \ force(FreeNodes);
```

Figura 14: Solución del sistema de ecuaciones.

Cálculo de reacciones

La solución al sistema de ecuaciones se encuentra reflejada en el vector `u` por lo que el cálculo de las reacciones se realiza mediante la expresión `R = StifMat*u – force` obviamente el valor de la reacción en aquellos nodos no prescritos es nulo. Para evitar realizar cálculos innecesarios se hace uso del vector `fix` que contiene la lista del número de las ecuaciones asociadas a los grados de libertad por lo que sólo se calcula el vector de reacciones en aquellas ecuaciones que son necesarias como lo muestra la figura 15.

```
%  Compute the reactions on the fixed nodes as a R = StifMat * u - F
  reaction = sparse(nndof,1);
  reaction(fix) = StifMat(fix,1:nndof) * u(1:nndof) - force(fix);
```

Figura 15: Cálculo de las reacciones.

Cálculo de los esfuerzos.

Una vez encontrados los desplazamientos en los nodos es posible evaluar los esfuerzos dentro de los elementos. Dichos esfuerzos son evaluados dentro de cada uno de los elementos mediante la expresión **DBu** y dado que la matriz de deformaciones **B** se encuentra calculada en los puntos de integración, las tensiones calculadas se refieren a dichos puntos. Para transportar los valores de los esfuerzos en los puntos de integración hacia los nodos del elemento es necesario revisar con más detalle y en un apartado posterior el cálculo de estos valores. En la figura 16 se muestra la llamada a la subrutina de cálculo de esfuerzos que regresa en la matriz `Strnod` el valor de los esfuerzos calculados en los nodos.

```
% Compute the stresses
  Strnod = Stress(dmat,poiss,thick,pstrs,u);
```

Figura 16: Cálculo de las reacciones.

Escritura para postproceso

Finalmente una vez calculados los desplazamientos nodales, las reacciones y los esfuerzos se procede a volcar estos valores a los ficheros de postproceso de donde GiD podrá presentarlos de manera gráfica dichos resultados. Esto se realiza en la llamada a la subrutina `ToGiD` mostrada en la figura 17

```
% Graphic representation.
  ToGiD (file_name,u,reaction,Strnod);
```

Figura 17:Llamada a la rutina de Postproceso.

En la consola de MATLAB aparecerá a lo largo de la ejecución del programa, el tiempo total utilizado por el programa así como el tiempo consumido en cada una de las fases descritas. Se puede observar que el consumo principal de tiempo se realiza en el cálculo y el ensamblaje de la matriz de rigidez global, mientras que la solución del sistema de ecuaciones realmente representa un porcentaje pequeño del tiempo consumido.

Obtenida la ejecución las variables utilizadas continúan activas dentro de MATLAB por lo que es posible experimentar con la batería de funciones propias de MATLAB.

Más sobre el cálculo de esfuerzos

El cálculo de los esfuerzos en el interior de los elementos requiere el uso de una subrutina especifica, no sólo para el cálculo de estos sino también para realizar la proyección del valor calculado en los puntos de integración hacia los nodos del elemento.

La rutina `stress` controla el flujo del programa hacia las rutinas para los elementos de tres nodos o de cuatro nodos. En el caso del elemento triangular los esfuerzos son constantes dentro de todo el elemento por lo que la extrapolación hacia los nodos es trivial. No así es el caso del elemento cuadrilátero donde los esfuerzos presentan una variación bilineal por lo que se realiza la extrapolación de los esfuerzos hacia los nodos haciendo uso de las mismas funciones de forma del elemento.

En la figura 18 se muestra la inicialización de la rutina `stress` en donde los datos de entrada son la matriz constitutiva del material `dmat`, el coeficiente de Poisson `poiss`, el espesor `thick`, el indicador del tipo de problema `pstrs` y los desplazamientos nodales encontrados `u`. Adicionalmente se utilizará las coordenadas nodales `coordinates` y las conectividades `elements` que han sido definidas como variables globales. Para simplificar el entendimiento de la rutina se extraen las variables `nelem` para indicar el número de elementos, `nnode` para indicar el número de nodos por elemento y `npnod` que indica el número total de nodos en el dominio.

No hay que perder de vista que para los problemas de tensión plana el número de esfuerzos es de tres σ_x, σ_y y γ_{xy} mientras que si el problema es de deformación plana el número de esfuerzos es de cuatro: σ_x, σ_y, σ_z y γ_{xy}. En este caso el esfuerzo en z es función de los esfuerzos σ_x y σ_y.

La matriz `nododstr` se inicializa a ceros para almacenar el valor de los esfuerzos nodales; en la última columna se almacenarán el contador de elementos que concurren en un mismo nodo para poder realizar sus avezados esfuerzos

```
function S = Stress (dmat,poiss,thick,pstrs,u)

%% Stress Evaluates the stresses at the gauss point and smooth the values
%         to the nodes.
%
  global coordinates;
  global elements;

  nelem  = size(elements,1);           % Number of elements
  nnode  = size(elements,2);           % Number of nodes per element
  npnod  = size(coordinates,1);        % Number of nodes

  if (pstrs==1)
      nstrs= 3;                        % Number of Strs. Sx Sy Txy
  else
      nstrs= 4;                        % Number of Strs. Sx Sy Sz Txy
  end
  nodstr = zeros(npnod,nstrs+1);
```

Figura 18:Inicialización de variables para el cálculo de los esfuerzos.

Al igual que el cálculo de la matriz de rigidez, el cálculo de los esfuerzos requiere realizar un ciclo sobre todos y cada uno de los elementos del dominio, recuperando las conectividades elementales (`lnods`), las coordenadas dichos nodos (`coord`) y los desplazamientos en la variable `displ` como se muestra en la figura 19

```
  for ielem = 1 : nelem

% Recover element properties
    lnods = elements(ielem,:);
    coord(1:nnode,:) = coordinates(lnods(1:nnode),:);
    eqnum = [];
    for i =1 : nnode
      eqnum = [eqnum,lnods(i)*2-1,lnods(i)*2];
    end
    displ = u(eqnum);
```

Figura 19:Recuperación de las coordenadas y desplazamientos nodales.

En la figura 20 se muestra primeramente la llamada a la rutina de cálculo de esfuerzos para elementos triangulares y en segundo termino, el cálculo para elementos cuadriláteros. En el caso de los elementos triangulares la rutina regresa el vector `ElemStr` que contiene los esfuerzos en el elemento; dichos esfuerzos se acumulan en la variable `nodstr` contabilizando en la última columna la contribución elemental para posteriormente realizar el promedio nodal.

En el caso de los elementos cuadriláteros la variable `ElemStr` representa una matriz que contiene los esfuerzos de cada uno de los puntos nodales del elemento, por lo que la acumulación se realiza tomando en consideración este hecho. De manera similar al caso anterior en la última columna se almacena la contabilidad de la contribución elemental.

Finalmente, después de haber calculado los esfuerzos en todos y cada uno de los elementos y haber realizado la acumulación de las contribuciones en la variable `nodstr` se realiza un promediado nodal para obtener así los esfuerzos suavizados en los nodos. Como se muestra en la misma figura.

```
% Stresses inside the elements.
    if (nnode == 3)

% Triangular elements constant stress
      ElemStr = TrStrs(coord,dmat,displ,poiss,thick,pstrs);

      for j=1 : nstrs
        nodstr(lnods,j) = nodstr(lnods,j) + ElemStr(j);
      end
      nodstr(lnods,nstrs+1) = nodstr(lnods,nstrs+1) + 1;

    else
% Quadrilateral elements stress at nodes
      ElemStr = QdStrs(coord,dmat,displ,poiss,thick,pstrs);

      for j=1 : 4
        for i = 1 : nstrs
          nodstr(lnods(j),i) = nodstr(lnods(j),i) + ElemStr(i,j);
        end
      end
      nodstr(lnods,nstrs+1) = nodstr(lnods,nstrs+1) + 1;
    end
  end
% Find the mean stress value

  S = [];
  for i = 1 : npnod
   S = [S ; nodstr(i,1:nstrs)/nodstr(i,nstrs+1)];
  end
```

Figura 20:Cálculo de esfuerzos y suavizado nodal.

El cálculo de los esfuerzos para cada uno de los elementos se muestra en las siguientes figuras. Para el caso del elemento triangular, se recalcula la matriz **B** de la misma manera que fue hecho para evaluar la matriz de rigidez. Como se ha comentado, se ha optado por recalcular dicha matriz en lugar de almacenarla dado que computacionalmente resulta más económico. Los esfuerzos son directamente el producto **DBu**

```
function S = TrStrs (nodes,dmat,displ,poiss,thick,pstrs)

  b(1) = nodes(2,2) - nodes(3,2);
  b(2) = nodes(3,2) - nodes(1,2);
  b(3) = nodes(1,2) - nodes(2,2);

  c(1) = nodes(3,1) - nodes(2,1);
  c(2) = nodes(1,1) - nodes(3,1);
  c(3) = nodes(2,1) - nodes(1,1);

  area2 = abs(b(1)*c(2) - b(2)*c(1));
  area = area2 / 2;

  bmat = [b(1),  0 ,b(2),  0 ,b(3),  0 ;
            0 ,c(1),  0 ,c(2),  0 ,c(3);
          c(1),b(1),c(2),b(2),c(3),b(3)];

  se = (dmat*bmat*displ)/area2;
```

Figura 21: Cálculo de los esfuerzos para el elemento triangular de tres nodos.

Dependiendo del tipo de problema seleccionado, los esfuerzos que se pueden calcular

son tres para un problema de tensión plana (σ_x, σ_y y γ_{xy}) y cuatro para un problema de deformación plana (σ_x, σ_y, σ_z y γ_{xy}) siendo σ_z = -poiss*(σ_x + σ_y) ; tal y como se muestra en la figura 22.

```
%  Plane Sress
  if (pstrs==1)
       S = se ;
%  Plane Strain
   else
       S = [se(1),se(2),-poiss*(se(1)+se(2)),se(3)];
 end
```

Figura 22: Cálculo de los esfuerzos para tensión o deformación plana.

Dada la similitud existen con el cálculo de la matriz de rigidez, es posible referirnos a dicho apartado para describir las variables básicas en la formación y evaluación de la matriz de deformaciones del elemento cuadrilátero de cuatro nodos. Sin embargo, el aspecto más importante a tomar en cuenta es la extrapolación de las tensiones evaluadas en los puntos de gauss hacia los nodos.

Como se ha comentado, la variación de los esfuerzos es lineal dentro del elemento, por lo tanto, se hace uso de las funciones de forma para extrapolar los valores a los nodos. Para ello se considera que los puntos *i* y *j* se encuentran en un dominio que varia de –1/p a +1/p siendo p la distancia de colocación de los puntos de gauss, lo que les hace estar a una distancia de –1 y +1 como se muestra en la figura 23 a).

Otro aspecto a tomar en cuenta consiste en establecer una adecuada correlación entre la numeración de los nodos y la numeración de los puntos de gauss, a fin de poder hacer coincidir adecuadamente los ciclos ya que el orden que guardan los puntos de integración no coinciden con la numeración nodal del elemento como se muestra en la figura 23 b).

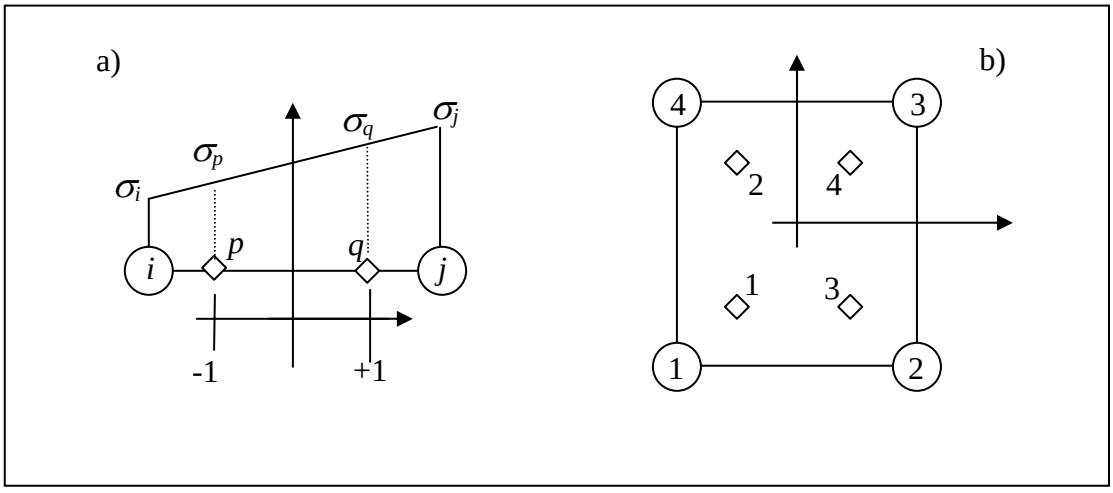

Figura 23: Extrapolación de esfuerzos

Dichas consideraciones son parte del código mostrado en la figura 24 donde calculan, armando la matriz de deformaciones B y extrapolan los esfuerzos a los puntos cálculos en los puntos de gauss a los nodos del elemento.

```
function S = QdStif (nodes,dmat,displ,poiss,thick,pstrs)

    fform = @(s,t)[(1-s-t+s*t)/4,(1+s-t-s*t)/4,(1+s+t+s*t)/4,(1-s+t-
s*t)/4];
    deriv = @(s,t)[(-1+t)/4,( 1-t)/4,( 1+t)/4,(-1-t)/4 ;
                   (-1+s)/4,(-1-s)/4,( 1+s)/4,( 1-s)/4 ];

    pospg = [ -0.577350269189626E+00 , 0.577350269189626E+00 ];
    pespg = [  1.0E+00 , 1.0E+00];

    strsg = [];
    extrap = [];
    order = [ 1 , 4 ; 2 , 3 ]; % Align the g-pts. with the element
corners

    for i=1 : 2
       for j=1 : 2
          lcder = deriv(pospg(i),pospg(j)) ;    % FF Local derivatives
          xjacm = lcder*nodes ;                 % Jacobian matrix
          ctder = xjacm\lcder ;                 % FF Cartesian derivates

          bmat = [];
          for inode = 1 : 4
            bmat = [ bmat , [ctder(1,inode),            0 ;
                                        0 ,ctder(2,inode);
                             ctder(2,inode),ctder(1,inode) ] ] ;
          end

          strsg(:,order(i,j)) = (dmat*bmat*displ) ;

          a = 1/pospg(i);
          b = 1/pospg(j);

         extrap(order(i,j),:) = fform(a,b) ;
       end
    end

    se = transpose(extrap*transpose(strsg));
```

Figura 23: Extrapolación de esfuerzos

Al igual que en el caso del elemento triangular, dependiendo del tipo de problema, se calculan los esfuerzos correspondientes al problema de tensión plana o de deformación plana como lo muestra la figura 24

```
%  Plane Sress
  if (pstrs==1)
       S = se ;
%  Plane Strain
   else
       S = [se(1,:) ; se(2,:) ; -poiss*(se(1,:)+se(2,:)) ; se(3,:)];
   end
```

Figura 24: Cálculo de los esfuerzos para tensión o deformación plana

Descripción de la interfase

En este apartado se revisa con detalle la interfaz implementada dentro del programa

GiD. Para acceder a ella es necesario seleccionar del menú DATA -> PROBLEM TYPE el módulo correspondiente a la interfaz MAT-.fem. Al iniciar, se presentara la imagen mostrada en la figura 25.

Todas las características de GiD forman parte ahora del modulo MAT-fem por lo que la generación de geometrías, importación y manejo de estas, así como las técnicas de discretización proporcionan a MAT-fem capacidades difíciles de superar para un código educativo.

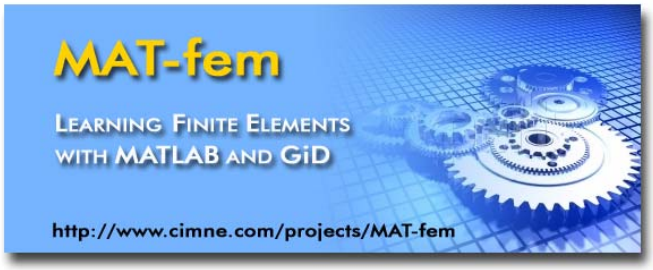

Figura 24: Interfaz de MAT-fem

Al igual que MATLAB, para GiD también existe mucha información disponible en Internet, se recomienda visitar la página principal de GiD en www.gidhome.com.

Seguir los pasos necesarios en MAT-fem para generar un problema es muy sencillo. Una vez definida la geometría basta con seguir de arriba a abajo los iconos mostrados en el menú gráfico.

El primer botón al sirve para identificar aquellas entidades geométricas que presentan restricción de movimientos. Al presionar sobre dicho botón una ventana emergente (Figura 25) aparecerá para seleccionar los nodos o líneas a los cuales se debe aplicar la condición de desplazamientos restringidos. En dicha ventana existen casilleros para identificar que dirección es la que está vinculada. Asimismo se puede optar por un valor diferente de 0 para el movimiento restringido.

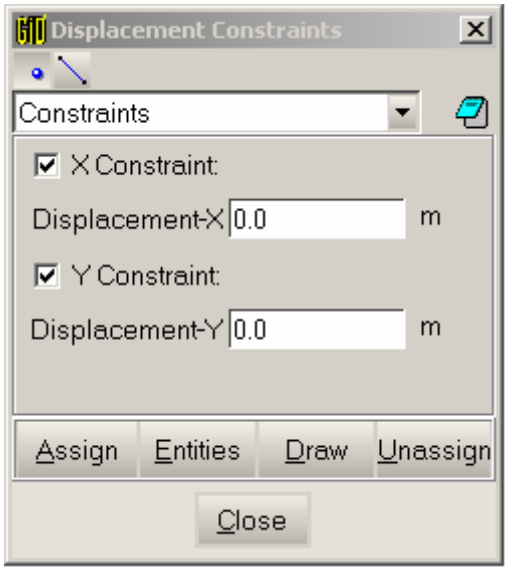

Figura 25: Asignación de restricciones del movimiento.

El botón mostrado se utiliza para la asignación de cargas puntuales. Al seleccionarlo una ventana emergente (figura 26) permite introducir el valor de la carga puntual en sus componentes x e y referidas al sistema global de coordenadas. Una vez definida la carga se procede a seleccionar los nodos sobre los cuales existe dicha carga.

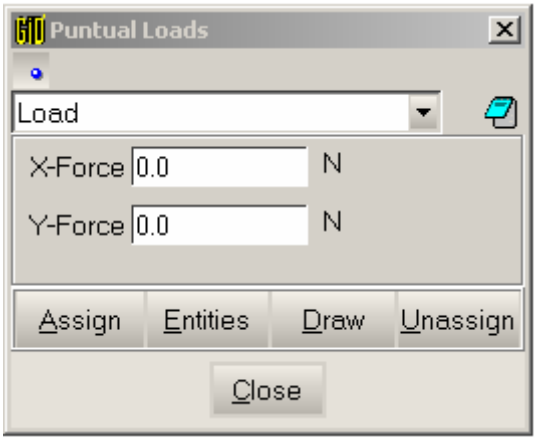

Figura 26: Asignación de cargas puntuales.

El botón asociado a las cargas uniformemente repartida permite asignar dicha condición sobre las líneas que forman parte de la geometría. La ventana emergente (Figura 27) permite introducir el valor de la carga por unidad de longitud siendo las cargas siempre referidas al sistema cartesiano de ejes.

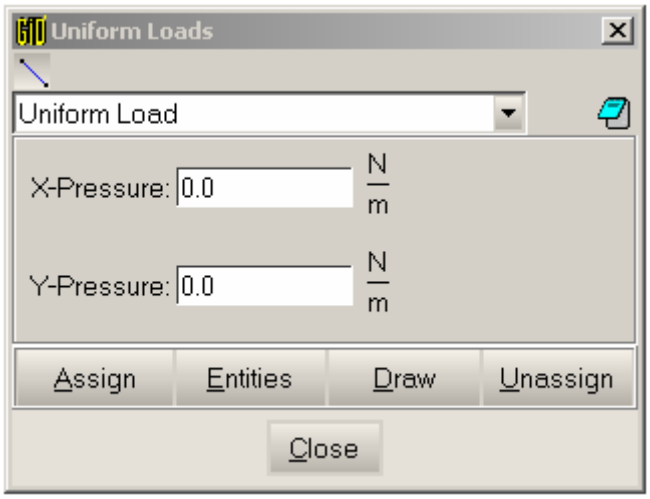

Figura 27 Asignación de cargas uniformemente repartidas.

La definición de las propiedades del material se realiza en el siguiente botón el cual conduce a una nueva ventana emergente (Figura 28) en donde se definen las variables asociadas al módulo de elasticidad, coeficiente de Poisson, densidad y espesor. Es necesario asignar este material a todas las áreas que definen el dominio del programa. En MAT-fem, por razones simplificativas sólo es permitido el uso de un único material.

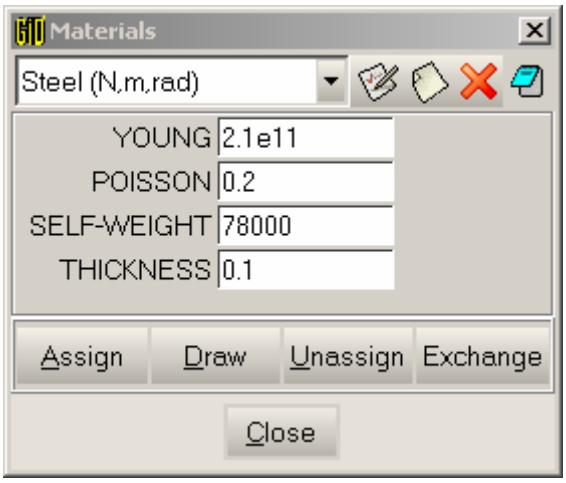

Figura 28 Definición de las propiedades del material.

El botón de propiedades generales permite acceder a la ventana mostrada en la figura 29 donde se identifica el título del problema, el tipo (Tensión o Deformación plana) si se ha de tener en cuenta el peso propio como condición de carga y finalmente el sistema de unidades en los cuales deben de expresarse los resultados calculados.

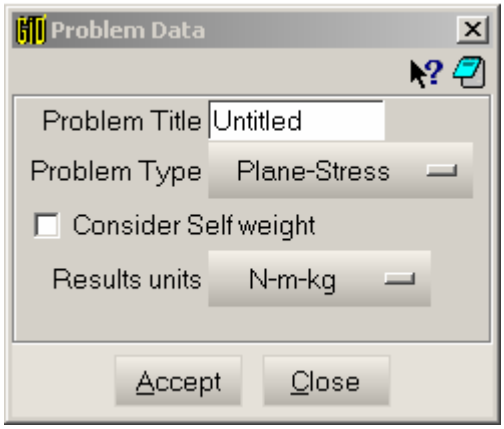

Figura 29 Propiedades generales del problema.

Una vez definidas las condiciones de contorno y las propiedades geométricas es necesario realizar la discretización del dominio; para ello se utiliza el botón mostrado mediante el cual se hacen uso de todas las herramientas disponibles en GiD para realizar la malla más conveniente, el tipo de elemento y su tamaño medio.

La escritura del fichero de datos se realiza al presionar el último botón mostrado en el menú. Tanto las propiedades geométricas como topológicas del problema, así como el material y las propiedades generales son volcadas al fichero de datos en el formato de lectura necesario para MAT-fem. No hay que perder de vista que es necesaria la extensión .m para dicho fichero.

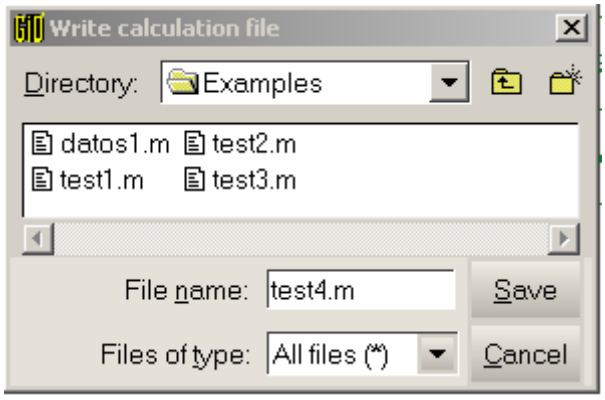

Figura 30 Escritura del fichero de datos.

Proceso.

El cálculo del problema se realiza, como ya ha sido descrito, dentro de programa MATLAB. La ejecución no tiene mayores complicaciones que el saber en que directorio se encuentra el fichero escrito y tener en cuenta que en el directorio de trabajo quedarán escritos los ficheros para postproceso.

Postproceso.

Una vez concluida la ejecución del problema en MATLAB, es necesario regresar a GiD dentro del postproceso con la finalidad de estudiar los resultados obtenidos. Para ello es necesario seleccionar y abrir cualquiera de los dos ficheros generados y que contienen la extensión *.flavia.msh o *.flavia.res.

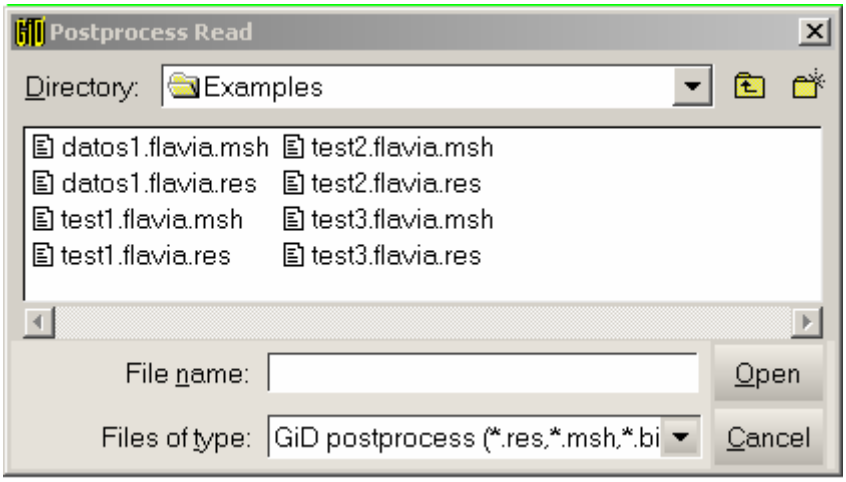

Figura 31 Lectura de los ficheros de postproceso.

La visualización de los resultados obtenidos puede hacerse en diversidad de formas, dadas las posibilidades gráficas de GiD que permite mostrar los resultados como un gradiente de colores, isolíneas, cortes y gráficas; permitiendo la sencilla interpretación de los resultados obtenidos.

En la figura 32 se muestra por ejemplo las iso áreas de desplazamiento vertical en una probeta empotrada en su extremo izquierdo y sujeta a una carga axial en el derecho.

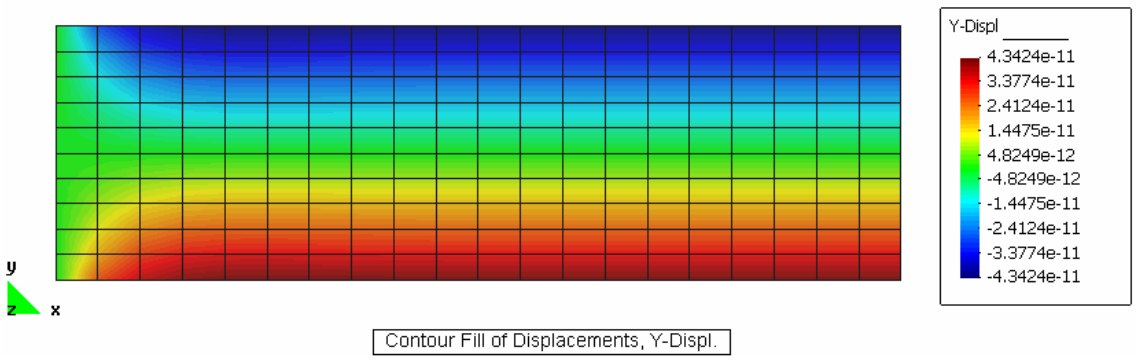

Figura 32 Postproceso de una probeta sujeta a carga axial.

Ejemplo.

De manera exclusivamente ilustrativa se muestra un sencillo ejemplo del uso del programa donde se calcula los esfuerzos y desplazamientos de una pared circular delgada sujeta a una carga puntual. El ejemplo es el ejercicio de validación IC6 propuesto por NAFEMS.

En este apartado se describe con detalle el fichero de datos para el ejemplo propuesto utilizando elementos cuadriláteros. También se analiza la convergencia de los resultados variando las discretizaciones del problema con elementos triangulares y cuadriláteros.

El ejemplo corresponde al problema IC6 propuesto por NAFEMS en linear statics benchmarks vol 1. Se trata de una laja circular con carga puntual. En el recuadro de la figura 33 se observa la geometría completa del problema mientras que haciendo uso de la simetría existente la geometría a utilizar, al igual que las propiedades del material son descritas en la misma figura. El problema es de deformación plana.

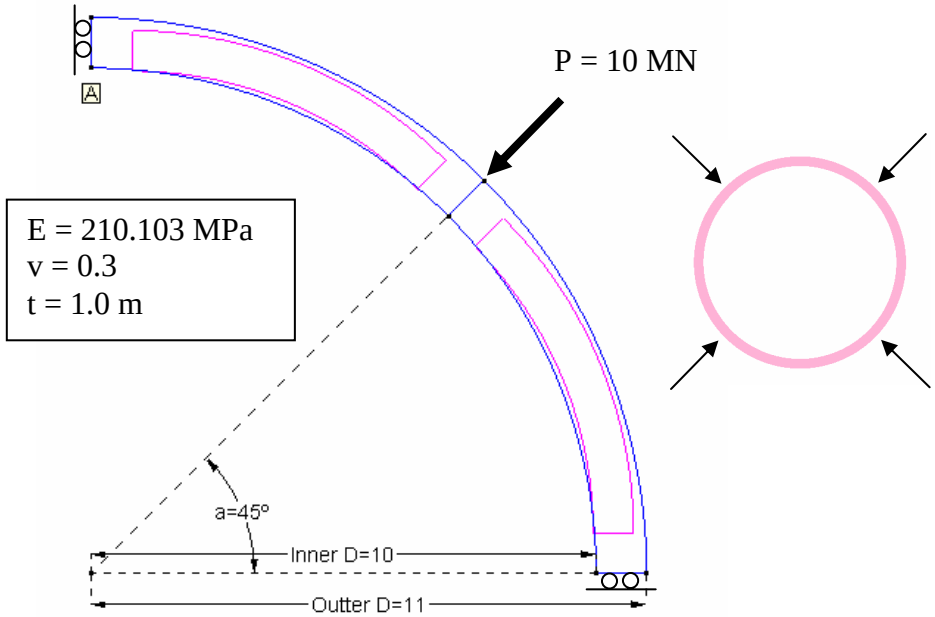

Figura 33 Laja circula con carga puntual NAFEMS IC6.

La laja se encuentra impedida en el desplazamiento normal a su sección transversal en ambos extremos y la carga puntual se encuentra aplicada en el punto medio de la cara exterior y a 45º de dirección. El objetivo es encontrar el valor de la tensión σx en el punto A, cuyo valor exacto es σx = 53.2MPa

La introducción de datos en el programa es como se describió en el apartado correspondiente a la interfase, haciendo uso del menú de opciones para asignar las condiciones de contorno, las cargas y el material como se muestra en las figuras 34 y 35.

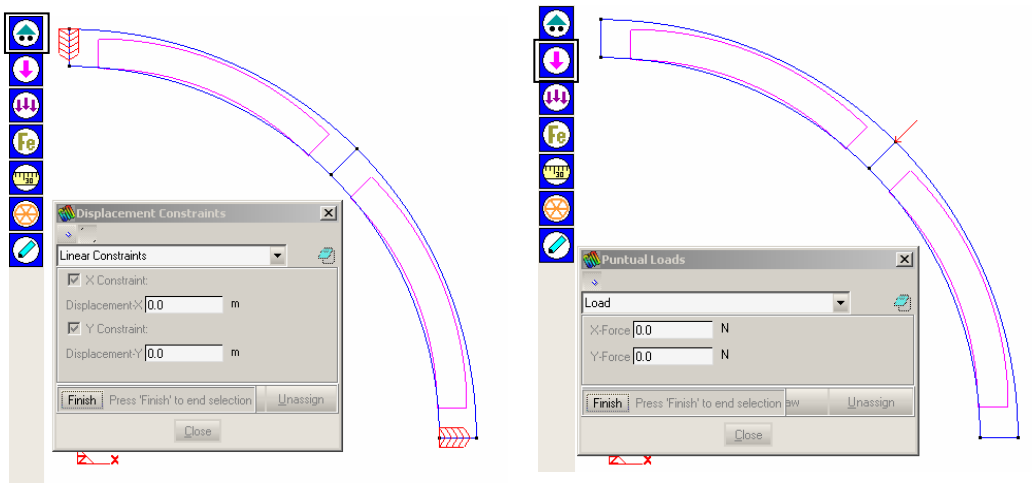

Figura 34 Condiciones de contorno y cargas sobre la estructura.

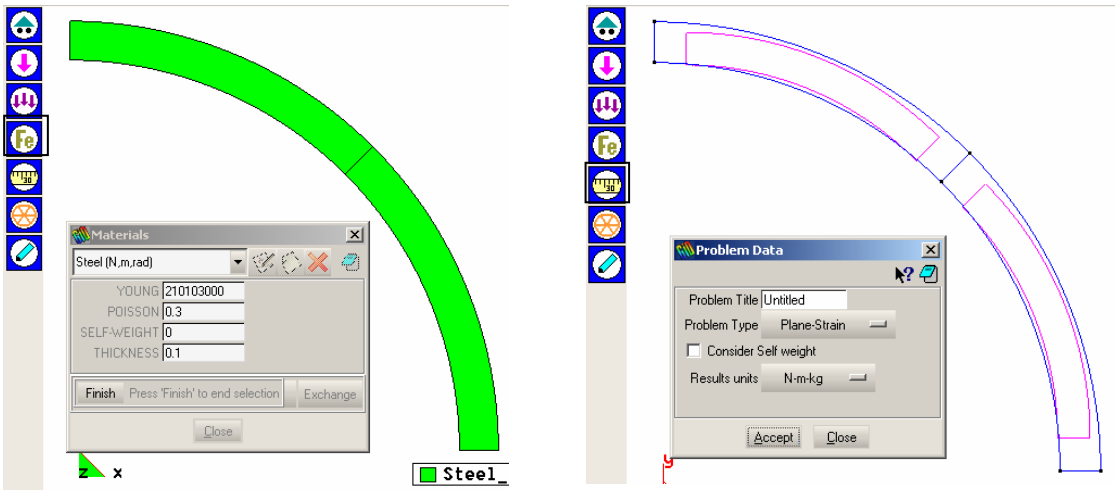

Figura 35 Definición del material y de las características del problema.

Finalmente la discretización y la escritura del fichero para MAT-fem se realizan con los dos últimos botones del menú de MAT-fem

Las mallas a utilizar son estructuradas y para el caso más simple de elementos cuadriláteros, consta de 8 elementos y 15 nodos. La figura 36 muestra la numeración nodal y elemental de la malla utilizada; misma que servirá de referencia para el fichero de datos que se describe en éste apartado. En la misma figura se muestra el código del fichero de entrada de datos.

.

```
%========================================================================
% MAT-fem 1.0  - MAT-fem is a learning tool for undestanding
%                the Finite Element Method with MATLAB and GiD
%========================================================================
% PROBLEM TITLE = NAFEMS IC6
%  Material Properties
%
  young =      210103000.00000 ;
  poiss =              0.30000 ;
  denss = 0.00 ;
  pstrs =  0 ;
  thick =  1 ;
%
% Coordinates
%
global coordinates
coordinates = [
        11.00000   ,         0.00000  ;
        10.50000   ,         0.00000  ;
        10.00000   ,         0.00000  ;
         9.23880   ,         3.82683  ;
         9.70074   ,         4.01818  ;
        10.16267   ,         4.20952  ;
         7.07107   ,         7.07107  ;
         7.42462   ,         7.42462  ;
         7.77817   ,         7.77817  ;
         3.82683   ,         9.23880  ;
         4.01818   ,         9.70074  ;
         4.20952   ,        10.16267  ;
         0.00000   ,        10.00000  ;
         0.00000   ,        10.50000  ;
         0.00000   ,        11.00000  ]                                    ;
%
% Elements
%
global elements
elements = [
      2   ,      5   ,      4   ,      3   ;
      1   ,      6   ,      5   ,      2   ;
      5   ,      8   ,      7   ,      4   ;
      6   ,      9   ,      8   ,      5   ;
      8   ,     11   ,     10   ,      7   ;
      9   ,     12   ,     11   ,      8   ;
     11   ,     14   ,     13   ,     10   ;
     12   ,     15   ,     14   ,     11   ] ;
%
% Fixed Nodes
%
fixnodes = [
      1  , 2 ,    0.00000  ;
      2  , 2 ,    0.00000  ;
      3  , 2 ,    0.00000  ;
     13  , 1 ,    0.00000  ;
     14  , 1 ,    0.00000  ;
     15  , 1 ,    0.00000  ] ;
%
% Point loads
%
pointload = [
      9  , 1 , -7071067.81200  ;
      9  , 2 , -7071067.81200  ] ;
%
% Side loads
%
sideload = [ ];
```

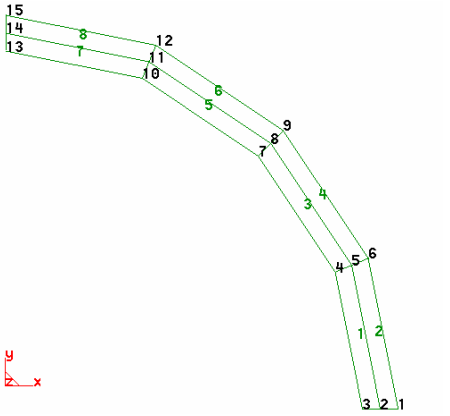

Figura 36 Fichero de entrada de datos para el problema NAFEMS IC6.

Como se puede observar, el código de entrada de datos contiene todas las secciones descritas en apartados anteriores, las propiedades de los materiales, las coordenadas nodales (en orden consecutivo), las conectividades elementales, así como las condiciones de apoyo, cargas puntuales y las cargas repartidas que están definidas por una matriz vacía.

La ejecución del programa se realiza dentro del MATLAB con el comando MATfem, después de haber seleccionado el directorio de trabajo donde se encuentra el programa. Se puede observar que el mayor consumo de tiempo se lleva en el ensamblaje de la matriz de rigidez, debido al sistema de almacenamiento donde los índices internos de la matriz sparse deben ser actualizados. El tiempo total de ejecución de este problema ha sido de 0.15 segundos.

```
>> MATfem
Enter the file name :cilQ2
Time needed to read the input file     0.003399
Time needed to  set initial values     0.000912
Time to assamble the global system     0.032041
Time for apply side and point load     0.000196
Time  to solve the stifness matrix     0.026293
Time to  solve the  nodal stresses     0.024767
Time  used to write  the  solution     0.066823

 Total running time     0.154431
>>
```

Figura 37 Consola de MATLAB para el problema NAFEMS IC6.

Las facilidades dadas por MATLAB se muestras en la figura 38 donde con el comando **`spy(StifMat)`** se puede visualizar el perfil de la matriz de rigidez de este problema. Con la ayuda de otros comandos es posible conocer con detalle las propiedades de la matriz de rigidez, como los autovalores, el rango, su determinante, etc.

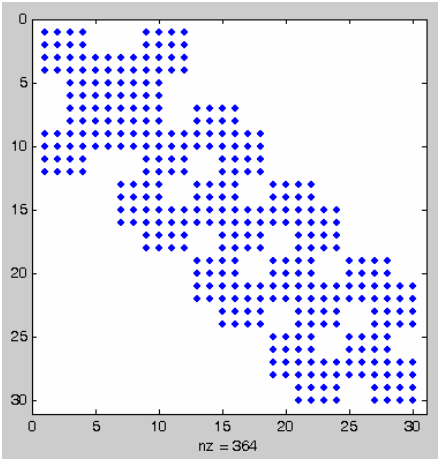

Figura 38 Perfil de la matriz de rigidez global de la estructura.

En la figura 39 se muestran los resultados para la deformada así como la distribución de esfuerzos σx en el arco

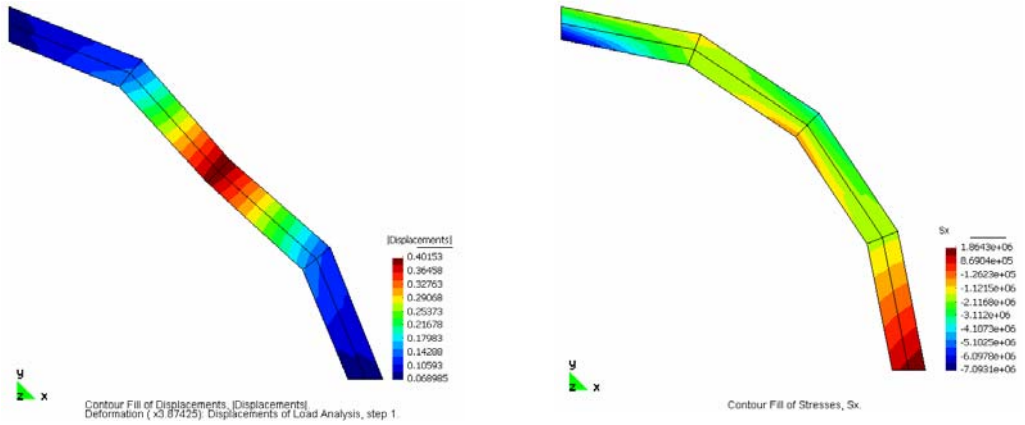

Figura 39 Deformada y distribución de esfuerzos σx

En el caso analizado la tensión σx en el punto A es de 14,186 MPa relativamente lejos de la solución del problema, sin embargo, al ir aumentando los grados de libertad del problema se alcanza el valor buscado como se muestra en la tabla de la figura 40 en donde se compara la convergencia del desplazamiento máximo y la tensión en el punto A para mallas estructuradas de triángulos y cuadriláteros. La grafica correspondiente se muestra en la figura 41.

TRIANGULOS				CUATRILATEROS			
NODOS	DOF	DESP	Sx TR	NODOS	DOF	DESP SQR	Sx SQR
15	30	0,58607	7,8835E+06	15	30	0,80306	1,4186E+07
28	56	0,80865	9,8442E+06	28	56	1,30890	2,1405E+07
45	90	1,06980	1,2561E+07	45	90	1,79900	2,7571E+07
66	132	1,34800	1,5697E+07	66	132	2,22210	3,2746E+07
120	240	1,88800	2,2193E+07	120	240	2,84280	4,0014E+07
231	462	2,54040	3,0504E+07	231	462	3,36800	4,5889E+07
1326	2652	3,73390	4,6827E+07	861	1722	3,91130	5,1518E+07
1891	3782	3,84840	4,8496E+07	1891	3782	4,03990	5,2669E+07
3321	6642	3,97260	5,0340E+07	3321	6642	4,09080	5,3056E+07

Figura 40 Convergencia

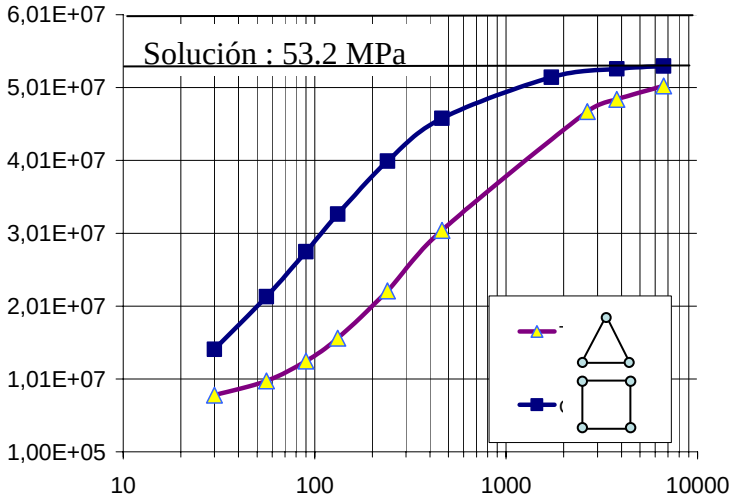

Figura 41 Convergencia en esfuerzos