Mecánica Computacional, Volumen XIX, pp. 405-410
Compilado por: F. Quintana y S. Felicelli
San Carlos de Bariloche, Noviembre de 2000

IMPLEMENTACIÓN DE UN PROGRAMA DE ELEMENTOS FINITOS DE PROPÓSITO GENERAL, MULTI-FÍSICA PARA USO EN ENTORNOS DE CÁLCULO DISTRIBUIDO

Norberto Nigro, Victorio Sonzogni, Mario Storti y Andrea Yommi
Centro Internacional de Métodos Computacionales en Ingeniería
CIMEC-CONICET-UNL, Santa Fe, Argentina
http://venus.arcride.edu.ar/CIMEC, mstorti@intec.unl.edu.ar

RESUMEN

Se ha desarrollado un programa de elementos finitos de uso general para uso en redes de cálculo distribuido, usando las librerías PETSc/MPI, llamado PETSc-FEM [3]. En [2] se describen las aplicaciones existentes mientras que en [5] se describe la eficiencia computacional del programa en un cluster Beowulf construido en el CIMEC [1]. En este trabajo se describen los detalles de programación del paquete (librería mas programas de aplicación). El diseño del paquete no está orientado a dar una serie de aplicaciones cubriendo toda la gama de posibilidades sino más a dar una librería con la cual los usuarios puedan desarrollar nuevas aplicaciones. El objeto básico en la librería no es un elemento sino, un conjunto de elementos o "elemset". Se asume que gran parte que la información relativa a los elementos de un elemset (propiedades físicas y parámetros u opciones numéricas) es la misma para todo el elemset, así como la rutina de cálculo. Esto permite factorizar esta información, de manera que los requerimientos de almacenamiento por elemento son mínimos. Además, con esta organización el programador que escribe la aplicación puede acceder a dos niveles de programación para la rutina elemental: una dentro del lazo sobre los elementos y otra fuera del lazo. De esta manera, se puede extraer al máximo posible todas aquellas operaciones que son idénticas para todo el elemset.

ABSTRACT

A general purpose finite element program, named PETSc-FEM [3], running on distributed processing architectures using the PETSc/MPI library has been developed. In [2] we describe the existing applications, whereas in [5] the computational efficiency of the program running on a Beowulf class cluster built at CIMEC [1] is discussed. Programming details of the computational code (applications and library) are discussed in this work. The most important underlying object is a set of similar elements or "elemset". It is assumed that most of the unerlying physical and numerical information about the elements in an elemset is the same, so that the amount of storage required is kept to a minimum. Furthermore, the application writer has access to two levels of programming at the elemental routine level, inside and outside the element loop, so that all computations common to all the elements in the elemset can be factored out.

INTRODUCCIÓN

Usualmente los paquetes de elementos finitos constan de una serie de módulos para diferentes aplicaciones en las cuales el usuario puede variar una serie de parámetros o funciones físicos o numéricos. Para tratar de hacer estos programas lo más generales posible se incluyen una gran cantidad de términos en las ecuaciones de gobierno lo que los torna ineficientes. Esta filosofía es limitada y nos llevó a la idea de desarrollar una librería lo más abstracta posible en cuanto a la física subyacente con la cual otros programadores puedan escribir aplicaciones (*"application writers"*).
PETSc-FEM, entonces, es a la vez una librería (`libpetscfem.a`) y un conjunto de aplicaciones, actualmente: Navier-Stokes (versión fractional-step y SUPG-PSPG de Tezduyar et.al. [4], Euler, shallow-water, advección y Ec. de Laplace. Las aplicaciones constan fundamentalmente de dos módulos, la *"rutina del elemento"* y el programa principal. En la rutina del elemento el programador define la relación entre vector de estado y residuo o matrices a nivel elemental mientras que en el programa principal se define la estrategia general de resolución (dependiente del tiempo/estacionaria, lineal/no-lineal, etc...). La librería se encarga de realizar la localización de vectores, ensamblaje de residuos y matrices, fijaciones de condiciones de contorno, etc... en forma generalizada, es decir, para todas las aplicaciones.
El diseño de la librería PETSc-FEM está orientado a usuarios que desean desarrollar nuevas aplicaciones con modelos complejos, más que a brindar una gama completa de aplicaciones. Distinguimos entonces tres niveles de interacción con PETSc-FEM (ver figura 1)

- el *"usuario"*, que usa las aplicaciones sin modificar el código,
- el programador que escribe aplicaciones (*"application writer"*), típicamente escribiendo un programa principal usando llamadas a la librería y rutinas de elemento que son llamadas por la librería, y
- *"programadores del núcleo"* de PETSc-FEM que escriben la librería.

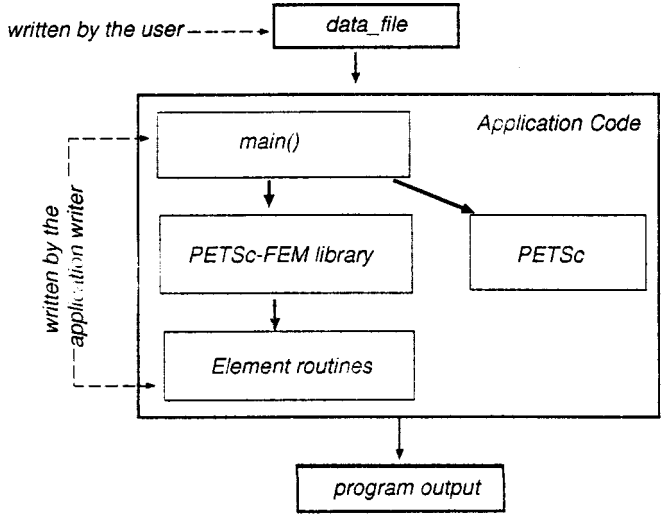

Figura 1: Niveles de interacción de PETSc-FEM

EL CONCEPTO DE "ELEMSET"

PETSc-FEM ha sido escrito en C++ siguiendo la filosofía general de propagación conocida como *"Programación Orientada a Objetos"* (OOP). En OOP, los datos se almacenan en *"objetos"* y el usuario accede a ellos a través de una interface abstracta. Las ventajas de la OOP es, entre otras, una *"compartamentalización"* del código de manera que es relativamente fácil actualizar o cambiar la implementación de ciertas capas de código, manteniendo inalterada la interface. Una primera aproximación a la OOP en un programa de FEM es crear clases para objetos como elementos o nodos. Sin embargo esto puede ser muy ineficien te en cuanto a requerimientos de RAM y CPU, ya que el acceso a cada elemento/nodo se hace pasando a través de toda la capa de software que define el elemento/nodo. La filosofía adoptada al desarrollar PETSc-FEM está orientada a usar la OOP para mantener un código lo más eficiente posible pero con buenas posibilidades de mantenimiento y reuso del código, de manera que hemos resuelto este problema definiendo como entidad básica un conjunto de elementos o *"elemset"*, los cuales se asume que comparten la mayor parte de propiedades físicas (p.ej. viscosidad, densidad, parámetros del modelo de turbulencia) y numéricas (número de puntos de Gauss, tipo de estabilización numérica). Esta es una situación muy comń en CFD donde para cada problema todos los elementos comparten las mismas propiedades.
Todos los residuos o matrices de los elementos de un dado elemset son calculados (esto es sus vectores residuos o matrices) por una misma rutina. De hecho esto se hace usando el polimorfismo, es decir que las diferentes clases de elemsets son obtenidas derivando de una clase virtual genérica `elemset`. En CFD, a diferencia de la mecánica de estructuras, no es usual, en general, mezclar elementos de diferente tipo. Sin embargo, hemos encontrado útil usar el concepto de diferentes tipo de elemento o elemsets para definir términos como condiciones de contorno o aplicar diferentes modelos en diferentes partes del dominio. Además el uso de diferentes elemsets del mismo tipo puede ser útil para reducir la memoria requerida cuando un propiedad es compartida por un conjunto de elementos, esto es, en ese caso la propiedad es almacenada una sóla vez para todo el elemset.

LA LIBRERÍA "PETSCFEM"

A continuación mencionaremos algunas de las posibilidades de la librería:

Preprocesamiento Posee etapas de preprocesamiento interno y externo del archivo de datos, con sustitución de variables, y operaciones aritméticas arbitrarias.

Condiciones de contorno tipo restricciones Las condiciones de contorno tipo Dirichlet generales (como de deslizamiento, periódicas, etc...) son implementadas como restricciones lineales y que se pueden utilizar recursivamente.

Condiciones de contorno tipo Neumann o mixtas Condiciones de contorno absorbentes pueden definirse via elemsets como operaciones de proyección. Las de tipo Neumann (flujos distribuidos) o mixtas (intercambio convectivo), son implementadas a través de elementos de contorno especiales. Condiciones de contorno dependientes del tiempo, pueden definirse a partir de una librería existente o el usuario puede agregar funciones adicionales.

Elementos advectivos Para problemas advectivos, existe una rutina general la cual puede ser usada para cualquier tipo de sistema con solo cambiar la función de flujo. Actualmente existen funciones de flujo (y por lo tanto aplicaciones) para las ecuaciones de Euler (dinámica de gases, flujo compresible), ecuaciones de escurrimiento de aguas poco profundas (ecs. de *"shallow water"*), y sistemas de ecuaciones de advección escalar.

Jacobianos numéricos Posibilidad de calcular jacobianos numéricos. Si bien esta es una opción muy costosa (en tiempo de CPU) para ser usada en grandes aplicaciones, es de interés

para pequeños problemas, para verificar jacobianos analíticos, para elemsets cuyo tamaño (en número de elementos) es pequeño.

Datos de elementos y parámetros generales Los datos de elementos (físicos y numéricos), así como parámetros generales del problema son pasados via arreglos asociativos (*"hashes"*) de strings. Esto permite una gran versatilidad en el paso de parámetros a las rutinas de elementos, funciones temporales para la definición de condiciones de contorno dependientes del tiempo, etc... Además como toda la información que se ingresa via el archivo de entrada de datos, estos soportan una gran capacidad de preprocesamiento via *ePerl*.

Balance de carga: Nuestra experiencia con clusters Beowulf nos indica que si bien es altamente deseable mantener la homogeneidad del cluster en cuanto a tipo de procesador (Intel en nuestro caso) y SO (GNU/Linux), es un hecho que debemos coexistir con procesadores de diferente generación o velocidad. Por ejemplo actualmente en nuestro cluster coexisten un P II 350Mhz, un P III 450Mhz y 5 P III 500Mhz. Esto se debe a que a medida que se agregan nodos al cluster es muy costoso actualizar los procesadores existentes, por ejemplo actualizar un P III 450Mhz a P III 500Mhz puede costar un 25% o más del valor del procesador, mientras que la diferencia en velocidad de procesamiento es sólo del 10%.

Ahora bien si la carga se distribuye en forma igual entre todos los procesadores, los más rapidos deberán esperar al más lento de todos, con lo cual el rendimiento del cluster será equivalente a como si todos los procesadores fueran igual al más lento. En nuestro caso, por ejemplo, podemos tomar los siguientes valores

1 procesador P II 350Mhz $\rightarrow$ 28 Mflops/proc

1 procesador P III 450Mhz $\rightarrow$ 38 Mflops/proc

5 procesadores P III 500Mhz $\rightarrow$ 40 Mflops/proc

De manera que la potencia total teórica (sin tener en cuenta la comunicación) es de 266 Mflops. Sin embargo, si la tarea se distribuye equitativamente la potencia total teórica es de $7 \times 28 = 196$ Mflops. Nótese que en este caso conviene descartar el P II 350Mhz, ya que en ese caso se cuenta contaría con una potencia total teórica de 228 Mflops. Sin embargo, particionando (con METIS) la malla de forma de distribuir la carga en forma proporcional a la velocidad del procesador se llega a una potencia total teórica igual a la suma de las velocidades individuales, es decir de 266 $Mflops$. La potencia total efectiva (es decir, incluyendo comunicación) demostrada en la resolución de un gran sistema proveniente del programa de Navier-Stokes y reportada por PETSc es de 230 Mflops aprox.

SOFTWARE UTILIZADO

PETSc: En general MPI no es llamado directamente sino a través de PETSc (versión 2.0.4) *"Parallel Extensible Toolkit for Scientific Computations"* que es un paquete orientado a métodos numéricos en procesamiento distribuido, y permite operaciones abstractas como definir vectores y matrices distribuidos y resolver los sistemas lineales asociados.

METIS: Este particionador de malla permite dividir el *"grafo dual"* (es decir áquel donde los elementos son vértices del grafo y los nodos son aristas que conectan los vértices) de conectividades en subdominios tratando de mantener la masa total de los vértices (esfuerzo computacional en computar los elementos) igual entre los diferentes subdominios manteniendo mínima la comunicación entre procesadores (la cual se define asignando un peso a las aristas del grafo dual). En el caso de realizar *"balance de carga"* la masa total de los vértices en cada procesador debe ser proporcional a la velocidad del procesador. (`http://www.cs.umn.edu/~metis`, `http://www.cs.umn.edu/~karypis/metis`).

MPICH: Librería de paso de mensajes MPI (*"Message Passing Interface"*, (`http://www.mcs.anl.gov/mpi`, `http://www.mcs.anl.gov/mpich`).

BLAS, LAPACK: Paquetes estándar de álgebra distribuidos por Netlib (`http://www.netlib.org/lapack/`).

egcs: Compilador GNU gcc, g++, g77 (actualmente versión 2.91.66, release 1.1.2-12).

Libretto, C++-STL: Contenedores abstractos para C (`http://pobox.com/~aaronc/tech/libretto/`) y la C++ STL Template Library que viene con el compilados *egcs*.

Newmat: Librería de matrices usada para cálculo a nivel de la rutina elemental (`http://webnz.com/robert/`)

FastMat: Una librería alternativa de matrices para cálculo a nivel de la rutina elemental con cache de direcciones. (Desarrollada en el CIMEC e incluida en la distribución de PETSc-FEM).

Perl/ePerl: Usado como preprocesador, (`http://www.perl.com/`, `http://www.engelschall.com/sw/eperl/`).

CONCLUSIONES

PETSc-FEM es una librería escrita con el objetivo de poder desarrollar aplicaciones usando el método de elementos finitos, aprovechando las posibilidades de cálculo que se obtienen usando procesamiento distribuido via MPI, por ejemplo en clusters de PC's de tipo Beowulf.

Agradecimientos
Este trabajo fue desarrollado gracias a subsidios de CONICET, ANPCyT y UNL a través de los proyectos CONICET-PIP-198/98 *"Germen-CFD*, SECyT-FONCyT-PICT-51 *"Germen"* and CAI+D-UNL-94-004-024. Se ha hecho uso intensivo de software libre como las librerías MPICH y PETSc, Libretto y Newmat, SO *GNU/Linux*, *Octave*, *Xfig*, *Tgif*, compiladores EGCS *gcc*, *g++*, *g77* y muchos otros.

REFERENCIAS

[1] Centro Internacional de Métodos Computacionales en Ingeniería. `http://venus.arcride.edu.ar/CIMEC`.

[2] **N. Nigro, M. Storti, A. Yommi, and V. Sonzogni**. Finite element parallel computations on a beowulf cluster. CFD applications. A ser presentado en ENIEF'2000.

[3] **M. Storti and N. Nigro**. Programa de Elementos Finitos de uso General para Uso en Redes de Cálculo Distribuido basado en PETSc. `http://minerva.arcride.edu.ar/petscfem`.

[4] **T. Tezduyar, S. Mittal, S. Ray, and Shih R.**. Incompressible flow computations with stabilized bilinear and linear equal order interpolation velocity-pressure elements. *Comp. Meth. Applied Mechanics and Engineering*, 95, 1992.

[5] **A. Yommi, N. Nigro, M. Storti, and V. Sonzogni**. Análisis de eficiencia de un programa de elementos finitos en un cluster Beowulf. A ser presentado en ENIEF'2000.