UNIÓN
REVISTA IBEROAMERICANA DE EDUCACIÓN MATEMÁTICA

ISSN: 1815-0640
Número 45. Marzo 2016
Página 242-268

www.fisem.org/web/union
http://asenmacformacion.com/ojs/index.php/union/index

MATLAB: Una herramienta para la didáctica del Método de los Elementos Finitos

Emilio Martínez-Pañeda

Fecha de recepción: 20/10/2015
Fecha de aceptación: 19/03/2016

Resumen	El método de los elementos finitos (MEF) es a día de hoy el método numérico más utilizado en aplicaciones de ciencia e ingeniería. Sin embargo, el desarrollo de procedimientos efectivos para la enseñanza del mismo continúa siendo un desafío para los docentes de todo el mundo. En el presente trabajo se plantea el uso del software matemático MATLAB para el desarrollo de códigos con orientación docente. Los ejemplos presentados revelan la idoneidad del enfoque adoptado para reducir la brecha existente entre el sustento matemático del método y la sencillez de uso de los potentes programas comerciales. **Palabras clave**: Método de los Elementos Finitos, didáctica, MATLAB
Abstract	The finite element method (FEM) is nowadays the most widely used numerical method in science and engineering applications. However, the development of effective ways to teach it remains a challenge for teachers worldwide. In the present work, the use of the mathematical software MATLAB is proposed for the development of teaching-oriented codes. The examples presented reveal the suitability of the suggested approach to close the gap between the mathematical theory of the method and the ease of use of the powerful commercial programs. **Keywords**: Finite Element Method, teaching, MATLAB
Resumo	O método dos elementos finitos (FEM) é hoje o método numérico mais amplamente utilizado em aplicações científicas e de engenharia. No entanto, o desenvolvimento de maneiras eficazes de ensinar continua a ser um desafio para os professores em todo o mundo. No presente trabalho, é proposta a utilização do software MATLAB para o desenvolvimento de códigos orientada para o ensino. Os exemplos apresentados revelam a aptidão da abordagem sugerida para fechar o espaço entre a teoria matemática do método e a facilidade de utilização dos programas comerciais. **Palavras-chave:** Método dos elementos finitos, Ensino, MATLAB

1. Introducción: La docencia del Método de los Elementos Finitos

Desarrollado en la década de los 60 para el análisis de estructuras, el método de los elementos finitos (MEF) se ha convertido en la herramienta computacional más utilizada en aplicaciones de ciencia e ingeniería, extendiendo su uso más allá del análisis de solicitaciones tensionales para abarcar un amplio rango de áreas de conocimiento, que van desde el análisis de vibraciones hasta la transferencia de calor, pasando por el flujo de fluidos o los campos electromagnéticos, entre otras (ver, p. ej., Martínez-Pañeda y Gallego, 2015, Martínez-Pañeda y Betegón, 2015, Martínez-Pañeda y Niordson, 2015).

El MEF es un método numérico general para la aproximación de soluciones de ecuaciones diferenciales parciales. A grandes rasgos, el proceder del método radica en la división del dominio, cuerpo o estructura en elementos o parcelas de reducido tamaño (elementos finitos) en los que se define la variable a calcular y sobre los que se aplica la aproximación. El procedimiento implica la caracterización y correspondiente definición del comportamiento de cada elemento por separado y el posterior ensamblaje de los mismos, obteniendo una solución aproximada del comportamiento global del sistema discretizado (Oñate, 1995).

De manera que la obtención, en determinadas aplicaciones, de una solución razonablemente precisa mediante el MEF conlleva la resolución de una enorme cantidad de ecuaciones diferenciales y en consecuencia, un uso efectivo y práctico del método está ligado al uso de ordenadores. El vertiginoso desarrollo de la tecnología informática ha traído consigo una rápida popularización del MEF en la industria y una extensa proliferación de programas comerciales con capacidad para modelizar geometrías complejas y resolver mediante el MEF un sinfín de problemas prácticos de ingeniería. De manera que un profundo conocimiento del MEF se postula a día de hoy como indispensable para cualquier ingeniero y en consecuencia el análisis por elementos finitos es ya objeto de estudio en casi la totalidad de las carreras técnicas en España y en el resto del mundo.

Sin embargo, el desarrollo de procedimientos efectivos para la enseñanza del MEF continúa siendo un desafío para los docentes (Kosasih, 2010). El proceder habitual en la docencia de la asignatura consiste en combinar las clases teóricas en el aula con sesiones de prácticas en ordenador. Siendo usual en estas últimas el uso de uno o varios de los programas comerciales de elementos finitos más populares (ABAQUS, ANSYS, etc.), a los que tienen acceso con licencia educativa muchas universidades. Este software comercial posee una interfaz de usuario muy sencilla de entender y manejar, lo que trae consigo una rápida familiarización del alumnado con los programas. Sin embargo este hecho, al contrario de lo que pudiera parecer, se transforma en numerosas ocasiones en un inconveniente para la docencia de la asignatura. Y es que uno de los principales obstáculos observados durante la docencia del método radica en la fuerte brecha existente entre la sencillez de uso de los programas comerciales y la teoría matemática del MEF, ya que el asequible manejo del software comercial

de elementos finitos permite al usuario desarrollar con éxito numerosos cálculos sin haber adquirido previamente los conocimientos de base del método. Así, en vista de la dificultad intrínseca al aprendizaje de un método con una fuerte carga matemática, los estudiantes tienden a elegir el camino más fácil, rehuyendo en lo posible el estudio del sustento matemático del MEF y adquiriendo vertiginosamente el conocimiento del uso del software comercial disponible.

Si este comportamiento no se remedia mediante los métodos de evaluación pertinentes, muchos estudiantes de ingeniería asumen la lógica errónea de estar capacitados para realizar cálculos por medio del MEF en sus futuros centros de trabajo por el hecho de poseer la habilidad de manejar correctamente el software comercial de elementos finitos sin comprender el procesamiento de los datos que realiza el mismo. Debido a que sólo un profundo conocimiento de la teoría que sustenta el MEF permite garantizar la validez de los resultados obtenidos y su posterior utilización en un diseño y dimensionamiento seguro, es necesario descubrir nuevas herramientas de docencia que ayuden a reducir la brecha existente entre el fundamento teórico del MEF y la cada día más sencilla interfaz de usuario que plantean los programas comerciales. Y es que aunque los métodos numéricos relacionados con el análisis por elementos finitos han sido desarrollados, investigados y discutidos en profundidad, las técnicas de enseñanza del MEF no han recibido aún la atención necesaria. En un primer momento varios autores (Zecher, 2002; Jolley, Rencis, & Grandlin, 2003) investigaron al respecto del enfoque más apropiado para la docencia del MEF en el contexto de asignaturas relacionadas con el estudio de la mecánica de los sólidos o los materiales, más tradicionales en los planes de estudio de las ingenierías. Sin embargo, debido a la rápida expansión de aplicaciones del MEF, la comunidad educativa ha adoptado un claro consenso al respecto de la necesidad de asignar a la docencia del MEF un espacio propio.

2. MATLAB como herramienta docente

Habida cuenta de que para una aplicación práctica del MEF es imprescindible el uso de ordenadores, la docencia del método debe obligatoriamente incluir el manejo de programas informáticos que basen sus cálculos en el MEF. En esa línea, varias propuestas han surgido en la comunidad académica. Así, mientras algunos autores se inclinan por la docencia mediante software comercial de elementos finitos como ANSYS (Earley, 1998; Backer, Capece, & Lee, 2001), ALGOR (Howard, Musto, & Prantil, 2001; Logue & Hall, 2001), COSMOS (Pike, 2001) o Pro/MECHANICA (Lissenden, Wagle, & Salamon, 2002), otros apuestan por el uso de programas de algebra computacional como Maple (Connell, Blyth, May, & Zorzan, 1999) o Mathematica (Jiang, & Wang, 2008), e incluso algunos docentes han recurrido a la utilización de las hojas de cálculo de Microsoft Excel (Teh, & Morgan, 2005).

Como ya se ha resaltado, los futuros profesionales de la ingeniería deben ser capaces no sólo de obtener resultados por medio de software comercial, sino

de conocer los detalles de la formulación matemática y las herramientas numéricas de cálculo que utilizan estos códigos. Los estudiantes deben tener la oportunidad de acceder al código fuente, lo que no es posible en los programas comerciales, que actúan como cajas negras. O, mejor aún, deben ser capaces de desarrollar sencillos códigos de elementos finitos que sirvan de nexo de unión entre la formulación matemática de las clases teóricas y la aplicación práctica. Para ello, el software matemático MATLAB es la herramienta más apropiada.

MATLAB es un programa de cálculo numérico orientado a matrices con un lenguaje de programación propio (lenguaje M). Su capacidad para manipular matrices y resolver ecuaciones matriciales hace del mismo un instrumento idóneo para la implementación y desarrollo de un código de elementos finitos. MATLAB se emplea en más de 5000 universidades a lo largo del mundo y su creciente popularidad y versatilidad hacen que sea utilizado en numerosas asignaturas de las carreras de ciencia e ingeniería. Esta circunstancia facilita en gran medida la tarea docente, ya que los alumnos se han familiarizado previamente con el software y su lenguaje de programación.

En los últimos años se han publicado numerosos trabajos de investigación que emplean el paquete de software MATLAB para el análisis por elementos finitos (Alberty, Carstensen, Funken, & Klose, 2002) y el desarrollo de mejoras en las prestaciones del programa con el objetivo de reducir el tiempo de cálculo mediante el MEF ha sido una constante en la comunidad científica (Rahman, & Valdman, 2013). Existen varios libros publicados dedicados exclusivamente al desarrollo de códigos de elementos finitos en el software matemático MATLAB (Kwon, & Bang, 1996; Kattan, 2007; Ferreira, 2009; Baaser, 2010), cuyas características se han tenido en consideración en la elaboración del presente código, aunque con el objetivo de adecuar el mismo a la didáctica del método, éste difiere significativamente de todos los trabajos citados.

En el presente artículo el código de elementos finitos desarrollado se emplea para resolver mediante diferentes perspectivas un sencillo problema de cálculo estructural que servirá para ilustrar las capacidades docentes del uso de MATLAB en la enseñanza del MEF. Así, por medio de elementos finitos lineales y cuadráticos se obtienen los campos de tensiones y desplazamientos de una barra a tracción de sección constante empotrada en un extremo que está sometida a una carga uniformemente distribuida. El código se ha implementado en la versión 8 de MATLAB (R2012b) y para facilitar una mejor comprensión y reutilización del mismo se muestran íntegramente en los apéndices los códigos correspondientes a cada enfoque y se citan y detallan a continuación los aspectos más relevantes de los mismos. A medida que se resuelve el problema mediante diferentes planteamientos se van introduciendo de forma progresiva diferentes características intrínsecas al método, de manera que el código no está planteado como un instrumento para resolver problemas estructurales reales mediante el MEF sino como una herramienta docente del mismo. A diferencia del primer ejemplo, que ha sido implementado en versiones anteriores de MATLAB con éxito, en el segundo caso es necesario utilizar una versión actualizada del programa, con el fin de evitar el error de software que se produce

en el comando MuPAD intrínseco a las versiones anteriores de MATLAB. El uso de MATLAB como herramienta de enseñanza requiere, como es obvio, de un conocimiento previo del manejo del programa por parte de los estudiantes. Sin embargo, esto no suele suponer un problema, ya que la utilización de MATLAB es habitual en las clases prácticas de las asignaturas relacionadas con el cálculo numérico de los primeros cursos de las carreras técnicas. Y en cualquier caso, para abordar plenamente los problemas que aquí se plantean sólo sería necesario adquirir unas nociones básicas de su uso, que podrían instruirse sin dificultades en una clase introductoria.

3. Planteamiento del problema

La configuración analizada se muestra en la Figura 1. La barra tiene una longitud L de 4 metros, está empotrada en el extremo x=0 y está sometida a una carga distribuida de s = 1000 N/m a lo largo del eje x. La sección transversal es constante con una superficie de A=0.5 m^2 y el módulo elástico tiene un valor de E = 5 MPa.

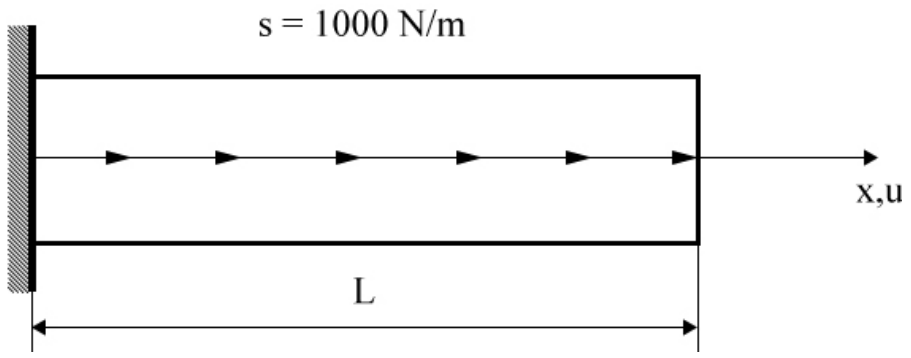

Figura 1. Barra de sección constante empotrada en un extremo sometido a una carga uniformemente distribuida.

3.1. Ejemplo 1: Elementos lineales

En el primer planteamiento del problema la barra se discretiza mediante cuatro elementos lineales de tipo barra con el objetivo de exponer el ensamblaje matricial característico del método y los campos de desplazamientos y tensiones calculados mediante el MEF se comparan a posteriori con la solución analítica. Este sencillo ejemplo permitirá al alumno comprender la formulación matricial del MEF y familiarizarse con las etapas de pre-proceso, proceso y post-proceso intrínsecas al método y al proceder del software comercial. La versión completa del código con comentarios se puede encontrar en el Apéndice 1. Es preciso comentar alguno de los aspectos relacionados con la programación que no han sido suficientemente detallados en el código. En la matriz *Nodos_elemento* se definen las conexiones entre elementos. Cada fila se corresponde con un

elemento y en cada columna se especifican los nodos que forman parte del mismo (línea 23, Apéndice 1).

En el vector *Coord_nodos* se almacenan las coordenadas de cada nodo en el espacio. En este caso se sitúa el origen del eje x en el empotramiento de la barra y se asigna a cada elemento una longitud de un metro (línea 31, Apéndice 1). Se definen el vector de desplazamientos (*Desplazamientos*), el vector de fuerzas (*Fuerza*) y la matriz de rigidez (*Rigidez*). Asimismo se inicializan todos ellos mediante la función *zeros* de Matlab, lo que permite agilizar el proceso de cálculo del programa en los bucles (líneas 40-42, Apéndice 1).

Para facilitar las operaciones entre matrices es conveniente definir un vector con los grados de libertad prescritos en el sistema global (*GDL_prescritos*), que a posteriori se descuentan del número total de grados de libertad del sistema para obtener un vector que almacene los grados de libertad activos en el mismo (*GDL_libres*). Esta operación se lleva a cabo mediante la función de MATLAB *setdiff*, una herramienta especialmente apropiada para la diferencia de vectores (líneas 50-54, Apéndice 1).

Se introduce la carga distribuida sobre la barra en el sistema en base al vector de fuerzas nodales equivalentes. Para ello es necesario programar un bucle del tipo *for/end* que distribuya el efecto de la carga en los nodos, quedando ésta almacenada en el vector de fuerzas (líneas 60-67, Apéndice 1). Una vez discretizado el problema e impuestas las condiciones de contorno finaliza la parte correspondiente al pre-procesador y se inicia la etapa de cálculo o procesador, donde se obtienen la matriz de rigidez, los desplazamientos nodales y las tensiones en cada elemento. Por medio de un bucle del tipo *for/end* se calculan la matrices de rigidez de cada elemento y se ensamblan en la matriz de rigidez global. Para obtener la matriz de rigidez de cada elemento es necesario almacenar los grados de libertad (*GDL_elemento*) y la longitud (*Lon*) del mismo (líneas 75-83, Apéndice 1). La matriz de rigidez de un elemento lineal de tipo barra es función únicamente de la geometría de la misma (*Lon, A*) y de sus propiedades mecánicas (*E*). De manera que en el presente ejemplo, donde el módulo de Young y la sección de la barra son constantes, la matriz de rigidez de cada elemento se corresponde con la siguiente expresión:

$$K^{(e)} = \left(\frac{EA}{Lon}\right)^{(e)} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \qquad (1)$$

Las matrices de rigidez de cada elemento, que han sido calculadas según (1) y que tienen un tamaño 2x2, se distribuyen para formar la matriz de rigidez global de acuerdo con el vector *GDL_elemento*. El proceso de ensamblaje se lleva a cabo mediante la siguiente línea de código: (líneas 80-81, Apéndice 1)

Rigidez(GDL_elemento,GDL_elemento)=...

Rigidez(GDL_elemento,GDL_elemento)+EA(e)[1 -1;-1 1];*

De manera que para este caso, donde la barra se ha discretizado mediante cuatro elementos finitos, la matriz de rigidez global toma la siguiente forma:

$$K=\begin{bmatrix} \left(\frac{EA}{Lon}\right)^{(1)} & -\left(\frac{EA}{Lon}\right)^{(1)} & 0 & 0 & 0 \\ -\left(\frac{EA}{Lon}\right)^{(1)} & \left[\left(\frac{EA}{Lon}\right)^{(1)}+\left(\frac{EA}{Lon}\right)^{(2)}\right] & -\left(\frac{EA}{Lon}\right)^{(2)} & 0 & 0 \\ 0 & -\left(\frac{EA}{Lon}\right)^{(2)} & \left[\left(\frac{EA}{Lon}\right)^{(2)}+\left(\frac{EA}{Lon}\right)^{(3)}\right] & -\left(\frac{EA}{Lon}\right)^{(3)} & 0 \\ 0 & 0 & -\left(\frac{EA}{Lon}\right)^{(3)} & \left[\left(\frac{EA}{Lon}\right)^{(3)}+\left(\frac{EA}{Lon}\right)^{(4)}\right] & -\left(\frac{EA}{Lon}\right)^{(4)} \\ 0 & 0 & 0 & -\left(\frac{EA}{Lon}\right)^{(4)} & \left(\frac{EA}{Lon}\right)^{(4)} \end{bmatrix}$$

Una vez calculada la matriz de rigidez global es posible resolver el sistema global de ecuaciones intrínseco al método de los elementos finitos: $K \cdot f = a$. Siendo K la matriz de rigidez global, f el vector de fuerzas global y a los desplazamientos nodales.

Si consideramos un sistema lineal tal que $A \cdot X = B$, el vector solución X puede ser obtenido en MATLAB empleando la barra inversa (\): X=A\B. En consecuencia, se aplica el mismo sistema para obtener los desplazamientos en cada nodo de la barra (líneas 87-89, Apéndice 1). A diferencia del resto de operaciones del código, en la obtención de los desplazamientos no se ha añadido un punto y coma (;) al final de la instrucción, éste se emplea para indicar a MATLAB que realice el cálculo sin presentar en pantalla el procedimiento o el resultado. De manera que se ordena al programa que muestre en pantalla los desplazamientos obtenidos en cada nodo. A partir de los valores de los desplazamientos nodales es posible calcular otros parámetros de interés, como las tensiones en cada elemento, que serán almacenadas en el vector *Tensiones*. Éstas se calculan a partir de las deformaciones, que a su vez se calculan a partir de los desplazamientos nodales (líneas 93-101, Apéndice 1).

Una vez que se ha obtenido toda la información que se necesitaba a partir de los desplazamientos nodales finaliza la etapa de cálculo o procesador y da paso a la etapa de visualización o post-procesador, donde se muestran con detalle los resultados obtenidos. Esta fase es muy importante cuando se resuelven geometrías complejas y el software comercial de elementos finitos incluye poderosas herramientas para visualizar los resultados obtenidos. En el presente ejemplo, al ser una sencilla configuración unidimensional, la solución analítica es conocida y en consecuencia existe la posibilidad de establecer comparaciones con la misma. Así, mediante la herramienta *plot* se representan gráficamente los desplazamientos calculados mediante el MEF y sobre la misma figura se sobreimprime la solución analítica. Es preciso utilizar el comando *hold on* para que la nueva representación gráfica se realice sobre la misma figura (líneas 111-122, Apéndice 1). De igual forma, se representan gráficamente en una nueva figura los campos tensionales obtenidos analíticamente y por medio del MEF (líneas 126-145, Apéndice 1).

Las gráficas obtenidas para el campo de desplazamientos y el campo de tensiones se muestran en las figuras 2 y 3 respectivamente. Los resultados se han representado de forma sencilla y funcional con el objetivo de eliminar obstáculos en la docencia, pero MATLAB ofrece un amplio abanico de herramientas para mejorar y complementar la estética y el detalle de las gráficas. Como se puede apreciar, se representan los valores obtenidos para los desplazamientos (Figura 2) y las tensiones (Figura 3) a partir de los datos del problema planteado en función de la posición en la barra. La solución obtenida mediante el MEF se reproduce mediante una línea roja discontinua, donde la posición de los nodos viene señalizada por pequeñas circunferencias del mismo color. Mientras que la solución exacta viene representada por una línea continua de color azul, tal y como se describe en la leyenda de ambas imágenes.

En la Figura 2 se observa que los desplazamientos nodales coinciden con los valores exactos y que, de acuerdo con la elección de elementos finitos del tipo lineal, en el interior de los mismos los desplazamientos varían linealmente aproximando razonablemente bien la solución analítica.

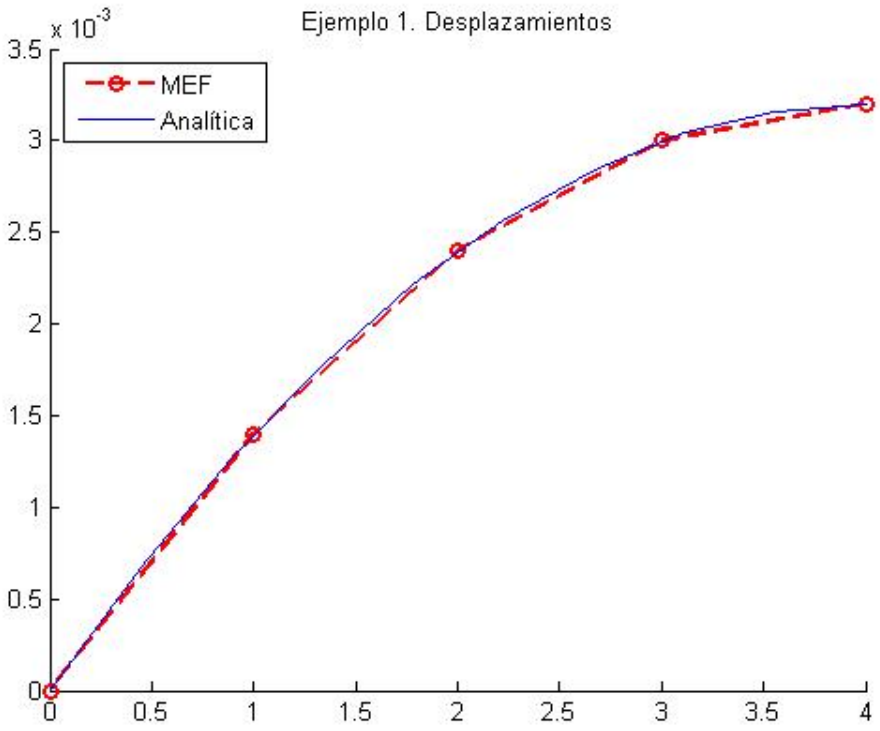

Figura 2. Desplazamientos obtenidos en función de la posición en el Ejemplo 1.

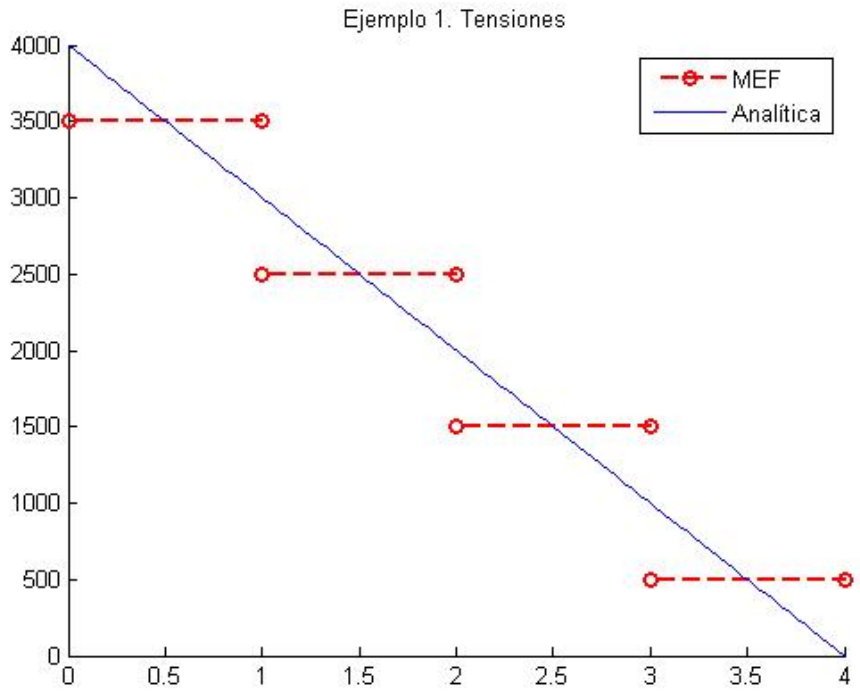

Figura 3. Tensiones obtenidas en función de la posición en el Ejemplo 1.

El alumno podrá apreciar cómo la compatibilidad se satisface siempre en los nodos y en consecuencia existe un campo de desplazamientos continuo. Sin embargo, en la aproximación del campo tensional de la Figura 3 se aprecian mayores diferencias entre la solución exacta y la obtenida mediante el MEF y por consiguiente es conveniente discretizar la barra con un mayor número de elementos para reducir el error en el cálculo del esfuerzo axil. El alumno podrá observar que en el MEF, generalmente, no existe equilibrio de tensiones en los nodos. Y es que las tensiones en los nodos se obtienen en cada elemento a partir de los valores de los desplazamientos en sus nodos y en consecuencia, las tensiones en un nodo común a varios elementos pueden tomar valores diferentes. De manera que a partir del presente ejemplo el alumno rápidamente podrá deducir que, al obtenerse las deformaciones y tensiones a partir de las derivadas del campo de desplazamientos, los errores en la aproximación de los campos de deformaciones y tensiones serán siempre superiores.

Además, el presente ejemplo permite modificar el mallado del problema con facilidad, requiriendo únicamente modificar las variables *Coord_nodos*, *Nodos_elemento* y *L*. Lo que permitirá al alumno obtener la solución mediante el MEF para diferentes discretizaciones, observando como la solución obtenida se aproxima cada vez más a la solución exacta a medida que aumenta el número de elementos empleado. Pero sobre todo, el presente ejercicio permitirá al alumno familiarizarse con las etapas básicas del MEF. Esto es:

- Discretizar el problema en una serie de elementos finitos conectados entre sí en los nodos (mallado).
- Calcular la matriz de rigidez $\left(K^{(e)}\right)$ y el vector de fuerzas nodales $\left(f^{(e)}\right)$ para cada elemento del sistema.
- Ensamblar y resolver la ecuación matricial de equilibrio global $(K \cdot f = a)$ para, una vez impuestas las condiciones de contorno, calcular los valores de los desplazamientos en los nodos.
- Calcular los parámetros que resulten de interés en cada caso a partir de los desplazamientos nodales obtenidos.
- Visualizar los resultados con claridad para tomar decisiones acertadas con respecto al diseño del componente analizado.

Una realización satisfactoria del ejemplo por parte del alumno le ayudará a comprender la estructura básica del método y esto le permitirá desarrollar el código para dar respuesta a problemas más complicados. En el segundo enfoque de este problema se introduce otro tipo de elemento finito a la par que se abordan dos aspectos claves del método que no se han considerado en el primer caso: la formulación isoparamétrica y la integración numérica de Gauss-Legendre.

3.2. Elementos cuadráticos

En este segundo enfoque la barra se discretiza mediante un solo elemento cuadrático de tipo barra con el objetivo de reflejar la importancia de una elección apropiada del grado de distribución de los desplazamientos en los elementos mediante el uso de diferentes funciones de interpolación o forma. Asimismo, este ejemplo permitirá al alumno familiarizarse con las transformaciones isoparamétricas intrínsecas al MEF y comprender el uso de la integración numérica más común en los análisis por elementos finitos: la cuadratura de Gauss-Legendre. Al igual que en el caso anterior, los campos de desplazamientos y tensiones calculados mediante el MEF se comparan con la conocida solución analítica.

La estructura del código apenas difiere de la concerniente al primer ejemplo, con la intención de que el programa pueda ser empleado, con la introducción de pequeños cambios, como herramienta docente para la resolución de una amplia variedad de ejercicios. A pesar de que las características del código se han comentado convenientemente en su interior, es preciso explicar con detalle algunas partes del mismo que no son comunes al primer ejemplo y sobre las que es fundamental hacer especial énfasis. En lo que a la etapa del pre-proceso se refiere, el único aspecto que difiere significativamente radica en la introducción en el vector de fuerzas del efecto de la carga distribuida sobre la barra. De acuerdo con los conceptos de equilibrio del Principio de los Trabajos Virtuales (PTV) se obtiene que, en un elemento

aislado, cualquier fuerza T que actúe sobre un contorno Γ se puede expresar tal que:

$$F = \int_{\Gamma^e} N^T \cdot T \cdot d\Gamma^e \qquad (2)$$

Siendo N^T la transformada de la matriz de funciones de forma. Las funciones de forma son unas funciones de interpolación que toman el valor 1 en el nodo de referencia y 0 en el resto, de manera que las variables a calcular se pueden obtener como la suma de los productos de estas funciones de interpolación o forma por los valores de la función en los nodos. Así, por medio de la interpolación, la distribución de la variable a calcular (por ejemplo, el desplazamiento) queda definida en todo el dominio mientras que el problema se ha reducido a un número finito de grados de libertad. Las funciones de forma conocidas se corresponden con formas geométricas muy sencillas, pero en la práctica la geometría de los elementos en los que se divide el dominio es más compleja. Para transformar estos perfiles complejos en las formas simples conocidas se emplea la formulación isoparamétrica. Con el objetivo de no alterar en exceso la base del código y para resaltar su relevancia, la transformación isoparamétrica del presente ejemplo se lleva a cabo en una función de MATLAB que se ha programado por separado y que recibe el nombre de *TransIso*. Esta función recibe la posición de los nodos en el espacio (*Coord_nodos*) y entrega al código principal el vector de funciones de forma (*Fforma*), el determinante (*detJacobiano*) y el inverso (*invJacobiano*) del Jacobiano y la coordenada natural (xi) correspondiente a la coordenada cartesiana x. Lo que nos permite obtener el vector de Fuerzas a partir de la expresión (2):

$$F = \int_{\Gamma^e} N^T \cdot T \cdot d\Gamma^e = \int_L N^T \cdot s \cdot dx = \int_{-1}^{+1} \begin{bmatrix} 0.5 \cdot \xi \cdot (\xi - 1) \\ 1 - \xi^2 \\ 0.5 \cdot \xi \cdot (\xi + 1) \end{bmatrix} \cdot s \cdot |J| d\xi \qquad (3)$$

Siendo s el valor de la carga distribuida, ξ la coordenada natural y $|J|$ el determinante del Jacobiano. De manera que el vector de fuerzas nodales equivalentes se calcula en el código directamente de acuerdo con (3) y se almacena, al igual que en el primer ejemplo, en el vector *Fuerza*. La integral se resuelve directamente por medio del operador *int* de MATLAB (líneas 62-73, Apéndice 2). La transformación isoparamétrica se lleva a cabo en la función *TransIso* (líneas 1-18, Apéndice 3). Y, al igual que en el código principal, es preciso hacer hincapié en algunos aspectos de la misma. La transformación isoparamétrica que se pretende llevar a cabo se describe en la Figura 4. De acuerdo con la notación empleada en la misma, las funciones de forma en el caso de un elemento cuadrático unidimensional, que han sido obtenidas mediante interpolaciones de Lagrange, son las siguientes:

$$N_1(\xi) = \frac{1}{2}\xi(\xi - 1) \quad ; \quad N_2(\xi) = (1 - \xi)(1 + \xi) \quad ; \quad N_3(\xi) = \frac{1}{2}\xi(\xi + 1)$$

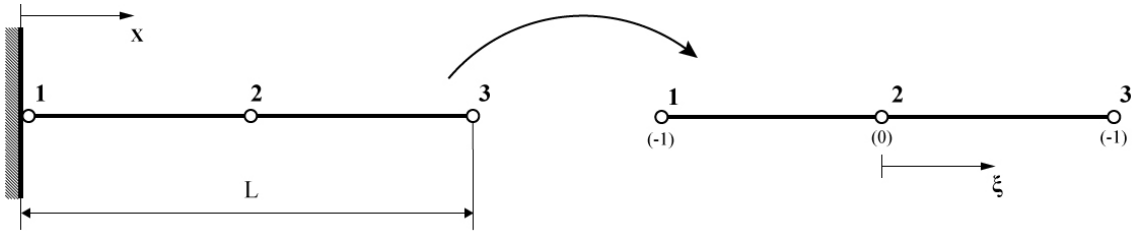

Figura 4. Transformación isoparamétrica para el problema planteado en el Ejemplo 2.

De manera que, definiendo *xi* como variable simbólica por medio del operador *syms*, las funciones de forma se almacenan en el vector *Fforma*. Y, de acuerdo con la formulación isoparamétrica, la matriz Jacobiano que define la transformación de coordenadas $x \rightarrow \xi$ en un elemento unidimensional es: $dx = J^{(e)} \cdot d\xi$. Y así, tal y como se procede en la función, el determinante del Jacobiano (*detJacobiano*) se obtiene fácilmente despejando en la expresión (4). Las derivadas presentes en la misma se resuelven por medio del comando de MATLAB *diff*.

$$x = N_1 \cdot x_1 + N_2 \cdot x_2 + N_3 \cdot x_3 \Rightarrow \frac{dx}{d\xi} = \frac{dN_1}{d\xi} \cdot x_1 + \frac{dN_2}{d\xi} \cdot x_2 + \frac{dN_3}{d\xi} \cdot x_3 \qquad (4)$$

Siguiendo con el análisis del código, se da paso a la etapa del procesador, donde se calculan la matriz de rigidez, los desplazamientos en los nodos y las tensiones en el elemento. Para no modificar en exceso el esqueleto del programa, la matriz de rigidez (*Rigidez*) se obtiene, al igual que en el primer ejemplo, por medio de un bucle del tipo *for/end*. Sin embargo en esta ocasión su uso no es necesario, ya que no hace falta realizar el proceso de ensamblaje al haber discretizado el dominio con un solo elemento. De acuerdo con las relaciones de equilibrio del PTV la matriz de rigidez de un elemento en un volumen Ω se puede obtener en base a:

$$K^{(e)} = \int_{\Omega^e} B^T \cdot D \cdot B \cdot d\Omega^e \qquad (5)$$

Donde K es la matriz de rigidez, D la matriz constitutiva y B la matriz de deformación del elemento. En el presente caso, aplicando la transformación isoparamétrica y teniendo en cuenta que el área y el módulo de Young son constantes en todo el elemento, la expresión (5) quedaría tal que:

$$K^{(e)} = \int_{\Omega^e} B^T \cdot D \cdot B \cdot d\Omega^e = EA \int_L B^T \cdot B \cdot dx = EA \int_{-1}^{1} B^T \cdot B \cdot |J| d\xi \qquad (6)$$

La matriz de deformación del elemento relaciona el vector deformación con el vector de desplazamientos nodales, y en consecuencia, para un elemento de tres nodos se corresponde con la siguiente expresión:

$$B = \left[\frac{dN_1}{dx}, \frac{dN_2}{dx}, \frac{dN_3}{dx}\right] \quad (7)$$

De manera que aplicando la transformación isoparamétrica quedaría tal que:

$$B = \left[\frac{dN_1}{dx}, \frac{dN_2}{dx}, \frac{dN_3}{dx}\right] = \left[\frac{dN_1}{d\xi}, \frac{dN_2}{d\xi}, \frac{dN_3}{d\xi}\right] \cdot \frac{d\xi}{dx} = \left[\frac{dN_1}{d\xi}, \frac{dN_2}{d\xi}, \frac{dN_3}{d\xi}\right] \cdot \frac{1}{|J|} \quad (8)$$

Sustituyendo (8) en (6) y operando se obtiene la siguiente expresión para la matriz de rigidez:

$$K^{(e)} = EA \int_{-1}^{1} B^T \cdot B \cdot |J| d\xi = EA \cdot \frac{L}{2} \int_{-1}^{1} \begin{bmatrix} \xi - 0.5 \\ -2\xi \\ \xi + 0.5 \end{bmatrix} \cdot [\xi - 0.5 \quad -2\xi \quad \xi + 0.5] d\xi \quad (9)$$

En este sencillo ejemplo, el producto matricial en el interior de la integral trae consigo expresiones polinómicas de segundo orden en función de ξ que no son difíciles de integrar. Sin embargo, en problemas de interés práctico será necesario el cálculo de integrales mucho más complejas en las que será imprescindible el uso de la integración numérica. En el MEF es especialmente popular la integración numérica de Gauss-Legendre y, por su singularidad, ésta se lleva a cabo en el presente ejemplo en una función programada en MATLAB externa al programa principal. Esta función, que recibe el nombre de *IntGauss*, recibe la variable de la coordenada natural (*xi*) y la matriz resultante del producto matricial en el interior de la integral de la matriz de rigidez (*R*), y devuelve la matriz resultante del proceso de integración (*H*), lo que permite resolver (9) para cada elemento y obtener la matriz de rigidez global (líneas 81-91 Apéndice 2 y líneas 1-11, Apéndice 4).

Aunque la función es sencilla, dada la relevancia de la integración Gaussiana en el MEF es preciso examinar en detalle sus características. Si se realiza el producto matricial del integrando en (9) queda:

$$K^{(e)} = EA \cdot \frac{L}{2} \int_{-1}^{1} \cdot \begin{bmatrix} (\xi - 0.5)^2 & -2\xi(\xi - 0.5) & (\xi - 0.5)(\xi + 0.5) \\ -2\xi(\xi - 0.5) & 4\xi^2 & -2\xi(\xi + 0.5) \\ (\xi - 0.5)(\xi + 0.5) & -2\xi(\xi + 0.5) & (0.5 + \xi)^2 \end{bmatrix} d\xi \quad (10)$$

La regla de integración o cuadratura de Gauss-Legendre expresa el valor de dicha integral como suma de los productos de los valores del integrando en una serie de puntos conocidos en el interior del intervalo (puntos de integración de Gauss) por unos coeficientes determinados (pesos). Habida cuenta de que la matriz de (10) contiene polinomios de segundo grado y teniendo en consideración que una cuadratura de Gauss-Legendre de orden n integra exactamente un polinomio de grado 2n-1 o menor, es obvio que será necesario recurrir a una cuadratura de orden 2 para integrar de forma exacta las

expresiones del presente ejemplo. En la Tabla 1 se muestran las coordenadas y los pesos para elementos lineales.

n	$\pm\xi_i$	W_i
1	0	2
2	$1/\sqrt{3}$	1
3	$\sqrt{3/5}$ 0	$5/9$ $8/9$

Tabla 1. Coordenadas naturales ξ_i y factores de peso W_i en elementos unidimensionales

De manera que el elemento cuadrático con el que se ha discretizado la barra tiene 2 puntos de integración de Gauss ubicados en las coordenadas $\xi_i = \pm 1/\sqrt{3}$. Los desplazamientos nodales se obtienen resolviendo el sistema $K \cdot f = a$, tal y como se ha detallado en el caso anterior. Sin embargo, en esta ocasión se ha discretizado el dominio mediante un elemento finito cuadrático y por consiguiente, los desplazamientos en el interior del mismo no varían de forma lineal entre los valores obtenidos en los nodos, tal y como sucedía en el primer ejemplo, sino que la distribución de desplazamientos viene caracterizada por las funciones de forma cuadráticas empleadas. De manera que el campo de desplazamientos (u) en el interior del elemento se calcula en base a: (líneas 101-102, Apéndice 2)

*u=Desplazamientos(1)*Fforma(1)+Desplazamientos(2)*Fforma(2)+...*

*Desplazamientos(3)*Fforma(3);*

Y el campo tensional (*n*), al igual que en el caso anterior, se obtiene a partir de las deformaciones, que están directamente relacionadas con los desplazamientos por medio de la matriz de deformación (*B*) (línea 106, Apéndice 2). Una vez resuelto el sistema matricial y obtenidos, a partir de los desplazamientos, todos los parámetros de interés, es importante reflejar convenientemente los resultados, en lo que se denomina la etapa de post-procesado. En este caso, a diferencia del primer ejemplo, al haber utilizado la transformación isoparamétrica será necesario realizar de nuevo el cambio de variable para reflejar los resultados de las variables de interés sobre el sistema de coordenadas cartesiano (líneas 117-126, Apéndice 2). El cambio de variable en la expresión de los desplazamientos (*u*) se lleva a cabo mediante el operador *subs* teniendo en consideración la relación existente entre la coordenada natural y la coordenada cartesiana (ver Figura 4):

$$\xi = 2\frac{x - x_c}{l^{(e)}} \qquad (11)$$

Siendo x_c la coordenada cartesiana en el centro del elemento. Para representar gráficamente una función se emplea el comando *ezplot*. Se procede de idéntica forma para reproducir el campo tensional calculado. Las gráficas obtenidas con MATLAB se muestran en las Figuras 5 y 6. En las mismas se comparan, por medio de la misma señalización que la empleada en el primer ejemplo, los desplazamientos y las tensiones calculados mediante el MEF con la solución analítica.

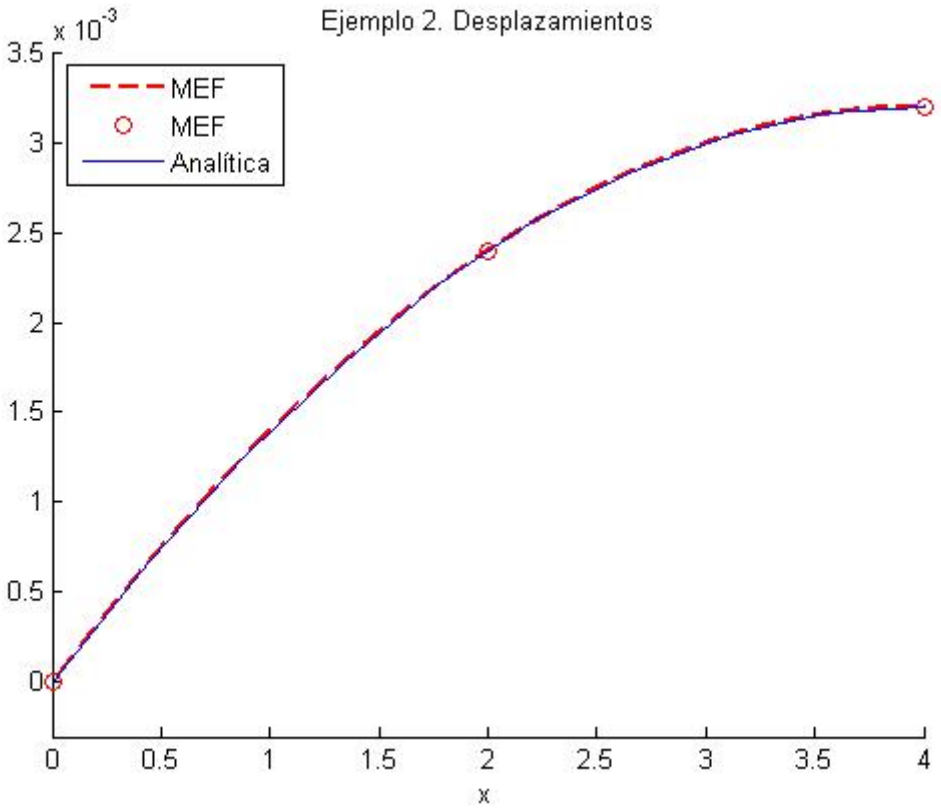

Figura 5. Desplazamientos obtenidos en función de la posición en el Ejemplo 2.

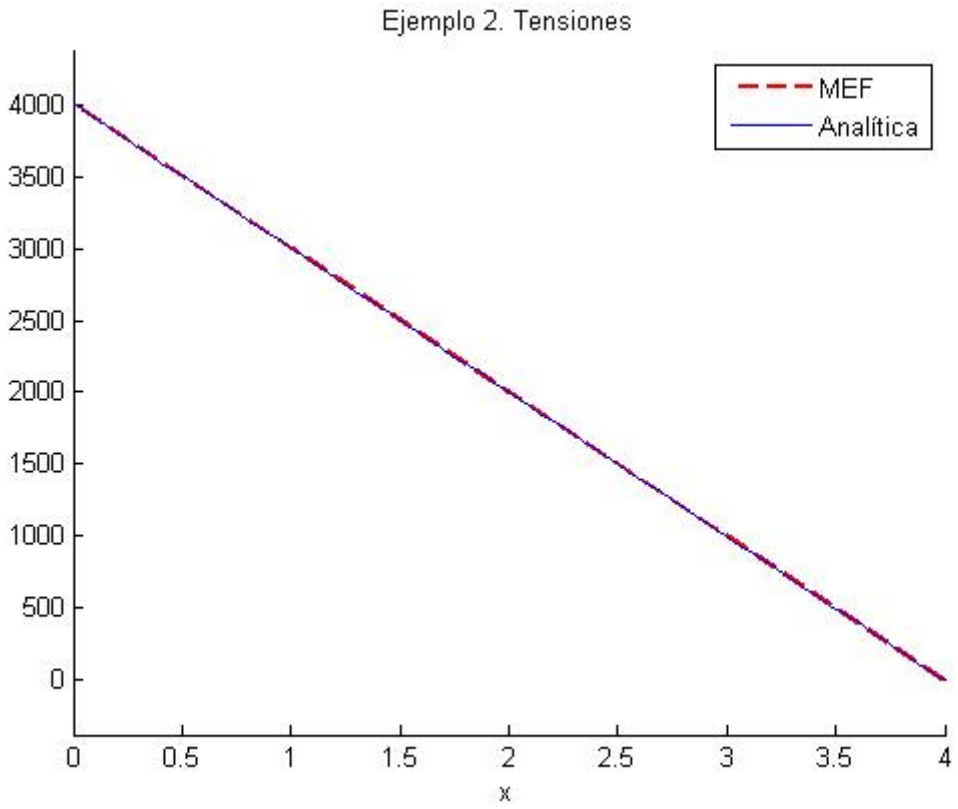

Figura 6. Tensiones obtenidas en función de la posición en el Ejemplo 2.

En la Figura 5 se observa que tanto los valores del desplazamiento en los nodos como el campo de desplazamientos en el interior del elemento coinciden con la solución exacta. Este resultado era previsible, pues se está aproximando una solución polinómica de segundo orden mediante un elemento finito cuadrático. De igual manera, en la Figura 6 se puede apreciar como el campo tensional en el interior del elemento reproduce fielmente la solución analítica. Con lo que en este segundo ejemplo no tendría sentido discretizar el dominio mediante un número mayor de elementos, ya que la solución obtenida mediante el MEF coincide con la exacta. Este hecho posibilita que el alumno fácilmente alcance la conclusión de que, en el caso de que el orden polinomial de la solución sea conocido, resulta conveniente el uso de elementos finitos con funciones de forma del mismo grado que la solución. Ya que esto garantiza no sólo que la solución en los nodos sea correcta, sino que la variación de los desplazamientos en el interior de cada elemento coincida con la exacta. Sin embargo, es necesario advertir a los estudiantes de que esta coyuntura rara vez se encuentra en un caso de aplicación práctica.

Observando las soluciones obtenidas para el primer (Figuras 2 y 3) y segundo (Figuras 5 y 6) ejemplo es posible establecer una comparativa entre los resultados obtenidos para diferentes tipos de elementos. De manera que el

alumno rápidamente puede deducir que, para el problema evaluado en el presente trabajo, la precisión del elemento cuadrático es superior a la del elemento lineal. Ésta es una conclusión acertada, puesto que cuanto más simple es el elemento, menos capacidad tiene de aproximar soluciones en las que el campo de desplazamientos sea una función polinómica de un orden alto. Sin embargo, el uso de elementos finitos de menor orden permite ahorrar tiempo en el cálculo de las matrices del elemento. De manera que, en líneas generales y desde el punto de vista de la eficiencia computacional, es recomendable el uso de elementos finitos con menos nodos, aunque sea en mayor número, frente a elementos de orden superior. Y es que, dado el vertiginoso avance de la tecnología informática en los últimos años, los problemas asociados a una mayor cantidad de variables son cada vez menos importantes. El uso de elementos de orden superior se recomienda únicamente en aquellas regiones en las que se pueda intuir que la variable a calcular varía significativamente. En cualquier caso, en la mayoría de las ocasiones será la experiencia del calculista la que determine el tipo de elemento a utilizar.

4. Conclusiones

Por medio del software matemático MATLAB se ha desarrollado un inteligible código de elementos finitos capaz de resolver sencillos problemas estructurales abordando gradualmente las características más relevantes del MEF. El mismo se ha utilizado para obtener, mediante los dos tipos de elementos finitos más comunes, los campos de desplazamientos y tensiones en una barra a tracción de sección constante empotrada en un extremo que está sometida a una carga uniformemente distribuida. El problema resuelto, el orden de los ejemplos y la secuencia de operaciones desarrolladas en el proceso de resolución de los mismos obedecen exclusivamente a fines puramente didácticos, con el objetivo de que el alumno se enfrente de forma progresiva a las particularidades inherentes al MEF.

La estructura del código se ha desarrollado con el objetivo de que pueda ser extendido, sin necesidad de realizar grandes modificaciones, para dar respuesta a problemas más complejos, ya sea por la dimensión de los mismos (2D, 3D) o por el tipo de elemento finito a utilizar (viga, placa, etc.). En cualquier caso, los ejemplos reflejados en el presente artículo han demostrado ser más que suficientes para introducir el MEF a alumnos de ingeniería, sirviendo de enlace entre la formulación matemática del método y el proceder del software comercial. Por su sencillez de uso y programación, el software matemático MATLAB ha evidenciado ser una herramienta didáctica muy valiosa a la hora de trasmitir a los alumnos los fundamentos básicos del MEF.

A1. Apéndice 1: Código correspondiente al 1er ejemplo – elementos lineales

```
%.....................................................................

% MATLAB: Una herramienta para la didáctica del método de los
% elementos finitos.
% Ejemplo1

% Se resetea la memoria del programa y se limpia el espacio de trabajo

clear all

% Se definen las propiedades del material:
% E: módulo elástico.
% A: área de la sección transversal.

E=5000000;
A=0.5;

% Pre-procesador: Se discretiza el problema y se imponen las
% condiciones de contorno.

% Se asignan los nodos correspondientes a cada elemento:

Nodos_elemento=[1 2;2 3;3 4;4 5];

% Se define el número de elementos:

Numero_elementos=size(Nodos_elemento,1);

% Se posicionan los nodos en el espacio:

Coord_nodos=[0 1 2 3 4];

% Se define el número de nodos:

Numero_nodos=size(Coord_nodos,2);

% Se definen e inicializan el vector de desplazamientos, el vector de
% fuerzas y la matriz de rigidez:

Desplazamientos=zeros(Numero_nodos,1);
Fuerza=zeros(Numero_nodos,1);
Rigidez=zeros(Numero_nodos);

% Se introducen las condiciones de contorno.

% De acuerdo a las condiciones del problema, se restringen los grados
```

```
% de libertad (GDL) necesarios, en este caso están restringidos los
% desplazamientos en el primer nodo.

GDL_prescritos=[1];

% Se designan los GDL libres (GDL_libres).

GDL_libres=setdiff([1:Numero_nodos]',[GDL_prescritos]);

% Se introduce la carga aplicada.
% L es la longitud de la barra y s=1000 N/m es la carga distribuida
% del presente ejemplo.

s=1000; L=Coord_nodos(5)-Coord_nodos(1);
b=s*L/Numero_elementos;

for j=1:Numero_nodos

Fuerza(j)=b*sum(sum(Nodos_elemento == j))/size(Nodos_elemento,2);

end

% Procesador: Se calcula la matriz de rigidez, los desplazamientos en
% los nodos y las tensiones en cada elemento.

% Se calcula la matriz de rigidez (Rigidez), siendo GDL_elemento los
% grados de libertad y Lon la longitud de cada elemento.

for e=1:Numero_elementos

GDL_elemento=Nodos_elemento(e,:);
Lon=Coord_nodos(GDL_elemento(2))-Coord_nodos(GDL_elemento(1));
EA(e)=E*A/Lon;
Rigidez(GDL_elemento,GDL_elemento)=...
Rigidez(GDL_elemento,GDL_elemento)+EA(e)*[1 -1;-1 1];

end

% Se calculan los desplazamientos:

Desp=Rigidez(GDL_libres,GDL_libres)\Fuerza(GDL_libres);
Desplazamientos=zeros(Numero_nodos,1);
Desplazamientos(GDL_libres)=Desp

% Se calcula el vector de tensiones (Tensiones):

for k=1:Numero_elementos

Dif_desp=Desplazamientos(Nodos_elemento(k,2))-...
Desplazamientos(Nodos_elemento(k,1));
Lon=Coord_nodos(Nodos_elemento(k,2))-...
Coord_nodos(Nodos_elemento(k,1));
Tensiones(k)=E*A*Dif_desp/Lon;

end

```

```
% Postprocesador: se muestran los resultados obtenidos.
% En el presente trabajo se conoce la solución analítica, por lo que
% los resultados obtenidos mediante el MEF se comparan con la misma:

% Desplazamientos

% Solución mediante el MEF

plot(Coord_nodos,Desplazamientos,'Marker','o','LineWidth',1.5,...
'LineStyle','--','Color',[1 0 0],'DisplayName','MEF');

% Solución analítica

x=linspace(0,L,10);
u=(-s*x.^2/2+s*L*x)/(E*A);
hold on
plot(x,u,'LineWidth',1,'DisplayName','Analítica');
h=legend('MEF','Analítica',2);
set(gca,'box','off')
title('Ejemplo 1. Desplazamientos');

% Tensiones

figure
hold on

% Solución mediante el MEF

f1=zeros(Numero_elementos,1);
for i=1:Numero_elementos
 p=Nodos_elemento(i,:);
 f1(i)=plot(Coord_nodos(p),repmat(Tensiones(i),1,...

length(Coord_nodos(p))),'Marker','o','LineWidth',1.5,'LineStyle',...
     '--','Color',[1 0 0],'DisplayName','MEF');
end

% Solución analítica

x=linspace(0,L,100);
y=s*(L-x);
f2=plot(x,y,'LineWidth',1,'DisplayName','Analítica');
h=legend([f1(end),f2],'MEF','Analítica');
title('Ejemplo 1. Tensiones');

% Fin Ejemplo1
```

A2. Apéndice 2: Código correspondiente al 2º ejemplo – elementos cuadráticos

```
%.....................................................................

% MATLAB: Una herramienta para la didáctica del método de los
% elementos finitos.
% Ejemplo2

% Se resetea la memoria del programa y se limpia el espacio de trabajo

clear all

% Se definen las propiedades del material:
% E: módulo elástico.
% A: área de la sección transversal.

E=5000000;
A=0.5;

% Pre-procesador: Se discretiza el problema y se imponen las
% condiciones de contorno.

% Se asignan los nodos correspondientes a cada elemento:

Nodos_elemento=[1 2 3];

% Se define el número de elementos:

Numero_elementos=size(Nodos_elemento,1);

% Se posicionan los nodos en el espacio:

Coord_nodos=[0 2 4];

% Se define el número de nodos:

Numero_nodos=size(Coord_nodos,2);

% Se definen e inicializan el vector de desplazamientos, el vector de
% fuerzas y la matriz de rigidez:

Desplazamientos=zeros(Numero_nodos,1);
Fuerza=zeros(Numero_nodos,1);
Rigidez=zeros(Numero_nodos);

% Se introducen las condiciones de contorno.

% De acuerdo a las condiciones del problema, se restringen los grados
% de libertad (GDL) necesarios, en este caso están restringidos los
% desplazamientos en el primer nodo.

GDL_prescritos=[1];
```

```
51
52  % Se designan los GDL libres (GDL_libres).
53
54  GDL_libres=setdiff([1:Numero_nodos]',[GDL_prescritos]);
55
56  % Se introduce la carga aplicada, formulando matricialmente el
57  % problema y haciendo uso de la transformación isoparamétrica.
58
59  % L es la longitud de la barra y s=1000 N/m es la carga distribuida
60  % del presente ejemplo.
61
62  s=1000;L=Coord_nodos(3)-Coord_nodos(1);
63
64  % Se realiza la transformación isoparamétrica por medio de la función
65  % TransIso
66
67  [Fforma,detJacobiano,invJacobiano,xi]=TransIso(Coord_nodos);
68
69  for j=1:Numero_nodos
70
71  Fuerza(j)=s*detJacobiano*int(Fforma(j),xi,-1,1);
72
73  end
74
75  % Procesador: Se calcula la matriz de rigidez, los desplazamientos en
76  % los nodos y las tensiones en cada elemento.
77
78  % Se calcula la matriz de rigidez (Rigidez), siendo GDL_elemento los
79  % grados de libertad de cada elemento.
80
81  for e=1:Numero_elementos
82
83   GDL_elemento=Nodos_elemento(e,:);
84
85  B(xi)=[diff(Fforma(1)),diff(Fforma(2)),diff(Fforma(3))]*invJacobiano;
86   R(xi)=B'*B;
87   [H]=IntGauss(R,xi);
88   EA(e)=E*A;
89   Rigidez(GDL_elemento,GDL_elemento)=...
90   Rigidez(GDL_elemento,GDL_elemento)+EA(e)*H*detJacobiano;
91
92  end
93
94  % Se calculan los desplazamientos:
95
96  Desp=Rigidez(GDL_libres,GDL_libres)\Fuerza(GDL_libres);
97  Desplazamientos=zeros(Numero_nodos,1);
98  Desplazamientos(GDL_libres)=Desp
99
100 % Se obtiene el campo de desplazamientos en el interior del elemento:
101
102 u=Desplazamientos(1)*Fforma(1)+Desplazamientos(2)*Fforma(2)+...
103     Desplazamientos(3)*Fforma(3);
104
105 % Se obtiene el campo tensional en el interior del elemento:
106
```

```
107  n=E*A*B*Desplazamientos;
108
109  % Postprocesador: se muestran los resultados obtenidos.
110  % En el presente trabajo se conoce la solución analítica, por lo que
111  % los resultados obtenidos mediante el MEF se comparan con la misma:
112
113  % Desplazamientos
114
115  % Se obtiene el campo de desplazamientos haciendo el cambio de
116  % variable al sistema cartesiano
117
118  syms x;
119  u=subs(u,xi,(2*(x-L/2)/L));
120
121  % Solución mediante el MEF
122
123  hold on
124  a=ezplot(u,[0,L]);
125  set(a,'LineWidth',1.5,'LineStyle','--','Color', [1 0 0]);
126  plot(Coord_nodos,Desplazamientos,'o','Color', [1 0 0]);
127  title('Ejemplo 2. Desplazamientos');
128
129  % Solución analítica
130
131  y=linspace(0,L,10);
132  u=(-s*y.^2/2+s*L*y)/(E*A);
133  plot(y,u,'LineWidth',1,'DisplayName','Analítica');
134  h=legend('MEF','MEF','Analítica',2);
135
136  % Tensiones
137
138  % Se obtiene el campo de tensiones haciendo el cambio de variable al
139  % sistema cartesiano
140
141  n=subs(n,xi,(2*(x-L/2)/L));
142
143  % Solución mediante el MEF
144
145  figure;
146  hold on;
147  b=ezplot(n,[0,L]);
148  set(b,'LineWidth',1.5,'LineStyle','--','Color', [1 0 0]);
149
150  % Solución analítica
151
152  x=linspace(0,L,100);
153  y=s*(L-x);
154  f2=plot(x,y,'LineWidth',1,'DisplayName','Analítica');
155  h=legend([b(end),f2],'MEF','Analítica');
156  title('Ejemplo 2. Tensiones');
157
158  % Fin Ejemplo
```

A3. Apéndice 3: Código correspondiente a la función *TransIso*

```
1   function [Fforma,detJacobiano,invJacobiano,xi]=TransIso(Coord_nodos)
2
3   % En esta función se lleva a cabo la transformación isoparamétrica
4
5   % Se define el vector de funciones de forma (Fforma), siendo z la
6   % coordenada natural correspondiente a la coordenada cartesiana x.
7
8   syms xi
9
10  Fforma=[0.5*xi*(xi-1);(1-xi)*(1+xi);0.5*xi*(1+xi)];
11
12  % Se calculan el determinante y el inverso del Jacobiano (detJacobiano):
13
14  detJacobiano = diff(Fforma(1))*Coord_nodos(1) + ...
15        diff(Fforma(2))*Coord_nodos(2) + diff(Fforma(3))*Coord_nodos(3);
16
17  invJacobiano=1/detJacobiano
18
```

A4. Apéndice 4: Código correspondiente a la función *IntGauss*

```
1   function [H] = IntGauss(R,xi)
2
3   % En esta función se lleva a cabo la integración Gaussiana.
4
5   % Se integra numéricamente por Gauss-Legendre con precisión
6   % cuadrática, siendo W el peso correspondiente al punto de
7   % integración.
8
9   W=1;
10
11  H=W*R(-1/(sqrt(3)))+W*R(1/(sqrt(3)))
```

UNIÓN
REVISTA IBEROAMERICANA DE EDUCACIÓN MATEMÁTICA
www.fisem.org/web/union

ISSN: 1815-0640
Número XX. Noviembre 2015
Páginas XX-XX

Bibliografía

Alberty, J., Cartensen, C., Funken, S.A., & Klose, R. (2002). MATLAB implementation of the finite element method in elasticity. *Computing*, 69, 239-263.

Baaser, H. (2010). *Development and Application of the Finite Element Method based on MatLab*. Springer.

Backer, J.R., Capece, V.R., & Lee J.R. (2001). Integration of finite element software in undergraduate engineering courses. *ASEE Annual Conference Proceedings*, Session 1520.

Connell, H., Blyth, B., May, R., & Zorzan, C. (1999). Teaching the finite element method using software. *Proceedings of the Delta'99 Symposium on Undergraduate Mathematics*, 65-68.

Earley, R.D. (1998). Use of FEA in an introductory strength of materials course. *ASEE Annual Conference Proceedings*, Session 3648.

Ferreira, A.J.M. (2009). *MATLAB Codes for Finite Element Analysis*. Solid Mechanics and its Applications, Springer.

Howard, W.E., Musto, J.C., & Prantil, V. (2001). Finite element analysis in a mechanics course sequence. *ASEE Annual Conference Proceedings*, Session 2793.

Jiang, Y., & Wang, C. (2008). On teaching finite element method in plasticity with Mathematica. *Computer Applications in Engineering Education*, 16, 233-242.

Jolley, W.O., Rencis J.J., & Grandin H.T. (2003). A module for teaching fundamentals of finite-element theory and practice using elementary mechanics of materials. *ASEE Annual Conference Proceedings*, Session 3268.

Kattan, P.I. (2007). *MATLAB Guide to finite elements, an interactive approach*. Springer, Berlin, 2nd ed.

Kosashi, P.B. (2010). Learning finite element methods by building applications. *International Journal of Mechanical Engineering Education*, 38 (2), 167-184.

Kwon, Y. W., & Bang H. (1996). *Finite element method using MATLAB*. CRC Press, Boca Raton, FL.

Lissenden, C.J., Wagle, G.S., & Salamon, N.J. (2002). Applications of finite element analysis for undergraduates. *ASEE Annual Conference Proceedings*, Session 3568.

Logue, L.J., & Hall, K.A. (2001). Introducing finite element analysis in an MET strength of materials course. *ASEE Annual Conference Proceedings*, Session 3248.

Martínez-Pañeda, E., & Betegón, C., (2015). Modeling damage and fracture within strain- gradient plasticity. *International Journal of Solids and Structures,* 59, 208-215.

Martínez-Pañeda, E., & Gallego, R., (2015). Numerical analysis of quasi-static fracture in functionally graded materials. *International Journal of Mechanics and Materials in Design,* 11, 405-424.

Martínez-Pañeda, E., & Niordson, C., (2015). On fracture in finite strain gradient plasticity. *International Journal of Plasticity,* (in press)

Oñate, E. (1995). *Cálculo de estructuras por el método de elementos finitos.* Centro Internacional de Métodos Numéricos en Ingeniería, UPC.

Pike, M. (2001). Introducing Finite Element Analysis in Statics*. Proceedings of the 2001 American Society for Engineering Education Annual Conference & Exposition*, Session 2268.

Rahman, T., & Valdman, J. (2013). Fast MATLAB assembly of FEM matrices in 2D and 3D: Nodal elements. *Applied Mathematics and Computation*, 219 (13), 7151-7158.

Teh, K., & Morgan, L. (2005). The application of Excel in teaching finite element analysis to final year engineering students. *Proceedings of the 2005 ASEE 4th Global Colloquium on Engineering Education*, paper 50.

Zecher, J. (2002). Teaching finite element analysis in an MET program. *ASEE Annual Conference Proceedings*, Session 3448

Emilio Martínez Pañeda es actualmente investigador visitante en la Universidad de Cambridge. Ha publicado numerosos artículos en revistas de prestigio en el ámbito de la mecánica computacional y ha impartido charlas como ponente invitado en varios congresos y universidades.
Email: mail@empaneda.com