XX Congreso de Ecuaciones Diferenciales y Aplicaciones
X Congreso de Matemática Aplicada
Sevilla, 24-28 septiembre 2007
(pp. 1–8)

OctMesh: un entorno de elementos finitos en *Octave*

R. Rodríguez Galván[1],

[1] *Dpto. Matemáticas, Universidad de Cádiz, Facultad de Ciencias, Campus del Río San Pedro s/n, 11510, Puerto Real, Cádiz. E-mails:* rafael.rodriguez@uca.es.

Palabras clave: Elementos finitos, ecuaciones en derivadas parciales, computación paralela

Resumen

En este trabajo se presenta a *OctMesh* [11], un entorno de herramientas o *toolbox* para la resolución de ecuaciones en derivadas parciales mediante el método de los elementos finitos sobre Octave. Octave [4] es un entorno de cálculo numérico con licencia libre que utiliza de forma nativa el lenguaje de *Matlab* y es altamente compatible con éste.

Técnicamente, *OctMesh*, constituye una interfaz para el acceso desde Octave a las posibilidades de *libMesh* [7, 8], una biblioteca (también con licencia libre) para la simulación numérica de ecuaciones en derivadas parciales mediante el método de elementos finitos 1D, 2D ó 3D en arquitecturas paralelas (aunque puede funcionar en entornos secuenciales), además de incluir métodos para refinado adaptativo de mallas. Para todo ello utiliza varios paquetes de software de calidad contrastada, como la biblioteca paralela PETSc[1, 2, 3]. *LibMesh* está escrita en el lenguaje C++ y desarrollada, en su mayor parte, en el CFDLab de la Universidad de Texas.

Para problemas con altos requerimientos de cálculo, *OctMesh* facilita el acceso desde octave a módulos programados directamente en *libMesh*, con resultados óptimos.

1. Introducción

Los programas asociados a la resolución numérica de ecuaciones en derivadas parciales requieren de una gran potencia de cálculo, tanto mayor cuanto más realistas pretenden ser los resultados. La utilización de máquinas cada vez más potentes y la programación de algoritmos paralelos nos permiten abordar problemas que, en otro caso, habrían sido difícilmente accesibles. Dejando de estar restringidas a grandes centros de computación, las máquinas paralelas de una potencia razonable se encuentran al alcance de cualquier universidad o incluso de grupos de investigación. Y, de hecho, la generalización de ordenadores de doble núcleo y de arquitecturas de 64 bits pueden convertir a cualquier equipo doméstico en un entorno válido para el aprendizaje y la experimentación, superando las

características que se podían encontrar, años atrás, en una estación de trabajo de tipo medio.

Sin embargo, el aprovechamiento de estas arquitecturas encuentra un inconveniente: la necesidad de formación por parte de los investigadores para que éstos puedan utilizar lenguajes de programación avanzados y bibliotecas paralelas dentro de su área de trabajo. Por otro lado, existe un gran número de investigadores que, ante la necesidad de experimentar con distintos algoritmos, comparar los mismos, y contrastar con resultados teóricos, necesitan entornos de desarrollo rápido y prefieren evitar el ciclo de trabajo de lenguajes compilados (diseño, programación, ejecución, depurado de errores), que resulta con frecuencia poco ágil ante problemas de la complejidad de la resolución numérica de EDP. Con frecuencia, éstos optan por utilizar lenguajes interpretados como *Freefem++*[9] o *Matlab* (dotado de un *toolkit* de elementos finitos). Además, sería deseable que lo anterior fuese compatible con el siguiente objetivo: que aquél investigador que conozca un lenguaje de programación avanzado como C++ pueda, de forma muy sencilla, utilizar una potente biblioteca de elementos finitos para desarrollar en este lenguaje las zonas críticas del programa, de cara al optimizar el rendimiento del modelo numérico final.

Así, la cuestión es conseguir contar con entornos de trabajo de una flexibilidad y comodidad de uso similar a los anteriores, permitiendo aprovechar en el mayor grado posible la potencia de los equipos y de las máquinas que, cada vez más, se encuentran a nuestra disposición. En este contexto, nace *OctMesh*.

2. Qué es Octmesh

En el contexto anterior, *OctMesh* intenta conseguir un compromiso entre la potencia de cálculo de *libMesh*, una librería de elementos finitos paralela, y la facilidad de uso de los lenguajes interpretados orientados al cálculo numérico y matricial.

En concreto, se trata de un entorno de herramientas o *toolbox* para la resolución de ecuaciones en derivadas parciales mediante el método de los elementos finitos sobre Octave.

2.1. Octave

Octave o *GNU Octave* [4] es un entorno de cálculo numérico con las siguientes características:

- Su lenguaje de programación es altamente compatible con el lenguaje de *Matlab*
- Permite tanto una ejecución interactiva o como la programación y ejecución de ficheros por lotes.
- Tiene licencia libre y, aunque originalmente fue desarrollado para entornos Unix también puede utilizarse en sistemas Windows y Mac.
- Está escrito en el lenguaje C++, usando la librería STL (Standard Template Library). Su código fuente está disponible y en particular contiene una librería de cálculo matricial para C++.
- Ha sido diseñado de forma que sea extensible dinámicamente, a través de nuevos procedimientos escritos en C++. Estas extensiones son programas “especiales” que

tienen la extensión `.oct` y, como programas compilados, cuentan con un rendimiento mucho mayor que las funciones escritas en el lenguaje interpretado de Octave.

A la hora de la elección de Octave frente a otros entornos y lenguajes interpretados (como *Python* o el propio *Matlab*), se valoraró positivamente la compatibilidad con este último programa, cuyo uso está ampliamente difundido, así como su posibilidad de extensión y las ventajas derivadas del libre acceso al código fuente. Como desventaja, no cuenta aún con una interfaz de usuario tan amigable como *Matlab*, aunque se trata de un inconveniente menor.

Para desarrollar *OctMesh*, la idea fue diseñar una serie de procedimientos que extendieran Octave, creando un marco que permitiera acceder desde este lenguaje a la biblioteca de elementos finitos *libMesh*. El objetivo era facilitar todo lo posible el uso de esta biblioteca, a la vez que mantener, dentro de lo posible, su potencia de cálculo. Por último, se trataba de hacer muy sencilla la tarea de desarrollar módulos `.oct` utilizando *libMesh*, en lenguaje C++, a los que se pudiera acceder desde Octave.

2.2. libMesh

LibMesh es una biblioteca de elementos finitos escrita en el lenguaje de programación C++ que ha sido diseñada en el CFDLab de la Universidad de Texas, aunque a ella han contribuido también otras personas y entidades. *LibMesh* fue diseñada desde el origen con la intención de aprovechar las características más avanzadas del lenguaje C++, como el uso de *templates* y la programación genérica, de forma similar y generalizando a la STL (Standard Template Library), que es ampliamente utilizada por *libMesh*. La existencia en la actualidad de compiladores de gran calidad que implementan el estándar C++, entre ellos el compilador del proyecto GNU, hacen que esta biblioteca sea utilizable en numerosas plataformas de hardware y en distintos sistemas operativos.

La biblioteca ofrece refinamiento adaptativo para mallas arbitrarias y permite la realización de cálculos en paralelo con poco esfuerzo por parte del programador. La paralelización se introduce a nivel del ensamblaje de las matrices que forman el sistema lineal asociado al método de elementos finitos, así como en su resolución. Para ello, se utiliza un método estándar descomposición de dominio sin solapamiento, mediante una descomposición de la malla en la cual cada procesador contiene la malla global pero realiza sus cálculos solamente en un subconjunto particular de la misma. Esta aproximación es válida tanto en sistemas de memoria compartida como en sistemas de memoria distribuida, de forma que es posible realizar simulaciones en un amplio abanico de sistemas, incluyendo a clusters de ordenadores Linux.

De forma interna, *libMesh* utiliza distintos paquetes de software de calidad contrastada. En concreto, para la división de la malla en sub-dominios existe la posibilidad de utilizar tanto METIS[6] como PARMETIS[10]. Como soporte de estructuras de datos y para la resolución paralela de sistemas de ecuaciones lineales, *LibMesh*, emplea a PETSC [1, 2, 3], un reputado entorno de computación científica paralela desarrollado en el Mathematics & Computer Science Division del Argonne National Laboratory (EE.UU.), que a su vez utiliza el estándar MPI[5] como sistema de intercambio de mensajes.

La biblioteca permite aplicar distintas formulaciones de elementos finitos, como métodos de Galerkin, Petrov-Galerkin y Galerkin discontinuo y ofrece varias familias de elementos finitos 1D, 2D y 3D, entre ellas elementos finitos de tipo Lagrange de primer y

```
   // #include (...)
int main (int argc, char** argv)
{
  // Inicializar la biblioteca (incluyendo el sistema de cálculo paralelo)
  libMesh::init (argc, argv);
  // Hacer que los objetos sean destruidos antes de terminar main()
  {
    unsigned int dim=3;    // Definir la dimensión de la malla
    Mesh mesh(dim);        // Crear una malla
    mesh.read("fichero-malla");  // Leerla de un fichero
    mesh.print_info();     // Mostrar información sobre la malla
  }
  return libMesh::close(); // Fin, cerrar la biblioteca
}
```

Figura 1: Programa simple, escrito en C++, con *libMesh*

```
   # Inicializar la biblioteca (incluyendo el sistema de cálculo paralelo)
libmesh_init;
dim=3;                  # Definir la dimensión de la malla
mesh = libmesh_mesh(dim);   # Crear una malla
mesh.read("fichero-malla"); # Leerla de un fichero
mesh.print_info();      # Mostrar información sobre la malla
clear;           # Hacer que todos los objetos creados sean destruidos
libMesh_close(); # Fin, cerrar la biblioteca
```

Figura 2: Programa simple, escrito en el lenguaje de Octave con *OctMesh*

segundo orden en triángulos, cuadriláteros, tetraedros, hexaedros, prismas, pirámides, etc, así como elementos finitos jerárquicos, elementos discontinuos y algunos elementos C^1.

3. El diseño de OctMesh

Técnicamente, *OctMesh* está constituido por una biblioteca de enlace dinámico, a la que se ha llamado *libOctMesh* y por una serie de ficheros `.oct`. Cada uno de estos ficheros constituye una nueva función en Octave y utiliza a la biblioteca anterior para acceder a las funcionalidades de un objeto *LibMesh*. La idea final es conseguir que cada uno de estos objetos, sea accesible a través de un fichero `.oct`, de tal forma que la Biblioteca al completo sea accesible desde Octave.

Al diseñar la sintaxis de *OctMesh*, se ha intentado hacerla tan similar como sea posible a las clases y métodos de *libMesh*. Por ejemplo, consideremos un sencillo programa escrito en lenguaje C++ utilizando la biblioteca *libMesh*, tal y como el mostrado en la figura 1

En la figura 2 podemos ver un programa equivalente, desarrollado en Octave con *OctMesh*. Obsérvese que esta secuencia de instrucciones podría haber sido introducida, de forma interactiva, dentro del intérprete de Octave.

3.1. La biblioteca libOctMesh

La biblioteca *libOctMesh* constituye la verdadera interfaz entre Octave y *libMesh*. Se trata de una biblioteca de enlace dinámico que se encarga de mantener abierta una sesión de *libMesh*, y que es accesible desde Octave mediante ficheros `.oct` que son enlazados con *libOctMesh*. En su parte fundamental, está formada, por una clase C++ llamada `Workspace`, que implementa el patrón de diseño "*Singleton*" de forma que se ofrece un único punto de acceso a *libMesh* en cada sesión de Octave. Entre los numerosos métodos proporcionados por `Workspace` se encuentran `init()` y `close()`, que se ocupan de inicializar y cerrar *libMesh* después de que haya sido utilizada desde Octave. Por otra parte, esta clase se ocupa de crear, almacenar en distintos contenedores y destruir después de que hayan sido utilizados, los objetos de *libMesh* que están siendo utilizados desde Octave (como mallas, ecuaciones, parámetros, etc.), así como de proporcionar métodos que permiten acceder a los mismos. Cada uno de estos objetos estará identificado a través de un número índice, que constituirá la única información almacenada en Octave y será la llave para acceder a los mismos.

3.2. Los ficheros .oct

Una vez construida la librería anterior, el desarrollar la interfaz *OctMesh* debería ser una tarea laboriosa pero, técnicamente, no excesivamente complicada. Sin embargo, puesto que el lenguaje de Octave no es orientado a objetos, el conseguir que *OctMesh* tuviera una sintaxis similar a la de *libMesh*, tal y como se mostró en se convirtió en un obstáculo importante. Por fortuna, fue posible utilizar una técnica que ha sido empleada por éxito por *Octaviz*, un entorno para Octave que hace que las clases de VTK (un entorno de visualización de gráficos 3D) sean accesibles desde Octave. Esta técnica se basa en definir una clase C++, llamada `octave_object`, derivada de `octave_base_value` (la clase de la cual, en el código C++ de Octave, se derivan todos los objetos, desde números hasta matrices o funciones) y por tanto un objeto "de pleno derecho" en Octave. A continuación, sobrecargamos del operador punto (".") en Octave con el fin de poder dotar a un `octave_object` de todos aquellos métodos que creamos conveniente.

Mediante ficheros `.oct`, se crean los objetos adecuados en Octave, de tipo `octave_object`, y se a cada uno de ellos se le asigna el número índice que le permitirá acceder, a través de *libOctMesh*, al objeto equivalente de *libMesh*. Por ejemplo, la orden

```
m = libmesh_mesh(3);
m.print_info();
```

localizará el fichero `libmesh_mesh.oct` para crear un objeto de tipo `octave_object` (asignado, este caso, a la variable `m`). En el momento de su creación, este objeto construirá, a través de la librería *libOctMesh*, un objeto de *LibMesh* de tipo `Mesh` y actuará como interfaz para manejar este objeto desde Octave. Por ejemplo, la orden `m.print_info()` llamará al método del mismo nombre del correspondiente objeto de *LibMesh*.

Llegados a este punto, el resto de la tarea de creación de *OctMesh* se limita a la laboriosa puesta apunto de nuevos envoltorios para cada una de las clases de *libMesh*, limando distintas particularidades (por ejemplo, el acceso a *templates* desde Octave). En la versión 0.1 de *OctMesh*, se puede acceder a los objetos de más alto nivel de *libMesh* (como `Mesh`, `Parameter`, `System`, `EquationSystems`, `FEType`, etc) y a bastantes de los

métodos de más bajo nivel, es decir, utilizados para el montaje de los sistemas lineales asociados a cada EDP (como `DofMap`, `Point`, etc).

3.3. Evitando problemas de rendimiento

Octave, como lenguaje interpretado, no se caracteriza por su alto rendimiento en bucles ni en el acceso a vectores y matrices. El utilizarlo para el montaje de los sistemas lineales significa optar por la comodidad del lenguaje pero a costa de sufrir una inevitable pérdida de rendimiento. En muchos de los casos, esta circunstancia puede no ser crítica, pero en otras puede llegar a convertirse en un factor determinante.

Por fortuna, podemos utilizar las extensiones `.oct` para obtener una solución al dilema anterior. Supongamos que deseamos utilizar *OctMesh* para resolver un problema de Poisson 3D. Para ello, desde Octave, creamos una malla, `m`, sobre la cual declaramos un sistema de ecuaciones, al cual añadimos un sistema lineal e implícito llamado "Poisson":

```
m = libmesh_mesh(3);
eq_sys = libmesh_equations_system(m);
poisson = eq_sys.add_system("LinearImplicitSystem", "Poisson");
```

En *libMesh* se debe identificar a una función que se encargue de ensamblar el sistema de ecuaciones. Esto lo realizaríamos en *OctMesh* a través de la siguiente orden:

```
poisson.attach_assemble_function ("assemble_poisson");
```

donde `assemble_poisson` es una función definida en Octave cuya tarea es el montaje del sistema. Pues bien, en caso de que detectemos problemas de rendimiento en las funciones de ensamblaje, podemos optar por sustituirlas por funciones equivalentes (y con una sintaxis muy similar) pero desarrolladas en C++ y compiladas como extensiones dinámicas de Octave (`.oct`). Estas funciones aprovecharán directamente el alto rendimiento de *libMesh* y eliminarán el cuello de botella. Sería deseable, incluso, contar con una colección de funciones de ensamblaje de este tipo para aquellos problemas que sean de uso más frecuente[1] en el ámbito de las EDP, de tal forma que se puedan reutilizar desde Octave tantas ocasiones como sea necesario. Esta es la técnica que se ha empleado en el siguiente ejemplo, consiguiendo óptimos resultados.

4. Un ejemplo: resolución de las Ecuaciones de Navier-Stokes

Con el fin de comprobar la eficacia de *OctMesh*, se ha partido de uno de los ejemplos incluidos en *libMesh* (concretamente, el ejemplo 13), creando un programa en Octave que realizara exactamente los mismos pasos que éste y comparando el tiempo de ejecución.

Este ejemplo consiste en la resolución de las conocidas ecuaciones de Navier-Stokes 2D, ecuaciones no lineales que gobiernan fluidos incompresibles

$$(N-S) \qquad \begin{cases} \mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} - \nu\,\Delta\mathbf{u} + \nabla p = \mathbf{f}, \quad \nabla\cdot\mathbf{u} = 0 \ \text{ en } \Omega\times(0,T), \\ \mathbf{u} = 0 \ \text{ sobre } \partial\Omega\times(0,T), \quad \mathbf{u}_{|t=0} = \mathbf{u}_0 \ \text{ en } \Omega, \end{cases}$$

[1] De hecho, ya se han desarrollado funciones para los problemas de Poisson, Stokes y Navier-Stokes

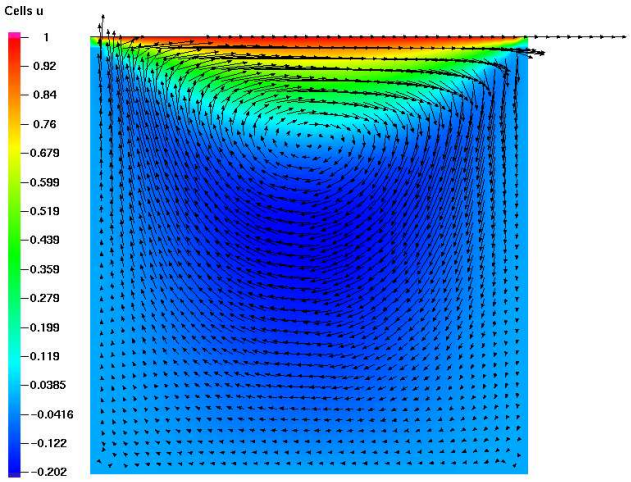

Figura 3: Test de la cavidad, velocidad y presión tras 15 iteraciones de tiempo.

siendo $\mathbf{u}(\mathbf{x}, t) \in I\!R^2$ la velocidad del fluido que en el instante $t \in (0, T)$ se encuentra en $\mathbf{x} \in \Omega$, $p(\mathbf{x}, t) \in I\!R$ la presión, $\nu > 0$ la viscosidad y $\mathbf{f}(\mathbf{x}, t) \in I\!R^2$ una fuerza externa.

El experimento, en concreto, es el conocido test de la cavidad rectangular, $[0, 1]^2$, llena de un fluido al que se impone una velocidad constante en lado superior, en dirección paralela éste, y en los lados restantes se impone una condición de no deslizamiento.

La discretización en espacio se realiza mediante una malla uniforme formada por 20×20 rectángulos, con 9 grados de libertad situados en los vertices, en los puntos medios de los lados y en el centro. Utilizamos elementos finitos de Lagrange de segundo orden para las velocidades y primer orden para la presión, consiguiendo un par velocidad-presión que es LBB-estable. Para la discretización en tiempo se utiliza un esquema de Euler implícito, mientras que la no linealidad se trata mediante un método de Newton inexacto.

A la hora de ejecutar el test, se optó por crear una extensión dinámica `.opt`, a la que se llamó `assemble_navier_stokes`, de tal forma que la tarea de montaje de la matriz y del segundo miembro del sistema lineal resultante (donde se requieren mayores prestaciones de cálculo) se pudiera realizar en C++. Este módulo esta diseñado con la intención de aprovechar las técnicas de paralelización de Octave, aunque esta faceta todavía no ha sido sometida a pruebas. El programa, escrito en el lenguaje de Octave, es responsable de la construcción del mallado de $[0, 1]^2$, definir el sistema de ecuaciones, las variables (velocidades y presión), el tipo de discretización, indicar la función de ensamblaje que se debe utilizar y definir distintos parámetros (paso de tiempo, número de iteraciones, etc.). Dentro de Octave, realizamos un bucle de iteraciones en tiempo, dentro e cada una de las cuales llevaremos a cabo iteraciones del método no lineal hasta que detectemos su convergencia (siempre que la norma L^2 entre dos iteraciones consecutivas y el residuo sean menores que una tolerancia previamente fijada).

Los resultados obtenidos, que se representan en la figura 3, coinciden con los obtenidos al ejecutar el ejemplo 13 de *libMesh*, que fue tomado como punto de partida. La pena-

lización en el tiempo de ejecución que introduce octave, como lenguaje interpretado, es del orden del 1.5 %. Concretamente, en un equipo Athlon 64 X2 Dual Core 4200+, los tiempos de proceso del programa C++ frente al programa equivalente en Octave fueron, respectivamente, de 32.027 y 32.577 segundos. Por supuesto, todo esto gracias a que el módulo usado en el montaje de las matrices era de tipo `.oct`.

Conclusiones y trabajos futuros

OctMesh es un entorno de herramientas para GNU Octave que nos permite realizar experimentos numéricos mediante el método de los elementos finitos de forma interactiva. OctMesh utiliza *libMesh*, una librería paralela de elementos finitos 1D, 2D y 3D en C++ con características avanzadas, como refinado adaptativo de mallas. Para problemas con altos requerimientos de cálculo, se facilita el acceso desde octave a módulos programados directamente en *libMesh*, con resultados óptimos.

A corto plazo, se realizarán pruebas del rendimiento de las características de paralelización desde *OctMesh*. Entre las tareas pendientes, sería deseable completar el acceso a tantas funcionalidades de *libMesh* como sea posible. Una posible línea de trabajo adicional, que técnicamente ofrecería pocos obstáculos, es la realización de una capa de compatibilidad con algunos de los toolkits de EDP existentes para Matlab (proporcionando un sustituto a éstos que tendría características avanzadas como la paralelización).

Agradecimientos

A Francisco Guillén González, por sus indicaciones y su apoyo. Este trabajo ha sido parcialmente subvencionado por el grupo de investigación FQM-315 de la Junta de Andalucía.

Referencias

[1] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc: Portable, extensible toolkit for scientific computation, (2001). `http://www.mcs.anl.gov/petsc`.

[2] Satish Balay, Victor Eijkhout, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, (1997).

[3] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, (2004). Science Division.

[4] J.W. Eaton y otros. *Octave*. `http://www.octave.org`

[5] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, **22**(6), 789–828 (September 1996).

[6] G. Karypis and V. Kumar. Metis. `http://www-users.cs.umn.edu/karypis/metis/metis/main.html`.

[7] B. Kirk, J. Peterson y otros. *LibMesh Finite Element Library*. `http://LibMesh.sourceforge.net/`

[8] B. Kirk, J. Peterson, R.H. Stogner, G.F. Carey *LibMesh: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening*. Engineering with Computers, preprint (2006).

[9] Hecht F., Pironneau O., Le Hyaric A., Ohtsuka K., FreeFem++ Manual, `http://www.Freefem.org`.

[10] ParMETIS home page. `http://www-users.cs.umn.edu/ karypis/metis/parmetis/index.html`.

[11] R. Rodríguez Galván, *OctMesh*. `http://octmesh.forja.rediris.es`