CNEA-NT 15/78

COMISION NACIONAL DE LA ENERGIA ATOMICA
DEPENDIENTE DE LA PRESIDENCIA DE LA NACION

CODIGO DE COMPUTACION PARA LA RESOLUCION DIRECTA DE SISTEMAS DE ECUACIONES ALGEBRAICAS LINEALES CON MATRIZ ESPARCIDA, SIMETRICA Y DEFINIDA POSITIVA: CODIGO SPFACT

Sergio Pissanetzky

Centro Atómico Bariloche, 8400 - San Carlos de Bariloche - Argentina

Buenos Aires
Mayo de 1978

INDICE

RESUMEN

Una variedad de métodos numéricos utilizados en la práctica conduce a la generación de grandes sistemas de ecuaciones algebraicas lineales. En muchos casos la matriz del sistema resulta esparcida o sea que la mayor parte de sus elementos son iguales a cero. En tales casos se deben emplear procedimientos de cómputos especiales para almacenar sólo los no-ceros en la memoria de la computadora, para realizar operaciones sólo con los no-ceros, y para reducir la generación de nuevos no-ceros durante los cálculos tanto como sea posible. Así se asegura un empleo eficiente del tiempo y memoria de computadora disponibles.

Durante los últimos años los algoritmos empleados para cálculo con matrices esparcidas se han vuelto muy sofisticados, y los cómputos se pueden realizar ahora de una manera muy densa y eficiente. Aquí presentamos un paquete de subrutinas en Fortran que incorpora los últimos adelantos en lo que ha empezado a llamarse la Tecnología de Matrices Esparcidas, para resolver grandes sistemas de ecuaciones lineales con matriz simétrica y definida positiva, del tipo que aparece en el método de Elementos Finitos.-

ABSTRACT

Large systems of algebraic linear equations are generated in practice by a variety of numerical methods. In many cases the matrix of the system is sparse, that is, most of its elements are equal to zero. Special computational procedures have to be used in such cases in order to store only the non-zeroes, to operate only with the non-zeroes, and to reduce the generation of new non-zeroes during the calculations as much as possible. In this way an efficient use of the available computer memory and time is secured.

During the last few years the algorithms for sparse matrix calculations have become very sophisticated, and the computation can now be performed in a very dense and efficient way. We present here a Fortran package of subroutines which incorporates the last advances of what has become to be known as the Sparse Matrix Technology, to solve large sparse systems of linear equations with symmetric and positive definite matrix, of the type that is generated by the Finite Element method.-

INTRODUCCION:

Una matriz se considera esparcida cuando la mayor parte de sus elementos son nulos. En los últimos 3 ó 4 años se ha producido un avance muy considerable en las técnicas de computación empleadas para trabajar con matrices esparcidas. Los objetivos esenciales de todos los esfuerzos han sido siempre los siguientes:

i) Hallar algoritmos que realicen operaciones aritméticas o lógicas solamente con aquellos elementos de las matrices que se sabe que son diferentes de cero.

ii) Realizar los cálculos de tal manera que la generación, durante el proceso, de nuevos elementos diferentes de cero, resulte tan reducida como sea posible, a efectos de que las matrices no dejen de ser esparcidas.

El desarrollo tecnológico genera problemas cada vez más grandes y más complejos, cuya solución demanda cálculos numéricos cada vez más largos. La respuesta a esta demanda ha sido, por un lado, el desarrollo de computadoras más rápidas y con mayor memoria, y por el otro, de técnicas de computación más económicas. La Técnica de Matrices Esparcidas (ME) pretende alcanzar el más alto grado de economía computacional que se pueda esperar una vez planteado un problema numérico, pues reduce hasta donde se pueda la cantidad de operaciones y el llenado de la memoria necesarios para terminar el cálculo.

Los sistemas de ecuaciones algebraicas lineales con matriz esparcida, simétrica y definida positiva aparecen frecuentemente en cálculo numérico. El método de Elementos Finitos (1) y varios otros métodos de resolución de ecuaciones diferenciales originan sistemas de este tipo. La importancia de implementar un código que resuelva tales sistemas se puede comprender, para no ir más lejos, recordando algunas de las aplicaciones del método de Elementos Finitos a problemas de Energía Nuclear (2), Petróleo (3), Cohetería (4), Guías de onda (5), Vibraciones (5), Aerodinámica, Lubricación, Mecánica Estructural, Cálculos Térmicos, y hasta Cálculos Atómicos (6,18). Cualquiera de estas aplicaciones puede llevar a un sistema con mil o varios miles

```
IA = 1 4 5 8
JA = 1 4 6 2 1 3 5                    R(C)FO
AN = 3 1 2 5 1 4 1
```

Obsérvese que IA es de largo N+1, y que IA(N+1) señala el lugar siguiente al último elemento de JA y AN.

Por ejemplo, la fila 2 comienza en IA(2)=4. ¿Donde termina la fila 2? Pues, como la fila 3 comienza en IA(3)=5, la fila 2 termina en IA(3)-1=4. O sea, la fila 2 está descripta en el lugar 4 de JA y AN. Tiene un elemento que vale AN(4)=5 cuyo índice de columna es JA(4)=2. En este caso IA, JA y AN definen una Representación Completa por Filas y Ordenada de la matriz A, que se abrevia R(C)FO.

En realidad, no hay necesidad de representar ordenadamente los elementos de cada fila de A: el Formato Esparcido permite representarlos en desorden, por ejemplo:

```
IA = 1 4 5 8
JA = 6 1 4 2 3 5 1      R(C)FD
AN = 2 3 1 5 4 1 1
```

Esta es una Representación Completa por Filas Desordenada ó R(C)FD. Veremos más adelante que la exigencia de tener las representaciones ordenadas resulta algorítmicamente muy gravosa. Por eso, los algoritmos de ME utilizan representaciones desordenadas siempre que sea posible. El descubrimiento de esta notable propiedad significó uno de los mayores avances en la técnica de ME(9).

Cuando la matriz considerada es simétrica se puede representar sólo los elementos diagonales y los de la mitad superior. Por ejemplo:

$$B = \begin{vmatrix} 3 & 0 & 0 & 1 \\ 0 & 5 & 0 & 1 \\ 0 & 0 & 7 & 0 \\ 1 & 1 & 0 & 2 \end{vmatrix}$$

```
IB = 1 3 5 6 7
JB = 4 1 2 4 3 4              R(SD)FD
BN = 1 3 5 1 7 2
```

Esto es una Representación Superior y Diagonal por Filas Desordenada , o R(SD)FD. Si la matriz, además de ser simétrica, es definida positiva, entonces todos sus elementos diagonales son nó-nulos, y se los puede almacenar en un vector BD en vez de describirlos en JB y BN. Así se obtiene una Representación Superior.

Por ejemplo, para la matriz B:

```
IB = 1 2 3 3 3
JB = 4 4                R(S)FD
BN = 1 1
BD = 3 5 7 2
```

Esta representación es muy compacta y eficiente. Si se emplea Fortran los vectores IB y JB pueden ser declarados INTEGER*2 para obtener una economía de memoria adicional, siempre y cuando no haya más de 32767 nó-ceros en la parte de la matriz que se describe.

La eliminación de Gauss por filas

Consideramos el sistema de ecuaciones lineales

$$Ax = b \qquad (1)$$

donde A es una matriz simétrica definida positiva de orden N y b es un vector columna denso. La eliminación de Gauss por filas consiste en restar de la segunda ecuación un múltiplo apropiado de la primera para lograr que el elemento a_{21} se reduzca a cero. Luego se restan de la tercera ecuación múltiplos apropiados de la primera y de la segunda para lograr que $a_{31}=0$ y $a_{32}=0$. Y así siguiendo, hasta eliminar todos los elementos por debajo de la diagonal. El orden de eliminación se elige por filas porque es más apropiado para aplicarlo a las ME representadas por filas.

Una vez concluída la eliminación, A se ha reducido a una matriz triangular superior. Si además dividimos cada ecuación por su elemento diagonal, obtenemos la matriz triangular superior unitaria U cuyos elementos diagonales son todos 1, y la matriz diagonal D formada por los elementos diagonales que se

usaron como divisores. Se verifica la propiedad (10):

$$A = U^T D U \qquad (2)$$

donde T significa "traspuesta".

El sistema (1) se puede escribir

$$U^T D U x = b \qquad (3)$$

Para obtener x basta con resolver sucesivamente los tres sistemas lineales siguientes, dos de los cuales tienen matriz triangular y el restante la tiene diagonal, por lo que son muy fáciles de resolver:

$$\begin{aligned} U^T z &= b \\ Dy &= z \\ Ux &= y \end{aligned} \qquad (4)$$

Este procedimiento permite separar la solución de (1) en dos partes: la factorización de A para obtener D y U, y la solución de (4). Como D y U dependen sólo de A, se obtiene así una ventaja que puede ser considerable: cuando hay que resolver varios sistemas con la misma matriz A y diferentes segundos miembros b, se puede factorizar A una sóla vez, y luego resolver las (4) para los diferentes b.

La eliminación de Gauss común, bien conocida, consiste en realizar sobre b las mismas operaciones que se realizan sobre A, simultáneamente. Se puede demostrar que este procedimiento requiere exactamente el mismo número de operaciones aritméticas que la factorización y solución por separado, para un solo vector b dado. Actualmente se prefiere usar la factorización y solución por separado: ya hemos señalado una de las ventajas que ofrece, y más adelante señalaremos otra aún más importante

Para factorizar una matriz cualquiera es necesario elegir como pivote el elemento de mayor valor absoluto o por lo menos uno cuyo valor absoluto sea mayor que una cierta "tolerancia de pivote" (8), para evitar que los errores de redondeo se hagan excesivos. En una matriz simétrica definida positiva, los elementos diagonales son grandes y se pueden elegir siempre como pivotes (10, pág.220). Falta aún especificar en qué orden se los elige como pivotes. En todo lo que sigue vamos a suponer

que la matriz A ya está ordenada de tal manera que se pueda pivotar secuencialmente a lo largo de la diagonal. Más adelante explicaremos con que criterio se la ordena, permutando entre sí sus filas y sus columnas sin perder la simetría. Por el momento baste con decir que este criterio no perjudica los requerimientos de la tolerancia de pivote (8).

La factorización de una matriz esparcida simétrica

Consideremos como ejemplo la siguiente matriz, de la cual indicamos sólo la estructura pero nó los valores de sus elementos nó-nulos:

$A =$

	1	2	3	4	5	6	7	8	9	10	11
1	X	0	0	0	0	X	0	0	0	X	0
2	0	X	0	X	0	0	0	0	X	0	0
3	0	0	X	0	X	0	X	0	0	0	X
4	0	X	0	X	0	0	X	0	0	0	0
5	0	0	X	0	X	0	0	X	0	0	0
6	X	0	0	0	0	X	0	0	X	0	0
7	0	0	X	X	0	0	X	0	0	0	X
8	0	0	0	0	X	0	0	X	0	0	X
9	0	X	0	0	0	X	0	0	X	0	X
10	X	0	0	0	0	0	0	0	0	X	X
11	0	0	X	0	0	0	X	X	X	X	X

Para factorizarla debemos comenzar con la cuarta fila, pues las filas anteriores no tienen nó-ceros a la izquierda de la diagonal. De la cuarta fila restamos la segunda, elemento por elemento, multiplicada por una constante tal que el elemento a_{42} quede nulo. Al hacerlo se introduce un nuevo nó-cero en la fila 4, el a_{49}. Seguidamente se resta la tercera fila de la quinta para eliminar a_{53}, introduciendo nuevos nó-ceros en a_{57} y $a_{5,11}$. Luego se elimina a_{61}, introduciendo un nó-cero en $a_{6,10}$. Seguidamente debemos proceder a aliminar los elementos a_{73} y a_{74} de la fila 7, y al hacerlo vamos a aprovechar para tocar más a fondo las dificultades algorítmicas que se presentan

aún en este sencillo ejemplo, y también los detalles para generar la descripción de las matrices U y D dada por los vectores IU, JU, UN y DI.

En primer lugar sólo el triángulo superior de A está representado en la computadora. Los elementos a_{73} y a_{74} no existen en la memoria, y no hay información que diga que dichos elementos son diferentes de cero y que deben ser eliminados restando de la septima fila múltiplos de la tercera y de la cuarta. Tampoco sabemos que al restar la fila 3 se producirá un nuevo no-cero en a_{75}, que también tendrá que ser eliminado restando la fila 5.

Lo que sí sabemos es que los simétricos de a_{73} y a_{74}, o sea a_{37} y a_{47}, son nó-nulos. Y también sabemos que el simétrico de a_{75}, o sea a_{57}, ha dejado de ser nulo durante el proceso de la fila 5. En otras palabras, debemos examinar la columna 7 por arriba de la diagonal para saber cuáles son los nó-ceros de la fila 7 a la izquierda de la diagonal que debemos eliminar.

En segundo lugar, sabemos que los elementos de la fila 7 a la izquierda de la diagonal tomarán el valor 0, así que no necesitamos calcularlos. En otras palabras, basta con calcular los elementos a_{7j} con $j \geq 7$, y para calcularlos sólo se necesitan aquellos elementos de las filas 3, 4 y 5 que estén sobre la columna 7 o a su derecha. Esto parece requerir una representación ordenada a efectos de poder determinar fácilmente cuáles son los elementos de cada fila cuyo índice de columnas es 7 ó más.

Supongamos entonces haber realizado los pasos necesarios para ubicar los elementos que necesitamos. Tendremos así los siguientes índices de columna:

fila 3 : 7 11
fila 4 : 7 9
fila 5 : 7 8 11
fila 7 : 7 11

Una vez terminado el cálculo la nueva fila 7 tendrá los índices de columna siguientes, que formarán parte del vector JU en la descripción de la matriz U:

nueva fila 7 : 7 8 9 11

Además, los valores de los nuevos elementos a_{77}, a_{78}, a_{79} y $a_{7,11}$ tendrán que ser calculados realizando las operaciones aritméticas necesarias, y quedarán almacenados en el vector UN (la inversa de a_{77} queda en DI (7)).

Por su parte el vector IU se genera contando cuántos elementos existen en la fila 7 a la derecha de la diagonal, en este caso 3:

IU (8) = IU (7) + 3

De esta manera natural, el proceso de factorización ha quedado separado en dos partes, a saber: la factorización simbólica, que es la generación de los vectores IU y JU que describen la estructura de U, y la factorización numérica, que es la generación de los vectores UN y DI que completan la descripción de U y la de D.

No existe razón para no separar totalmente estos dos pasos. Se puede hacer una pasada sobre A calculando sólo los índices de columna de cada fila, y almacenándolos en JU, y sus señaladores en DI. Y luego hacer otra pasada sobre A calculando numéricamente UN y DI. Veremos enseguida que esta separación, a primera vista extraña, trae ventajas de tal magnitud que ha sido universalmente adoptada. Estas son las ideas de Chang (11) y Gustavson (12).

Si sólo queremos calcular la estructura de la fila 7 podemos simplemente reunir las listas asociadas a las filas 3,4, 5 y 7, en cualquier orden. Antes de inscribir en la lista final un número proveniente de alguna de las listas, debemos preguntar si ese número ha sido inscripto antes. Esto se hace utilizando un vector auxiliar entero llamado "indicador expandido de nó-ceros" (9): cada vez que un número es inscripto en la lista final, el elemento correspondiente del indicador expandido de nó-ceros toma el valor I, donde I es el índice de la fila que se está procesando, en este ejemplo la 7(es la "técnica de llave múltiple", ref. 13). Antes de inscribir un nuevo número en la lista final, se pregunta si el indicador expandido de

nó-ceros contiene el valor I en el lugar correspondiente, y así se evita inscribir dos veces el mismo número. La selección del valor I como indicador es para evitar poner a cero todo el vector antes de procesar cada fila.

El proceso que acabamos de describir se llama "reunión de listas esparcidas desordenadas de enteros", y es de $O(N_c)$, donde N_c es la cantidad de nó-ceros finales, es decir que por cada nó-cero final requiere una cantidad fija de operaciones elementales. Claramente, tal cosa no se puede hacer si lo que se desea es generar aritméticamente la fila 7, ya que no bastaría con saber si un cierto elemento está inscripto o nó en la lista: habría que saber además dónde está dicho elemento para poder procesarlo aritméticamente. Esto es un proceso de $O(N_c^2)$ porque requiere generar las listas ordenadamente (14).

Hay además un teorema (15) que establece que, en el proceso de reunión de listas que estamos aplicando, basta considerar aquellas filas cuyo nó-ceros en la columna procesada sea el primero de la fila a la derecha de la diagonal. Volviendo al ejemplo, podemos limitarnos a considerar las filas 4 y 5, además de la 7, descartando la fila 3, cuyo primer nó-cero está en la columna 5. La demostración del teorema se basa simplemente en que, al ser $a_{35} \neq 0$, por simetría $a_{53} \neq 0$; al eliminar a_{53}, se resta la fila 3 de la 5, y todos los nó-ceros de la fila 3 se repiten en la fila 5. En este caso, $a_{5,11}$ es generado por $a_{3,11}$. Por lo tanto basta considerar la fila 5, que contiene, además de los nó-ceros propios, todos los que estaban en la fila 3. El proceso de reunión de listas esparcidas desordenadas se realiza entonces con las filas 7, 4 y 5 como sigue:

```
fila 7                      : 7 11
fila 7 + fila 4             : 7 11 9
fila 7 + fila 4 + fila 5: 7 11 9 8
```

Hemos obtenido correctamente la estructura de la fila 7, pero está desordenada. Este proceso lo realiza nuestra subrutina SYMFAK. No habríamos ganado nada si no dispusiéramos de un procedimiento para ordenar la estructura completa del factor triangular superior U en $O(N_c)$ operaciones y utilizando $O(U_c)$

memoria. Tal procedimiento es la clasificación lexicográfica tipo "cucharón" (16), y lo realiza nuestra subrutina SP26A. La clasificación lexicográfica de la estructura de una matriz esparcida dada en R(S)FD tiene el efecto de generar una Descripción Superior por Columnas Ordenada ó R(S)CO de la misma matriz. Este curioso efecto ha sido descubierto recientemente y puede usarse también para trasponer y permutar matrices esparcidas en $O(N_c)$ operaciones (16). Nuestra SP26A provee la R(S)CO de U, simbólicamente, en IUT y JUT.

Una segunda aplicación de SP26A vuelve a trasponer la matriz U y deja IU y JU en R(S)FO.

Como la generación de la estructura por SYMFAC requiere $O(N_c)$ operaciones y memoria, y cada aplicación de SP26A también lleva $O(N_c)$ operaciones y memoria, tenemos que hemos obtenido la estructura ordenada del factor superior U en $O(N_c)$ operaciones y memoria. Gracias a muchos teoremas y procedimientos descubiertos en los últimos dos años, estos cálculos se realizan hoy de manera muy densa y con una eficiencia sorprendente. En la práctica, en SYMFAC, una vez generada la estructura de cada fila I, se busca cuál es el nó-cero de índice más bajo, sea J, y se inscribe la fila I en el conjunto asociado con la columna J. Más adelante, cuando se procesa la fila J, el conjunto J dice que la fila I tiene un nó-cero sobre la columna J y que por lo tantp la fila I debe ser reunida con la fila J(17). Estos conjuntos se manejan en SYMFAC en el vector entero IP de largo N.

Ahora estamos preparados para calcular UN y DI, completando así la descripción de las matrices U y D.

La factorización numérica

Disponemos de la estructura ordenada de la matriz U dada en IU y JU. Volviendo a nuestro ejemplo, queremos calcular los elcmentos nó-nulos de la fila 7 dc U, y sabemos que la fila 7 tiene nó-ceros en los lugares 7, 8, 9 y 11. Examinando la columna 7, vemos que debemos sumar a la fila 7 las filas 3, 4 y 5

multiplicadas por constantes apropiadas, pero solamente aquellos elementos de cada fila cuyo índice de columna sea 7 o más. Para encontrar dichos elementos necesitamos un señalador móvil (12) que en nuestra subrutina NUMFAK se maneja en el vector entero IUP de largo N. Este señalador indica, para cada fila, en qué lugar de JU y UN comienza la descripción ordenada de aquella parte de la fila que debe ser sumada. Por ejemplo, al procesar la fila 7, IUP (3) indicaría la posición en UN del elemento a_{37}, IUP(4) la de a_{47} y IUP(5) la de a_{57}. Este indicador móvil IUP se actualiza cada vez que una parte de una fila ha sido procesada. Por ejemplo, al terminar de sumar, a la fila 7, los elementos a_{37} y $a_{3,11}$ de la fila 3, en IUP(3) se suma 1, con lo cual IUP(3) pasa automáticamente a señalar la posición en UN del elemento $a_{3,11}$, que es el siguiente nó-cero de la fila 3 después de a_{37}. Claramente, la actualización de IUP requiere una representación ordenada de las filas.

La información relativa a los nó-ceros de cada columna se maneja en N conjuntos, asociados biunívocamente con las columnas (17). Por ejemplo, al procesar la fila 7, el conjunto 7 contiene los números 3, 4 y 5, indicando que las filas 3, 4 y 5 deben ser sumadas a la fila 7. Inmediatamente que cada parte de fila es sumada, se inscribe la fila en el conjunto correspondiente a su próximo nó-cero. Por ejemplo, al terminar con la fila 3, se la inscribe en el conjunto de la columna 11. El conjunto J queda completo sólo inmediatamente antes de procesar la fila J. En NUMFAK todos los N conjuntos se manejan de manera extremadamente densa y eficiente, en la forma de listas circulares encadenadas desordenadas de números enteros, en un solo vector IP de largo N.

En la práctica se trabaja sobre un vector real X de largo N, llamado de "reunión expandida de elementos". Al procesar una fila, se anulan solamente aquéllos elementos de X que se sabe que serán nó-ceros. En nuestro ejemplo se anulan X(7), X(8), X(9) y X(11). Luego se carga la fila 7 en X, o sea los elementos a_{77} y $a_{7,11}$, y después se van recorriendo las demás filas y se van sumando sus elementos, multiplicados por los

factores apropiados, en los lugares de X que les corresponden.

Finalmente, los nó-ceros calculados en X se cargan en UN y DI. Todo el proceso es de $O(N_c)$ en operaciones y de $O(N,N_c)$ en memoria. Como $N_c > N$, y como la factorización simbólica y ordenamiento de la estructura por SYMFAK y SP26A también son $O(N_c)$, podemos concluir que hemos logrado la factorización completa de la matriz A en $O(N_c)$ operaciones y memoria.

La resolución del sistema lineal

Contamos ya con una R(S)FO de la matriz U dada en IU, JU y UN, y con la matriz D dada en DI. Resolvemos los sistemas (4), calculando sucesivamente z, y, x utilizando los procedimientos usuales (12). Cada uno de los sistemas (4) se resuelve en $O(N_c)$ operaciones y memoria. En conjunto hemos logrado resolver el sistema (1) en $O(N_c)$ operaciones y memoria para cada vector b dado.

Nuestra subrutina NUMBKS resuelve los sistemas (4).

En el caso de tener que resolver varios sistemas (1) con diferentes segundos miembros b pero con la misma matriz A, es suficiente con factorizar A una sola vez, y luego se utiliza NUMFAC para cada uno de los vectores b dados.

Descripción de las rutinas

La factorización completa de la matriz A se hace por medio de la subrutina SPFACT, que se utiliza como sigue:

CALL SPFACT(IA,JA,AN,AD,IU,JU,UN,DI,IP, IUP, IUT,JUT,N,M)

donde:

IA,JA,AN,AD son vectores de entrada que contienen la descripción de la matriz dada A en R(S)FD. IA es de largo N+1. JA y AN son de largo igual a la cantidad de nó-ceros en el triángulo superior de A. AD es de largo N.

IU,JU,UN son vectores de salida que contienen la descripción de la matriz calculada U en R(S)FO. IU es de largo N+1. El largo

de JU, y el de UN, es igual a la cantidad de elementos nó-nulos en el triángulo superior de U.,
DI es un vector de salida de largo N que contiene las inversas de los elementos de la matriz D.
IP, IUP son vectores de trabajo de largo N cada uno.
IUT, JUT son vectores de trabajo. IUT es de largo N+1 y JUT es de largo igual al de JU. En la salida los vectores IUT y JUT contienen la descripción simbólica de U en R(S)CO. Si esta información no es necesaria, se puede economizar memoria declarando en el programa principal lo siguiente:

```
EQUIVALENCE (IUT(1),DI(1)),(JUT(1),UN(1)).
```

N es un parámetro de entrada que contiene el orden de las matrices A y U.
M es un parámetro de entrada que contiene el largo de memoria reservado para cada uno de los vectores JU y UN. En el caso de que durante la generación de JU en la subrutina SYMFAK, dicho largo de memoria sea excedido, SYMFAK emite un mensaje y se detiene.
Todas las variables enteras son declaradas INTEGER*2.

La resolución del sistema (1) para cada vector de segundos miembros b la realiza la subrutina NUMBKS que se usa como sigue:

```
CALL NUMBKS (IU,JU,UN,DI,B,X,N,NM)
```

dónde:
IU,JU,UN son vectores de entrada que contienen la descripción del factor triangular superior unitario U en R(S)FO. IU es de largo N+1. JU y UN son, cada uno, de largo igual a la cantidad de nó-ceros en el triángulo superior de U.
DI es un vector de entrada de largo N que contiene las inversas de los elementos de la matriz D.
B es un vector de entrada de largo N que contiene los segundos miembros del sistema (1).
X es un vector de salida de largo N que contiene los valores calculados de las incógnitas.
N es un parámetro de entrada que contiene el orden de las matrices U y D.
NM es un parámetro de entrada que contiene el valor N-1.

La generación de nuevos nó-ceros

Durante la factorización de una matriz A se van introduciendo nuevos elementos nó-nulos en el triángulo superior de la matriz. La diferencia entre la cantidad de nó-ceros en el triángulo superior de U y en el triángulo superior de A se denomina llenado. El llenado de una matriz A, y también la cantidad de operaciones necesarias para factorizarla, dependen drásticamente del orden en que se realiza la factorización. En otras palabras, en vez de resolver (1) podemos resolver:

$$(PAP^T)(Px) = Pb. \qquad (5)$$

donde P es una matriz de permutación y PAP^T continúa siendo simétrica. Podemos seleccionar P de tal manera que el llenado o la cantidad de operaciones sean mínimos. El problema de determinar P es demasiado extenso para ser tratado aquí, y nuestra intención es orientar al lector hacia la literatura más importante.

El procedimiento de la dirección anidada (19) es muy apropiado para las matrices esparcidas que se obtienen en el método de Elementos Finitos o en el de Diferencias Finitas. En estos casos basta con numerar los nodos del problema para definir la permutación de la matriz A. El procedimeinto es lo suficientemente sencillo para poder hacerlo a mano si el problema no es demasiado grande. El llenado y la cantidad de operaciones resultan del orden de N log N para problemas en dos dimensiones, donde N es la cantidad de nodos, en lugar del orden N^2 usual. En tres dimensiones resultan $O(N^{4/3})$ y en cuatro dimensiones $O(N^{3/2})$. Existe un teorema que prueba que estos órdenes de magnitud son asintóticamente óptimos.

Otro algoritmo eficiente (15) se utiliza para matrices esparcidas generales, empleando un código que genera la permutación P automáticamente.

Pruebas numéricas

El código que aquí se describe se ha utilizado para resolver probelmas de Elementos Finitos. Tomamos como ejemplo un problema típico de electrostática que generó un sistema de 700 ecuaciones lineales con 700 incógnitas, cuya matriz A contiene 1924 elementos nó-nulos en su triángulo superior, además de los 700 elementos diagonales.

La matriz U resultó con 8452 elementos en el triángulo superior, además de los 700 diagonales.

Al reordenarse A por el método de disección anidada, la cantidad de nó-ceros de U se redujo a 8190.

En nuestro sistema IBM/360-44 los tiempos de ejecución fueron de 2 seg. para la factorización simbólica, 4 seg. para ordenar la estructura de U, 12 seg. para la factorización numérica y 3 seg. para la resolución final. La memoria de 128 Kbytes fue más que amplia para este problema.

Agradecimientos

El autor desea expresar su agradecimiento al Ing. H. Cingolani y a todo el personal del Centro de Cómputos del Centro Atómico Bariloche por la amplia y amistosa colaboración manifestada durante las etapas de este trabajo.

Referencias

1. O.C. Zienkiewicz. "The Finite Element Method in Engineering Science". McGraw-Hill, London (1971)

2. F.G. Basombrío y G. Sánchez. "El método numérico de elementos finitos y su aplicación a problemas de física del continuo. Desarrollo adquirido y trabajos realizados en el CAB". CNEA-NT 13/77 (1977).

3. S.Pissanetzky. "Modelo de reservorio bidimensional formulado en base al análisis de elementos finitos". Primeras Jornadas Científico-técnicas. Fac. de Ingeniería, Univ. del Zulia, Venezuela (1977).

4. G. Sánchez, P.A. Laura y M.E. Olivetto. "Determination of the unsteady state temperature distribution in a solid propellant rocket motor".Eighth U.S. National Congress of Applied Mechanics, Los Angeles, California (1978).

5. G. Sánchez y M.E. Olivetto. "Resolución de la ecuación de Helmholtz en sistemas de simetría plana arbitraria por el método de elementos finitos". CNEA-NT 4/78 (1978).

6. A. Askar. "Finite Element Method for Bound State Calculations in Quantum Mechanics". J. Chem. Physics, 62,732-734 (1975).

7. P.T. Woo, S.J.Roberts and F.G. Gustavson. "Application of Sparse Matrix Techniques in Reservoir Simulation". Soc. of Petroleum Engineers J. Paper number SPE 4544 (1973).

8. R.P. Tewarson. "Sparse Matrices". Vol.99 in "Mathematics in Science and Engineering". Ed. R. Bellman. Ac. Press. New York (1973).

9. F.G. Gustavson. "An efficient algorithm to perform sparse matrix multiplication". IBM Research Report RC 6176 (#26569) (1976).

10. J.H. Wilkinson. "The Algebraic Eigenvalue Problen". Clarendon Press. Oxford (1965).

11. A. Chang. "Application of Sparse Matrix Methods in Electric Power System Analysis". Sparse Matrix Proceedings. Willoughby, ed. IBM Research Report RA1, Yorktown Heights, N.Y., 113-122 (1968).

12. F.G. Gustavson. "Basic Techniques for Solving Sparse Systems of Linear Equations". Sparse Matrices and their Applications. Eds. Rose and Willoughby. Plenum Press., N.Y., 41-52 (1972).

13. F.G. Gustavson. "Finding the Block Lower Triangular Form of a Sparse Matrix". Sparse Matrix Computations (J. Bunch and D.Rose, ed.) Academic Press Inc. N.Y. (1976)Pp.275-289.

14. S.Pissanetzky. "Implementación de un conjunto de algoritmos que emplean la Tecnología de las Matrices Dispersas para resolver problemas de Análisis de Elementos Finitos". XXVII Jornadas Nacionales de ASOVAC, 6 al 12 de Nov. de 1977. Valencia - Venezuela.

15. D.J.Rose, R.E. Tarjan, G.S. Lueker. "Algorithmic Aspects of Vertex Elimination on Graphs". SIAM Journal on Computing 5, 266-283 (1976).

16. F.G. Gustavson. "Permuting Matrices Stored in Sparse Format". IBM Research RC 6181 (#26575) (1976).

17. S.C. Eisenstat, M.H.Schultz, and A.H. Sherman. "Efficient Implementation of Sparse Symmetric Gaussian Elimination". Adv. in Computer Methods for Partial Differential Equations. R. Vichnevetsky, ed. Publ. AICA (1975).

18. M. Friedmann, Y.Rosenfeld, R.Thieberger. "Finite Element Method for solving the Two-Dimensional Schroedinger Equation". J.of Computational Physics 26. 169-180 (1978).

19. A.George, J.W.H.Liu. "An automatic Nested Dissection Algorithm for Irregular Finite Element Problems". Research Report CS-76-38.Dept.of Computer Science,University of Waterloo, Waterloo, Ontario, Canadá, Octubre 1976.