Software libre para el análisis de estructuras por el método de los elementos finitos

Por: **Luis Claudio Pérez, Iturribizia, S.L.**

La empresa Iturribizia está desarrollando un programa de elementos finitos de fuente abierta orientado al cálculo de estructuras, y aunque todavía está en fase de desarrollo desean ponerlo a disposición de otros profesionales de esta disciplina para que puedan evaluarlo, realizar comentarios, establecer colaboraciones, etc. El autor del artículo nos explica las motivaciones que le llevaron a la elaboración de dicho programa, las conclusiones que obtuvo del estudio, así como las características que tendrá el XC. El enlace mediante el que se puede acceder al programa es: www.iturribizia.com/descarga_software.html

Xc es un programa desarrollado en Iturribizia, destinado a resolver problemas de análisis estructural mediante el método de los elementos finitos. El programa puede resolver varios tipos de problemas, desde un simple análisis lineal hasta simulaciones complejas no lineales. Posee una biblioteca de elementos finitos que permite modelar distintas geometrías, así como múltiples modelos de material, permitiendo su aplicación en distintas áreas del análisis de estructuras. El programa puede descargarse en la dirección: www.iturribizia.com/descarga software.html.

Motivación

Al parecer el escalador francés Lionel Terray, cuando se le preguntó el motivo por el que escalaba montañas respondió: porque están ahí. Algo parecido ocurre con el desarrollo de este programa. Desde que comencé el estudio del método de los elementos finitos, tras haber aprendido a emplear las soluciones analíticas a los problemas elásticos (tan limitadas), las posibilidades de su

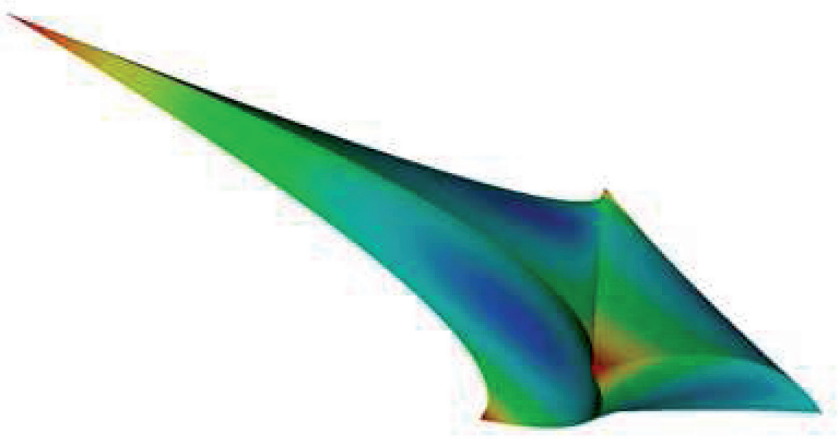

Figura 1: Representación de una malla de tetraedros (tomada de la web de VTK).

empleo para resolver problemas estructurales ejercieron sobre mí una gran atracción. Ello, unido a mi afición por la informática, hicieron que me decidiera a desarrollar un programa de elementos finitos que fuera útil para calcular estructuras y que pudiera modificar y ampliar del modo que quisiera. Así existió una primera versión escrita en Pascal que sólo podía trabajar con elementos de tipo barra. Tras ella vino otra escrita en C++ «desde cero» que nunca llegó a resolver un problema no trivial. Por último, al descubrir las posibilidades que ofrecía el núcleo de cálculo de OpenSees y al estar disponible el código fuente de este programa, me decidí a modificarlo de modo que perdiera su carácter académico y construir a partir de él un programa «de uso industrial» por así decirlo.

Como se ha dicho, el programa se ha desarrollado a partir del denominado OpenSees http://opensees.berkeley.edu desarrollado por el «Pacific earthquake engineering research center». El objetivo que se persigue con el trabajo efectuado es llegar conseguir un programa apto para ser utilizado en tareas de producción en cualquier estudio de ingeniería estructural. Para conseguir dicho objetivo, las principales modificaciones realizadas al código original han sido las siguientes:

1. Incorporación de un modelador para generación de la malla de elementos finitos. El modelador es capaz de crear mallas estructuradas a partir de la descripción de la geometría por medio de puntos, líneas, superficies y volúmenes.
2. Generación de gráficos mediante la biblioteca VTK (http://www.vtk.org). Esta es una biblioteca de fuente abierta para la generación de gráficos para uso científico.
3. El lenguaje de macros, desarrollado desde cero, facilita la obtención de los resultados producidos por cálculo sin necesidad de extraerlos de listados predefinidos. Dota al programa de un mecanismo por el que se facilita la expresión de una sentencia como obtén el cociente del movimiento vertical del nodo más próximo al centro de vano y la longitud total del vano.
4. Utilidades para la formación y cálculo de las combinaciones de acciones que prescriben las normas (EHE, EAE, Eurocódigos,. . .) de modo que se facilita la obtención de resultados y comprobación de los mismos para cada una de ellas.
5. Posibilidad de activar y desactivar elementos para posibilitar el análisis de estructuras construidas por fases, de problemas geotécnicos y del refuerzo de estructuras existentes.
6. Escritura de macros para la comprobación de la estructura y sus elementos de acuerdo con los criterios prescritos por distintos códigos (eurocódigos, EHE, EAE, CTE, etc.)
7. Modificación del código de modo que se enlace con las versiones «estándar» de las bibliotecas de álgebra lineal (BLAS, Arpack, Lapack, SuperLU,etc.) De este modo se elimina la necesidad de incluir en el programa versiones «ad-hoc» de dichas bibliotecas.
8. Modificación de los materiales de modo que admitan deformaciones impuestas de manera que sea posible resolver problemas en los que intervengan acciones térmicas y reológicas.

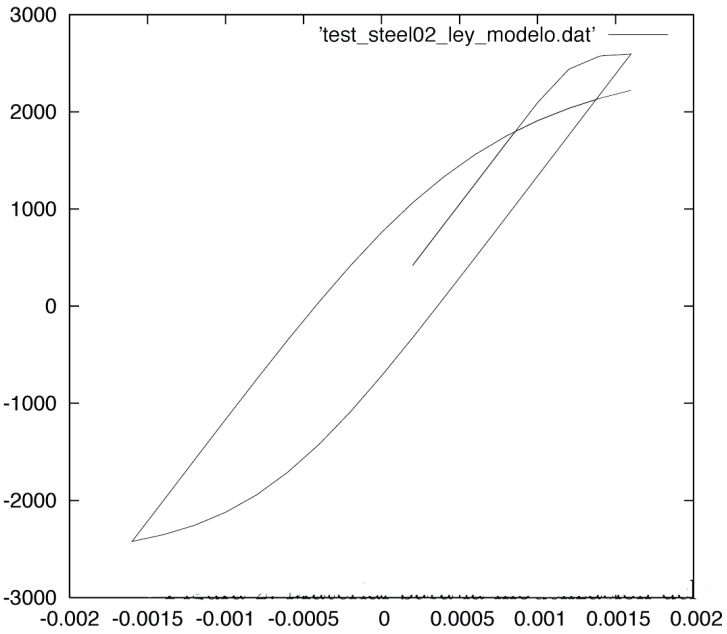

Figura 2: Material para la modelización del acero.

Criterios seguidos para el desarrollo del programa

De la experiencia adquirida en las labores de desarrollo realizadas con anterioridad se obtuvieron las siguientes conclusiones:

1. Probar, comprobar y recomprobar. Para cada funcionalidad del programa se deberá escribir un test de verificación que permita comprobar la corrección del código. Además, para cada modificación del código que se haga se deberá verificar que se completa correctamente toda la batería de test escrita hasta el momento. De este modo se hace más difícil introducir errores en el código inadvertidamente (¡Alguien dijo que un ordenador es una máquina que permite cometer miles de errores en muy poco tiempo). A pesar de esto se detectarán errores para los que también se deberá escribir un test de comprobación que per-

XC es un programa destinado a resolver problemas de análisis estructural mediante el método de los elementos finitos. Puede resolver varios tipos de problemas, desde un simple análisis lineal hasta simulaciones complejas no lineales, y posee una biblioteca de elementos finitos que permite modelar distintas geometrías, así como múltiples modelos de material, permitiendo su aplicación en distintas áreas del análisis de estructuras madera como CADIX

mita verificar que no vuelven a repetirse.

2. No reinventar la rueda. Emplear tanto como sea posible las bibliotecas de fuente abierta a las que puede accederse fácilmente a través de Internet. Siempre que, para alguna de las funciones del programa, es posible «delegar» la escritura del código (solución de sistemas de ecuaciones, procesamiento paralelo (MPI), teoría de grafos, etc.) debe hacerse así.

3. No perder el tiempo en el desarrollo de un elegante GUI (Graphics User Interface). Esto probablemente resulte bastante impopular porque las interfaces gráficas de usuario crean en él la ilusión de que sabe manejar el programa (Dice José Calavera que los programas se hacen para que los que saben calcular lo hagan más rápido no para que calculen los que no saben hacerlo.). Permiten adoptar un procedimiento de prueba y error que, a nuestro juicio, quizá pueda ser adecuado para manejar un procesador de textos en el que el resultado queda a la vista, pero no lo es tanto para un programa de cálculo cuyo manejo requiere una concienzuda revisión de los datos e hipótesis de partida. Por otra parte el empleo de un lenguaje de macros dota al programa de una flexibilidad mucho mayor. Basta pensar que para definir un segmento de recta se podrá hacer mediante dos puntos, un punto y un vector, un punto, una longitud y un ángulo, la intersección de dos planos, etc. Esta flexibilidad es prácticamente imposible de conseguir con una interfaz gráfica de usuario y ello a un coste exorbitante en tiempo de desarrollo.

4. La normas cambian, las leyes físicas permanecen. Mientras que, en lo que se refiere a la solución del problema mecánico seguimos manejando las leyes de la mecánica newtoniana, los avances en el conocimiento del comportamiento de los materiales y los cambios en el nivel de exigencia de la sociedad respecto al nivel de calidad exigible a sus infraestructuras, hacen que periódicamente se renueve el contenido de las normas de diseño (EHE,EAE, etc.). En consecuencia, y dado que no existe la necesidad de ocultar el código al usuario, se procurará escribir los algoritmos relativos a la normativa (comprobaciones,. . .) en forma de macros del intérprete de comandos que pueden modificarse con facilidad, reservando la escritura de código en C++ para aquellos algoritmos que expresan leyes más estables (equilibrio de fuerzas, inercia, modelos constitutivos de los materiales, etc.).

Especificaciones

En este apartado se exponen las características que tiene (o tendrá) el programa y que, a nuestro juicio, lo hacen (harán) particularmente apto para su empleo en el cálculo de estructuras.

Herramientas para la generación de informes

El programa debe tener utilidades que faciliten la generación de informes, tanto numéricos como gráficos, listos para incluir en la memoria de cálculo.

La base de esta capacidad para generar informes está en el empleo de LATEX (Ver www.latex-project.org.) como sistema de preparación de documentos. Respecto a los informes numéricos (listados u otros documentos similares) en el directorio «xc/macros/listados»pueden verse varios ejemplos de macros que se emplean para la generación de este tipo de informes. En cuanto a la generación de gráficos se emplean tres vías.

1. La primera, destinada a generar gráficos bi o tridimensionales con información acerca de los valores que presenta un campo escalar o vectorial en la malla de elementos finitos, se basa como ya se ha dicho, en el empleo de la biblioteca VTK. En el directorio «xc/macros/vtk» se encuentran distintas macros que se emplean para generar estos gráficos.
2. La segunda, basada en el empleo de gnuplot (Ver www.gnuplot.info.) tiene por objeto generar gráficos de funciones o datos numéricos como por ejemplo los que aparecen en la **figura 2**. Las macros que se emplean para generar estos gráficos se guardan en «xc/macros/gnuplot».
3. Por último existe la posibilidad de generar gráficos en PostScript (PostScript es un lenguaje de descripción de páginas utilizado en muchas impresoras y, de manera usual, como formato para gráficos en documentos de LATEX) mediante macros que llamen a funciones de la biblioteca plotutils. Esta última alternativa es la menos desarrollada de las tres. Su uso permitirá incluir en los informes

representaciones esquemáticas de las secciones de hormigón armado o perfilería de acero, etc., asignadas a los elementos de manera que se facilite la verificación de la bondad de los datos introducidos.

Herramientas para el cálculo de múltiples combinaciones de acciones

El programa debe facilitar en lo posible el cálculo de múltiples combinaciones de acciones tanto en cálculo lineal como no lineal. Como es sabido, la comprobación de una estructura requiere obtener la respuesta de la misma frente a un gran número de combinaciones (desde pocos cientos en estructuras sencillas de edificación hasta más de un millar en estructuras con cargas móviles con acciones térmicas y sísmicas). Además, las comprobaciones a realizar para cada una de estas combinaciones serán distintas para los elementos de acero, de hormigón, de madera y para las piezas especiales (bulones, pernos de anclaje, etc.) que formen parte de la estructura. Cuando el cálculo es lineal, la respuesta a la combinación podrá obtenerse como suma de las respuestas a cada una de las acciones actuando aisladamente. Por el contrario, cuando se trata de un cálculo no lineal será necesario realizar el análisis para cada una de las combinaciones planteadas, aprovechando en lo posible los resultados intermedios ya calculados.

Las utilidades de las que se dispondrá para automatizar las comprobaciones serán las siguientes:

1. Rutinas que permitan la generación automática de las combinaciones a calcular.
2. Rutinas que permitan agrupar los componentes de la malla en conjuntos cuyos miembros se selecciones por el material, por su posición, por su geometría, etc. De modo que para cada uno de esos conjuntos puedan programarse distintas series de comprobaciones según la norma que se debe aplicar y las magnitudes a considerar (esfuerzos, deformaciones, movimientos, etc.)
3. Implementación de un «dosificador de cargas» que permita la aplicación gradual y ordenada de las acciones en el cálculo no lineal. Es necesario tener en cuenta que cuando se realiza un cálculo no lineal, para el que la solución no es única, es importante el orden de aplicación de las cargas (Considérese por ejemplo un cable sobre el que actúe el sismo antes que la tensión a la que estará sometido o que su peso propio.) También es importante la aplicación gradual de las cargas. Esto es, la obtención de la solución en «puntos intermedios» del proceso de carga de modo que se facilite la convergencia del algoritmo de solución (En ocasiones la ausencia de estos pasos intermedios hace sencillamente imposible la obtención de la solución).
4. Rutinas que permitan el uso de la operación «restart». Es decir, que permitan realizar un análisis a partir de un estado previo ya calculado. Supongamos que se quiere calcular la hipótesis 1,35 9 G + 1,5 9 V + 1,0 9 S y anteriormente se ha calculado la hipótesis 1,359G+1,59V . Parece claro que, en general, se empleará menos tiempo de proceso si se cargan los resultados del análisis que se obtuvieron para 1,35 9 G + 1,5 9 V y a continuación se aplica 1,0 9 S y se obtiene la solución.

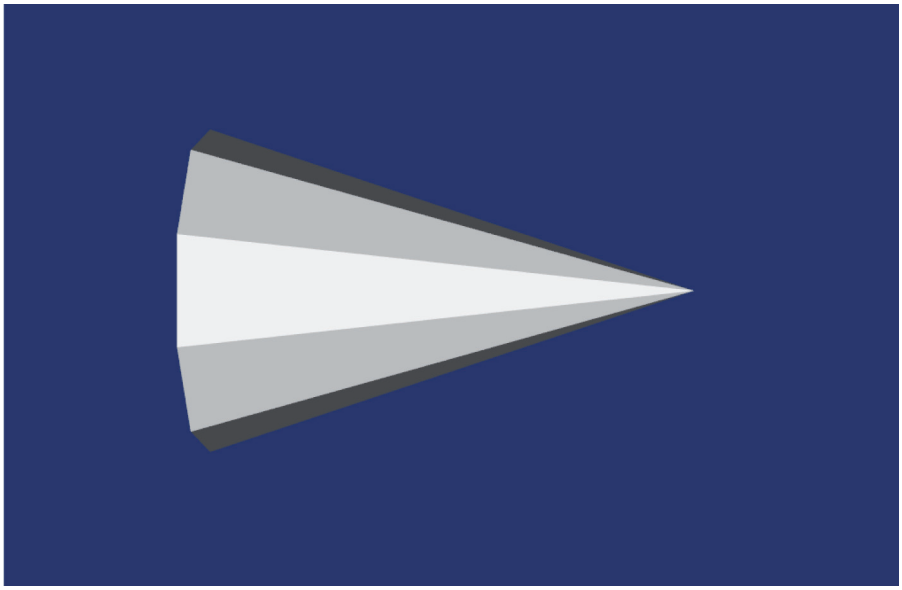

Figura 3. Cono generado mediante el lenguaje de macros de XC..

Herramientas para la comprobación de la estructura

La comprobación de la bondad del diseño de una estructura de acuerdo con las prescripciones de las normas suele estar basada en la aplicación de determinados criterios respecto a:

-Estabilidad: Comprobación del equilibrio de la estructura y de su seguridad frente a fenómenos de pandeo. Para permitir la comprobación de estos aspectos el programa ofrece las siguientes vías:

- Empleo del método P-Delta (sólo en elementos de tipo barra).
- Permite realizar «pushover analysis».
- En un futuro se podrán realizar «linear buckling analysis» que sirvan para seleccionar los modos de pandeo a analizar a posteriori.

-Resistencia del material: tensiones y deformaciones máximas que pueden soportar los materiales. En estas com-

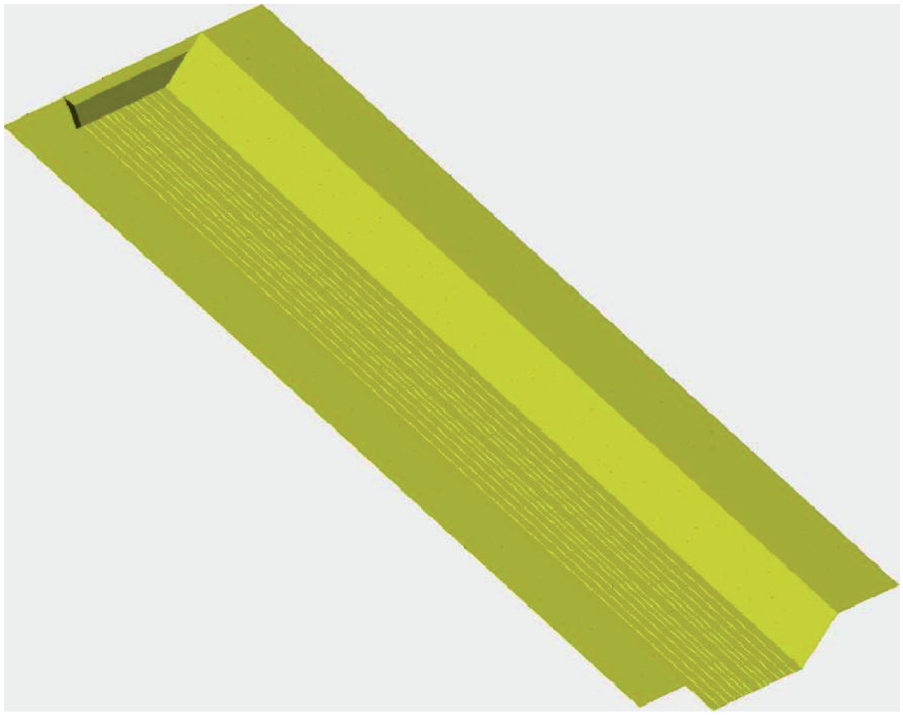

Figura 4. Modelo de elementos shell de un tablero de puente..

probaciones es en las que más se ha avanzado por ser las más sencillas de realizar y las que han de realizarse siempre. En el directorio «xc/macros/materiales» pueden encontrarse ejemplos de implementación de estas comprobaciones.

-Rigidez de la estructura: movimientos máximos admisibles en puntos característicos (límites para la flecha, desplome, etc.). Estas comprobaciones, en general, se pueden expresar con sencillez conocidos los desplazamientos de los nodos más próximos a los que hemos llamado puntos característicos. A diferencia de las comprobaciones de resistencia del material, normalmente es necesario programar estas comprobaciones en cada caso particular pero esta programación resulta trivial por lo que no parece necesario establecer mecanismos más elaborados. Resulta de utilidad la capacidad del programa de buscar el nodo o elemento más próximo a punto dado (definido por sus coordenadas).

La tarea de definir la malla de elementos finitos que modelen adecuadamente las estructuras de ingeniería civil (edificios, presas, puentes, etc.) es una de las que más tiempo consume puesto que la geometría de sus elementos raramente admite una representación analítica y, además, es frecuente la presencia de huecos, aligeramientos y refuerzos que complican el modelo. En algunas estructuras será necesario modelar tendones pretensados embebidos en el hormigón u otros elementos similares

-Comportamiento frente a vibraciones: comprobación de condiciones de confort y frecuencias naturales alejadas de las frecuencias de la excitación. Para realizar estas comprobaciones el programa permite obtener las frecuencias naturales de la estructura tanto si se emplean materiales lineales como no lineales.

-Comportamiento frente a fatiga: Capacidad de la estructura para resistir los ciclos de carga a los que se verá sometida a lo largo de su vida. En este terreno está todo por hacer.

Herramientas que faciliten la generación de la malla

La tarea de definir la malla de elementos finitos que modelen adecuadamente las estructuras de ingeniería civil (edificios, presas, puentes, etc.) es una de las que más tiempo consume puesto que la geometría de sus elementos raramente admite una representación analítica y, además, es frecuente la presencia de huecos, aligeramientos y refuerzos que complican el modelo. En algunas estructuras será necesario modelar tendones pretensados embebidos en el hormigón u otros elementos similares.

Representación gráfica de los resultados

Para dotar al programa de la capacidad de representar gráficamente los resultados obtenidos se tantearon las siguientes posibilidades:

1. Programación de la salida gráfica empleando directamente la interfaz OpenGL, presente en casi cualquier máquina de fabricación reciente.
2. Empleo de la biblioteca OpenDX, un software de fuente abierta basado en el visualization data explorer de IBM.
3. Empleo de la biblioteca VTK, de Kitware Inc.

Además del problema de la interfaz a utilizar, existía la cuestión de decidir el modo en que el programa generaría los gráficos. Se presentaban dos posibilidades, por una parte cabe programar una serie de salidas gráficas «precoci-

El programa debe facilitar en lo posible el cálculo de múltiples combinaciones de acciones tanto en cálculo lineal como no lineal. Cuando el cálculo es lineal, la respuesta a la combinación podrá obtenerse como suma de las respuestas a cada una de las acciones actuando aisladamente. Por el contrario, cuando se trata de un cálculo no lineal será necesario realizar el análisis para cada una de las combinaciones planteadas, aprovechando en lo posible los resultados intermedios ya calculados.

nadas» (deformada, tensiones, deformaciones, cargas,. . .) cuyo aspecto, escala, etcétera
pudiera ser controlado mediante una serie de parámetros y por otra el diseño de lenguaje de comandos que permitiera definir la salida gráfica.
El primer enfoque es el comúnmente empleado en programas de cálculo matricial y en otros destinados a tareas específicas como pueda ser el análisis de estabilidad de taludes o la solución de problemas de elasticidad plana. La principal desventaja de esta solución es su rigidez ya que hace muy difícil su aplicación es situaciones que no fueron consideradas previamente por el programador.
La segunda solución es la que emplean la mayor parte de los programas de análisis por elemento finitos de prop´osito general (ANSYS,Abacus,etc.). Su dificultad fundamental es la concepción del lenguaje de comandos de forma que sea flexible y fácil de utilizar. Durante el análisis de esta solución estudiamos diversos manuales de otros programas con objeto de conocer (ANSYS, Abacus, Solvia, Calculix Graphix, etc.) las capacidades de las que debería dotarse al lenguaje de gráficos.
Tras estudiar ambas soluciones decidimos que la primera resultaba casi inviable si pretendíamos que el programa fuera capaz de tratar múltiples tipos de problemas (análisis de edificios, puentes, depósitos, etc.). La segunda de las soluciones resultaba más adecuada para conseguir el objetivo puesto que con ella se dota al usuario de las herramientas que necesita para generar los gráficos y, de paso, se le traslada la tarea de crearlos a partir de las posibilidades que éstas le brindan.
Una vez decidida la solución a adoptar quedaba la tarea, nada trivial, de concebir un lenguaje de comandos que permitiera generar los gráficos manejando adecuadamente todas las posibilidades (transparencia, iluminación, texturas, etc.) que ofrecen los gráficos por ordenador. Aquí entraron en juego las posibilidades que ofrece VTK. Por una parte, al estar esta biblioteca escrita en C++ (el mismo lenguaje con el que escribimos el resto del programa), resultaba bastante sencillo enlazarla con el código que ya teníamos escrito. Además el hecho de que VTK disponga de interfaces en Tcl/Tk, Java y Python nos hizo darnos cuenta de que el lenguaje de comandos ya estaba escrito (la propia API de VTK) y lo único que teníamos que hacer era trasladarlo al lenguaje de macros que emplea el resto del programa. Además procediendo de este modo permitíamos emplear los gráficos con cualquier fin que el usuario desee (no sólo representar el modelo de elementos finitos). Así lo hicimos y, por ejemplo en la **figura 3** (que resultará reconocible para todos los que manejan VTK por ser uno de los ejemplos que acompañan al código) se representa el resultado de la macro que figura en el **cuadro 1**. En la figura 4 se representa un modelo de elementos finitos generado con el programa.

```
\vtk
{
\define["vtkConeSource","cone"]
{ \altura{3.0} \radio{1.0} \resol{10} }
\define["vtkPolyDataMapper","coneMapper"]
{ \set_input{"cone"} }
\define["vtkActor","coneActor"]
{ \set_mapper{"coneMapper"} }
\define["vtkRenderer","ren1"]
{
\add_actor{"coneActor"}
\set_background{0.1,0.2,0.4}
}
\define["vtkRenderWindow","renWin"]
{ \add_renderer{"ren1"} \set_size{1024,768}
}
\define["vtkRenderWindowInteractor","iren"]
{ \set_render_window{"renWin"} }
\define["vtkInteractorStyleTrackballCamera"
,"style"]
{}
\iren{\set_interactor_style{"style"}}
\iren{\initialize{} \start{}}
}
```

Cuadro 1: Código de la macro que genera el cono de la figura 3.

Agradecimientos

Quisiéramos manifestar nuestro agradecimiento a todos los que de uno u otro modo han hecho posible el enorme desarrollo que el software libre tiene todo el mundo. Gracias también a los profesores Filip C. Filippou y Frank Mackenna del departamento de ingeniería civil y medioambiental de la universidad de California (Berkeley), por tomarse la molestia de atender nuestros correos electrónicos. Gracias a mi familia por su paciencia y a mis amigos por su apoyo.