Versión inicial de una biblioteca para la enseñanza de álgebra lineal numérica

Gastón Simone, Pablo Ezzatti

Instituto de Computación
Facultad de Ingeniería, Universidad de la República
J. Herrera y Reissig 565, Montevideo, Uruguay
gaston.simone@gmail.com
pezzatti@fing.edu.uy

Junio 2008

Palabras claves: Álgebra Lineal Numérica, C, Algoritmos.

Resumen

La resolución de una gran cantidad de modelos presentes en la computación científica se basan en la solución de problemas del álgebra lineal numérica (ALN). Esta situación ha motivado fuertemente el desarrollo del área. En contraposición al importante desarrollo se ha incrementado en forma abrupta la complejidad de las estrategias de resolución utilizadas, dificultando la comprensión por parte de los alumnos de los algoritmos utilizados.

En este contexto la propuesta se centra en el desarrollo de una biblioteca de ALN de carácter didáctico. Por esta razón, la documentación es vasta y el código fue escrito pensando en su fácil lectura. Las rutinas implementadas son eficientes desde el punto de vista algorítmico. Pero determinadas mejoras de desempeño, propias de la implementación, fueron descartadas para mantener la legibilidad del código.

Este documento presenta el diseño, las funcionalidades y la implementación realizada. Se describe también la documentación técnica que acompaña a la implementación, la cual es de utilidad para el estudio. Finalmente se mencionan conclusiones y se describen posibles trabajos futuros sobre la herramienta construida.

1. Introducción

La resolución de una gran cantidad de problemas presentes en la computación científica, en particular aquellos que utilizan modelados matemáticos para simular procesos físicos, necesitan como parte fundamental en cuanto a tiempos de cómputo la resolución de problemas de álgebra lineal numérica (ALN). Esta realidad ha motivado a que muchos grupos de investigación a nivel mundial centren su trabajo en el desarrollo del área. Esta situación se puede verificar observando por un lado la gran cantidad de bibliotecas desarrolladas en los últimos años y por otro constatando la importante disminución en los tiempos de ejecución necesarios para la resolución de grandes sistemas lineales.

En contraposición con el gran desarrollo en cuanto a los niveles de eficiencia en los métodos de resolución logrados se ha incrementado en forma notoria la complejidad de los algoritmos. Entre otras cosas se ha perdido en la legibilidad de los programas, en lo intuitivo de las técnicas, etc.

Estas dificultades en la comprensión de los algoritmos utilizados en el área obstaculizan el acercamiento por parte de los alumnos de grado a las distintas problemáticas y las estrategias de resolución. Teniendo como consecuencias directas, por un lado la disminución de profesionales con capacidades en los temas de ALN y por otro lado el alejamiento de los jóvenes investigadores en épocas tempranas del área.

Este trabajo presenta una propuesta de biblioteca de ALN, la cual busca facilitar la comprensión de los algoritmos involucrados en el tratamiento de matrices dispersas. La intensión es que la biblioteca posea un carácter didáctico, resignando en muchas ocasiones la eficiencia en los algoritmos para priorizar la facilidad de comprensión.

Los requisitos planeados en el párrafo anterior condicionaron en la etapa de diseño que la arquitectura de la biblioteca permitiera diversas extensiones. Previendo en un futuro cercano poder trabajar con otros formatos de matrices dispersas, incluir nuevos métodos o incluir estrategias de programación en paralelo. También se tomó especial énfasis en la documentación de la biblioteca.

Este documento describe la primera versión de la biblioteca, que se denomina BAL (por el -acrónimo- Biblioteca de Álgebra Lineal). También se presentan posibles usos de la biblioteca en la enseñanza del álgebra lineal numérica. Es importante establecer que el trabajo es el puntapié inicial de un proyecto más ambicioso. Esperando desarrollar más material que sirva para potenciar el trabajo de enseñanza e investigación en temas referentes al ALN.

El documento está estructurado de la siguiente forma. La sección 2 brinda un detalle de las funcionalidades actualmente disponibles en BAL. La sección 3 brinda una reseña de la estructura interna de BAL, cómo fueron agrupadas las distintas funcionalidades dentro de la biblioteca. En la sección 4 se describen componentes adicionales que acompañan la implementación de la biblioteca. La sección 5 describe los pasos necesarios para utilizar BAL. Luego, la sección menciona algunas ideas para el uso de BAL como una herramienta didáctica. Finalmente, la sección 7 menciona las conclusiones sobre el trabajo realizado y describe el trabajo futuro a partir de la situación actual.

2. Funcionalidades actuales de la biblioteca

A continuación se detallan las distintas funcionalidades disponibles actualmente en BAL.

1. Cargas de datos desde archivo
   a) Leer definiciones de matrices dispersas en el mismo formato utilizado por MatLab [10].
   b) Leer definiciones de matrices en un formato para matrices dispersas por coordenadas.
2. Formatos de matrices dispersas manejados [1]
   a) Simple por coordenadas.
   b) Empaquetado por columnas (CSC).
   c) Comprimido por diagonales (CDS).
3. Impresión de estructuras de datos
   a) Impresión de detalle de estructuras en memoria, para todos los formatos de matrices dispersas soportados.
   b) Impresión de matrices en memoria a formato matlab, para formatos CSC y CDS.
   c) Impresión de matrices en memoria en formato simple al mismo formato esperado en 1*b*.
4. Funciones de conversión entre formatos dispersos
5. Funciones de soporte
   a) Liberación de memoria utilizada en estructuras creadas con BAL.
6. Algoritmos
   a) Recorrer por filas una matriz empaquetada por columnas en forma eficiente.
   b) Producto matriz-vector optimizado para matrices simétricas en formato CSC.
   c) Producto matriz-vector optimizado para matrices en formato CDS.
   d) Producto matriz-matriz optimizado para matrices en formato CDS.
   e) Cálculo del árbol de eliminación [**?**] de una matriz en formato CSC.
   f) Factorización simbólica de una matriz en formato CSC.
   g) Factorización de Cholesky de una matriz en formato CSC.
   h) Resolución de un sistema lineal triangular, representado con una matriz en formato CSC.

La lista completa de las funcionalidades se pueden obtener en la documentación técnica [9].

3. Estructura

A continuación se menciona cómo están distribuidas las distintas funcionalidades de BAL dentro de sus archivos fuente. Conjuntamente se brindan algunos detalles técnicos de la implementación de las distintas funcionalidades.

El directorio raíz de BAL contiene (además de archivos de soporte como el `makefile` y el archivo de configuración de la documentación autogenerada) los módulos llamados `bal`, `oper` y `utils`. El módulo `bal` define un medio de acceso a todas las funcionalidades de BAL de forma homogénea. De este modo, un programa cliente que desee utilizar BAL solo debe agregar la instrucción de preprocesador

```
#include <bal.h>
```

y ya estará en condiciones de utilizar todas las funcionalidades de BAL. Por otra parte, el módulo `utils` contiene funciones de utilidad general que son utilizadas a lo largo de todo el código de BAL. El módulo `oper` implementa los algoritmos de operaciones básicas, del tipo matriz-vector y matriz-matriz.

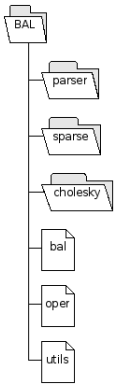

Figura 1: Estructura de directorios de BAL

El directorio `parser` de BAL contiene la implementación del parser de definición de matrices en formato matlab [10]. Este parser es implementado mediante código autogenerado utilizando las herramientas `GNU bison` [5] y `flex` [7]. El archivo `matriz_scanner.lex` contiene la definición del analizador lexicográfico generado por `flex`, mientras que el archivo `matriz_parser.y` define una gramática libre de contexto, junto con el código necesario para que `bison` genere el parser buscado.

El directorio `sparse` contiene las definiciones de las estructuras de datos para matrices dispersas, así como también la implementación de las funciones de conversión entre estas representaciones. El módulo `sp_coord` define la estructura simple por coordenadas, así como también implementa las herramientas básicas para trabajar con la estructura. De la misma forma, los módulos `sp_packcol` y `sp_cds` lo hacen para los formatos CSC y CDS.

Finalmente, el directorio `cholesky` contiene la implementación de las distintas rutinas que permiten la resolución de un sistema de ecuaciones mediante la factorización de Cholesky [3]. Esta resolución se entiende como la ejecución de las siguientes etapas [4]:

1. Reordenamiento.
2. Cálculo del árbol de eliminación.
3. Factorización simbólica.
4. Factorización numérica.
5. Resolución de sistemas triangulares.

El módulo `cholesky` implementa todas las etapas de la resolución, menos la etapa de reordenamiento, para la cual aun no se cuenta con la implementación de un algoritmo. Pero la biblioteca ya dispone de la estructura necesaria como para agregar tal etapa fácilmente. Esto se hará mediante la extensión del módulo `reordenamiento`.

La gran mayoría de la implementación de este método de resolución está basado en el paper de Stewart [8].

4. Componentes adicionales

4.1. Documentación técnica

La biblioteca viene acompañada de un conjunto de documentación técnica que describe los detalles de las funcionalidades implementadas, describe los algoritmos y explica las estructuras de datos definidas.

Esta documentación fue creada con la herramienta Doxygen [12]. La misma es generada en los formatos HTML y PDF, siendo el primero el recomendado debido a la facilidad de navegación que presenta. La misma documentación técnica explica cómo ésta es generada.

4.2. Juego de pruebas

Junto con la implementación de la biblioteca, se escribió un conjunto de programas que sirven no solo para verificar la correctitud de BAL, sino también como ejemplo de su uso.

Estos tests se encuentran en el directorio `bal_test`. Cada prueba es un programa `C` diferente que se limita a probar una funcionalidad particular de BAL. El nombre del archivo `.c` da idea de lo que se pretende probar.

El juego de pruebas tiene su propio `makefile` para la fácil compilación. Pero más aun, el directorio de pruebas contiene un script llamado `run_tests.sh` para la ejecución fácil de las pruebas. Este script, primero ejecuta el comando `make` para cerciorar que los tests han sido compilados. Luego busca aquellos tests que fueron compilados correctamente y los ejecuta. Para cada test ejecutado, redirecciόna su salida estándar a un archivo de extensión `.out` y su salida de errores a una archivo de extensión `.error`. El prefijo de ambos archivos es el nombre del ejecutable del test[1].

Al finalizar, el script muestra en pantalla una lista de aquellos tests que generaron algún tipo de salida de errores. De este modo es fácil localizar cuáles tests fallaron.

5. Uso de la biblioteca

La biblioteca fue construida y probada bajo un ambiente UNIX utilizando el lenguaje `C`. Sin embargo, dada la portabilidad del lenguaje, no debería ser difícil conseguir que la misma funcione en otras plataformas.

[1] Nota: Para que este mecanismo funcione es importante que todos los ejecutables de las pruebas tengan sufijo `_test`.

La biblioteca se construye como un paquete precompilado. Una vez construida, un programa cliente puede utilizarla enlazando dicho paquete junto con su código, en tiempo de compilación. El archivo `makefile` principal contiene una secuencia de pasos para instalar la biblioteca de forma permanente en el sistema. De este modo se facilita su utilización. Este modo de instalación también genera una copia de la documentación técnica dentro del sistema, para que esté disponible para todo usuario del mismo junto con la biblioteca.

La documentación técnica detalla todos los pasos necesarios para crear una aplicación que utilice la biblioteca, incluyendo compilación e instalación de la biblioteca, así como también compilación de un programa externo que haga uso de la misma.

6. Utilización en la enseñanza del Álgebra Lineal Numérica

Como ya fue mencionado, uno de los objetivos de este trabajo fue construir una herramienta que sirva para el estudio del álgebra lineal numérica. La biblioteca construida es el primer paso en este cometido. Con la versión 1.0.0 de BAL se cuenta con suficientes herramientas como para complementar el material teórico y práctico de algunos temas tratados durante un curso básico de ALN, como por ejemplo formatos de matrices dispersas, algoritmos eficientes con matrices dispersas, algoritmos de reordenamiento, factorización simbólica y factorización numérica.

Se pueden construir aplicaciones basadas en BAL que permitan realizar ejemplos durante una clase. Se pueden realizar comparaciones de performance entre BAL y otras herramientas, como por ejemplo matlab. Los alumnos pueden hacer sus propias pruebas con la biblioteca e incluso extenderla. También se puede tomar la extensión de la biblioteca como trabajo importante en la comprensión de los algoritmos.

También es importante destacar que la documentación técnica que acompaña la implementación aporta el soporte necesario para el estudio de los algoritmos implementados y las estructuras de datos utilizados. La forma en la que es presentada la documentación junto con el código al que corresponde cada sección beneficia su lectura.

7. Conclusiones y trabajo futuro

La biblioteca es un trabajo en progreso. Esta versión fue un paso inicial en la construcción de una herramienta de enseñanza, de práctica y una fuente de conocimiento en la rama del álgebra lineal numérica. La disponibilidad de un conjunto de implementaciones fuertemente documentadas de distintos algoritmos básicos y otros más complejos es de gran valor para toda persona relacionada con la materia. El material desarrollado se puede obtener en [11].

El trabajo futuro, en lo que refiere a la extensión de la biblioteca, tiene varios aspectos a ser considerados, a saber:

- Se puede agregar soporte para más plataformas, en particular la plataforma Windows. Dado que gran parte de las instalaciones disponibles en facultad proveen esta plataforma, esto facilitaría el uso de la herramienta por parte de los alumnos.
- La biblioteca puede ser extendida para que determinados algoritmos produzcan salida que muestre la evolución de la ejecución de los mismos. Esto puede ser de gran ayuda a la hora de estudiar un comprender como trabaja un algoritmo. Sin embargo, esto debe ser hecho con cuidado, pues agregar tal tipo de funcionalidad, puede perjudicar en la legibilidad del código.
- El manejo de errores dentro del código de la biblioteca puede ser mejorado. Algo de trabajo en este aspecto ya está hecho en el módulo de utilidades. Sin embargo, se puede mejorar la robustez de la biblioteca frente a errores graves, por ejemplo manejando señales del sistema. La biblioteca debería ser capaz de brindar información de utilidad frente a cualquier tipo de error, para el posible estudio posterior del problema.
- Para completar la secuencia de resolución eficiente de un sistema lineal basado en la factorización de Cholesky, es necesario contar con al menos la implementación de un algoritmo de reordenamiento. Como ya fue mencionado, la estructura necesaria para agregar este algoritmo en la biblioteca ya fue construida.
- Se puede extender el soporte a más formatos para matrices dispersas.
- Al soportar más formatos de matrices dispersas, será necesario contar con más funciones de conversión entre formatos.
- Puede ser interesante extender el parser de definición de matrices en formato matlab para que cargue las matrices en memoria ya en un formato disperso, por ejemplo, simple por coordenadas.
- Agregar soporte para más algoritmos de factorización y factorización incompleta, como LU.
- Agregar soporte para técnicas de precondicionamiento de matrices.
- Agregar soporte para algoritmos iterativos.

Por supuesto, también es un trabajo futuro el estudiar y explorar los métodos para utilizar la nueva herramienta con el objetivo de mejorar la enseñanza de la materia y explotarla de la mejor forma posible.

Referencias

[1] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia, 2000

[2] T. A. Davis. *Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms 2)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, SIAM 2006, ISBN=0898716136.

[3] I. Duff, A. Erisman and J. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, New York, 1986.

[4] A. George and J. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice Hall, Englewood Cliffs, NJ, 1981.

[5] GNU Project. *Bison*. http://www.gnu.org/software/bison/. Consultado julio 2008.

[6] J. Liu. *The role of elimination trees in sparse factorization*. SIAM J. Matrix Anal. Appl. 11:134-172, 1990.

[7] V. Paxson. *Flex*. http://flex.sourceforge.net/. Consultado julio 2008.

[8] G.W. Stewart. *Building an Old-Fashioned Sparse Solver*. Univerity of Maryland, Institute of Advanced Computer Studies, Department of Computer Science. TR-2003-95, TR-4527. Agosto 2003. Disponible en: http://hdl.handle.net/1903/1312

[9] G. Simone and P. Ezzatti. *Documentación técnica de BAL 1.0.0*. Disponible en: http://www.fing.edu.uy/inco/cursos/numerico/aln/bal/bal.pdf. Consultado julio 2008.

[10] *Sitio web de MATLAB*. Disponible en: www.mathworks.com. Consultado julio 2008.

[11] Sitio de BAL en Internet. Disponible en: http://www.fing.edu.uy/inco/cursos/numerico/aln/bal/. Consultado julio 2008.

[12] D. Van Heesch. *Doxygen*. http://www.doxygen.org/. Consultado julio 2008.