ESQUEMAS COMPUTACIONALES DE LA FACTORIZACION DE CROUT Y SU INCIDENCIA EN LA RESOLUCION DE SISTEMAS LINEALES Y NO LINEALES SPARSE

M.D. Galera Martínez
Departamento de Informática y Sistemas. Univ. de Las Palmas de G.C.
Avenida Marítima del Sur s/n, 35071 LAS PALMAS DE G.C.
P. Almeida Benítez, P.D. Cuesta Moreno y M.J. Galán Moreno
Departamento de Matemática Aplicada. Univ. de Las Palmas de G.C.
Campus Universitario Tafira, 35194 LAS PALMAS DE G.C.

RESUMEN.- Con este Trabajo se hace un estudio del tratamiento computacional de la resolución de sistemas de ecuaciones lineales o no lineales dispersas o huecas, que surgen al discretizar problemas planteados mediante ecuaciones diferenciales. Para ello utilizamos un método numérico de resolución directa de sistemas: Método de Crout, y desarrollamos varias posibilidades de estructurar computacionalmente dicha factorización. Estas opciones surgen, al realizar de diferentes maneras, el diseño de los esquemas de almacenamiento de los factores L y U de matrices dispersas. Por último, implementamos los esquemas computacionales utilizando como soporte de la matriz del sistema un archivo en disco (virtual o real) para cualquier tipo de sistema operativo y/o procesador.

INTRODUCCION

Es bien conocido que dada una matriz regular A, puede factorizarse como el producto de una matriz triangular inferior L con una matriz triangular superior U, de forma que para resolver el sistema usando A, solo se necesitará hacer una sustitución hacia adelante y otra hacia atrás (vease Burden y otros), además dicha factorización es única, siempre que se fije la diagonal principal de L ó U.

Consideremos el método de Crout, en el que $l_{ii}=1$ para cada "i". Ahora bien la forma LU es "asimétrica" en un sentido: a lo largo de su diagonal principal L lleva siempre unos, mientras U contiene los pivotes. Esto se corrige factorizando U como DU', siendo D una matriz diagonal constituida por los pivotes, y U' la matriz obtenida de U, donde :

$$u'_{ij} = 1, \quad i = j, u'_{ij} = u_{ij}/d_{ii} \quad i \neq j.$$

A partir de ahora y sin que induzca a error denotaremos la nueva matriz U' como U. Escribimos ahora la descomposición triangular de A como: $A=L\circ D\circ U$ donde se tratan imparcialmente la L y la U.

A continuación demostraremos la "existencia y unicidad" de la factorización anteriormente citada, a efectos de mayor comprensión en desarrollos posteriores para los cómputos operacionales, pues en parte se basan en la construcción de la misma.

TEOREMA:

Dada una matriz regular A, admite de forma única una factorización LDU, salvo permutaciones.

Demostramos en primer lugar la existencia por inducción sobre el orden de la matriz A. El resultado es evidente para las matrices uno por uno:

$$(a_{11})=(1)\ (d_{11})\ (1)=(d_{11})$$

Supongamos que la afirmación es cierta para matrices de orden n-1. Sea A una matriz regular de orden n, se puede particionar de la forma:

$$A = \begin{pmatrix} d & u^t \\ v & \overline{H} \end{pmatrix} \qquad (1)$$

donde d es un escalar no nulo, y H una submatriz de orden n-1, no necesariamente regular. La matriz particionada se puede escribir como el producto:

$$A = \begin{pmatrix} 1 & 0 \\ v/d & I_{n-1} \end{pmatrix} \begin{pmatrix} d & 0 \\ 0 & H \end{pmatrix} \begin{pmatrix} 1 & u^t/d \\ 0 & I_{n-1} \end{pmatrix} \quad (2)$$

donde $H = H - vu^t/d$. Obviamente la matriz H es regular; en efecto, ya que por ser A regular, para cualquier x de orden n-1, se verifica:

$(-x^tv/d \quad x^t)A = 0 \Rightarrow (-x^tv/d \quad x^t)=0$

de lo que se deduce que $x^t=0$.

Por otra parte:

$$(-x^tv/d \quad x^t)A = (-x^tv/d \quad x^t)\begin{pmatrix} d & u^t \\ v & \bar{H} \end{pmatrix} =$$

$$= (0 \quad -x^tvu^t/d+x^tH) = x^t(0 \quad H-vu^t/d) =$$

$$= x^t(0 \quad H) = 0 \Leftrightarrow x^tH = 0$$

lo que implica por ser A regular que:

$$x^t=0$$

Por la suposición de inducción, H tiene una factorización "LHDHUH". Así, A se puede expresar como:

$$A = \begin{pmatrix} 1 & 0 \\ v/d & I_{n-1} \end{pmatrix} \begin{pmatrix} d & 0 \\ 0 & H \end{pmatrix} \begin{pmatrix} 1 & u^t/d \\ 0 & I_{n-1} \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 0 \\ v/d & I_{n-1} \end{pmatrix} \begin{pmatrix} d & 0 \\ 0 & L_HD_HU_H \end{pmatrix} \begin{pmatrix} 1 & u^t/d \\ 0 & I_{n-1} \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 0 \\ v/d & I \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & L_H \end{pmatrix} \begin{pmatrix} d & 0 \\ 0 & D_H \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & U_H \end{pmatrix} \begin{pmatrix} 1 & u^t/d \\ 0 & I \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 0 \\ v/d & L_H \end{pmatrix} \begin{pmatrix} d & 0 \\ 0 & D_H \end{pmatrix} \begin{pmatrix} 1 & u^t/d \\ 0 & U_H \end{pmatrix} = L\ D\ U \quad (3)$$

Demostremos ahora la unicidad de los factores L, D y U:

$$\text{Sea} \begin{cases} A=L_1D_1U_1 \\ A=L_2D_2U_2 \end{cases} \Rightarrow L_1D_1U_1=L_2D_2U_2 \quad (4)$$

Multiplicando en (4), por L_1^{-1}:

$D_1U_1 = L_1^{-1}(L_2D_2U_2)$ (5)

Multiplicando en (5), por U_2^{-1}:

$D_1U_1U_2^{-1} = L_1^{-1}L_2D_2$ (6)

Por último, multiplicando en (6), por D_1^{-1}:

$U_1U_2^{-1} = D_1^{-1}L_1^{-1}L_2D_2 = \Delta$ (7)

Donde Δ es la matriz identidad, ya que el primer miembro es una triangular superior con diagonal unitaria y el segundo miembro, una triangular inferior. Obtenemos los siguientes resultados a partir de (7):

$$U_1U_2^{-1} = I_n \Rightarrow U_1 = U_2$$

$$D_1^{-1}L_1^{-1}L_2D_2 = I_n \Rightarrow L_1^{-1}L_2 = D_1D_2^{-1} = \Delta$$

$$\Rightarrow \begin{cases} L_1^{-1}L_2 = I_n \Rightarrow L_2 = L_1 \\ D_1\ D_2^{-1} = I_n \Rightarrow D_1 = D_2 \end{cases} \quad (8)$$

Con lo que hemos demostrado que lafactorización "LDU" está determinada de manera única por A.

EL COMPUTO DE LA FACTORIZACION

En esta sección, examinaremos algunas de las diferentes formas en que L y U pueden ser calculadas; estas opciones son importantes porque nos proporcionan flexibilidad en el diseño de los esquemas almacenamiento para los factores L y U de matrices dispersas.

1. Esquema del producto externo

La prueba constructiva del teorema anterior sugiere un esquema computacional para determinar los factores L, D y U. Describimos este esquema paso a paso en términos matriciales de la forma siguiente:

$$A = A_0 = H_0 = \begin{pmatrix} d_1 & u_1^t \\ v_1 & \bar{H}_1 \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 0 \\ v_1/d_1 & I_{n-1} \end{pmatrix} \begin{pmatrix} d_1 & 0 \\ 0 & H_1 \end{pmatrix} \begin{pmatrix} 1 & u_1^t/d_1 \\ 0 & I_{n-1} \end{pmatrix} =$$

$$L_1 \begin{pmatrix} d_1 & 0 \\ 0 & H_1 \end{pmatrix} U_1 = L_1\ A_1\ U_1 \quad (1.1)$$

donde: $H_1 = H_1 - v_1u_1^t/d_1$. Repetimos el proceso para A_1:

$$A_1 = \begin{pmatrix} d_1 & 0 \\ 0 & H_1 \end{pmatrix} = \begin{pmatrix} d_1 & 0 & 0 \\ 0 & d_2 & u_2^t \\ 0 & v_2 & \bar{H}_2 \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & v_2/d_2 & I \end{pmatrix} \begin{pmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & H_2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & u_2^t/d_2 \\ 0 & 0 & I_{n-2} \end{pmatrix} =$$

$$= L_2\ A_2\ U_2$$

donde $H_2 = \bar{H}_2 - v_2u_2^t/d_2$.

Repitiendo el proceso se tiene:

$A_{n-1} = L_n A_n U_n$, donde $A_n = D$.
Aquí, para $1 \le i \le n$, d_i es un escalar no nulo, v_i y u_i son vectores de tamaño $n-i$ y H_i es una matriz regular de tamaño $n-i$ por $n-i$. Después de n pasos del algoritmo se tiene:

$$A = L_1A_1U_1 = L_1(L_2A_2U_2)U_1 =$$
$$= L_1L_2(L_3A_3U_3)U_2U_1 =$$
$$= L_1L_2 \ldots L_nDU_n \ldots U_2U_1 = L_n D\, U.$$

LEMA:

Dadas dos matrices E y F subtriangulares de orden n, tales que para algún k, satisfacen:

$$e_{jj} = 1, \text{ para } j > k;$$
$$e_{ij} = 0, \text{ para } i > j \text{ e } i > k;$$
$$f_{jj} = 1, \text{ para } j \le k;$$
$$f_{ij} = 0, \text{ para } i > j \text{ y } j \le k;$$

en estas condiciones se verifica que:

$$E \cdot F = E + F - I.$$

Para la demostración de este lema se consideran los cinco casos siguientes:

$$j \le k,\ j > k,\ k \ge i > j,$$
$$i > k \ge j,\ i > j > k;$$

y en todos ellos demostramos que:

$$(EF)_{ij} = (E)_{ij} + (F)_{ij} - \delta_{ij},$$

es decir, EF=E+F-I; sin más que tener en cuenta el producto de matrices y las condiciones impuestas a dichas matrices (Almeida (1989) tesis doctoral pp 30-32).

Análogamente es válido este lema para las matrices triangulares superiores E' y F' que verifiquen:

$$e'_{jj} = 1 \text{ para } j > k;$$
$$e'_{ij} = 0 \text{ para } i < j \text{ y } j > k;$$
$$f'_{jj} = 1 \text{ para } j \le k;$$
$$f'_{ij} = 0 \text{ para } i < j \text{ e } i \le k;$$

PROPOSICION:

i) El producto $L_1L_2 \ldots L_n$ que origina la matriz L en el algoritmo, es igual a: $L_1+L_2+\ldots+L_n-(n-1)I_n$.

ii) El producto $U_nU_{n-1}\ldots U_2U_1$ que origina la matriz U en el algoritmo, es igual a:

$$U_n+U_{n-1}+\ldots+U_2+U_1 - (n-1)I_n.$$

Demostración:

i) Las matrices E y F utilizadas en el lema anterior, son de la forma:

$$E = \begin{pmatrix} E_k & 0 \\ 0 & I_{n-k} \end{pmatrix}, \quad F = \begin{pmatrix} I_k & 0 \\ 0 & F_{n-k} \end{pmatrix}$$

o bien:

$$E = \begin{pmatrix} * & 0 \\ 0 & I_{n-k} \end{pmatrix}, \quad F = \begin{pmatrix} I_k & 0 \\ 0 & * \end{pmatrix}$$

En el desarrollo del algoritmo anterior mediante el producto externo, se tiene:

$$L_1 = \begin{pmatrix} * & 0 \\ * & I_{n-1} \end{pmatrix}, \quad L_2 = \begin{pmatrix} I_1 & 0 & 0 \\ 0 & * & 0 \\ 0 & * & I_{n-2} \end{pmatrix},$$

$$L_3 = \begin{pmatrix} I_2 & 0 & 0 \\ 0 & * & 0 \\ 0 & * & I_{n-3} \end{pmatrix}, \ldots, \quad L_n = \begin{pmatrix} I_{n-1} & 0 \\ 0 & * \end{pmatrix}$$

ii) Análogamente les ocurre a las matrices $U_nU_{n-1}\ldots U_1$:

$$U_1 = \begin{pmatrix} * & * \\ 0 & I_{n-1} \end{pmatrix}, \quad U_2 = \begin{pmatrix} I_1 & 0 & 0 \\ 0 & * & * \\ 0 & 0 & I_{n-2} \end{pmatrix}, \quad U_n = \begin{pmatrix} I_{n-1} & 0 \\ 0 & * \end{pmatrix};$$

por tanto verifican las condiciones de las matrices E' o F', siendo éstas de la forma:

$$E' = \begin{pmatrix} * & 0 \\ 0 & I_{n-k} \end{pmatrix}, \quad \text{ó} \quad F' = \begin{pmatrix} I_k & 0 \\ 0 & * \end{pmatrix}$$

luego las matrices L_i y U_i utilizadas en nuestro desarrollo verifican las condiciones del lema anteriormente enunciado, y aplicando reiteradamente dicho lema a los productos $L_1L_2\ldots L_n$, y $U_n\ldots U_2U_1$, se tiene:

i) $L_1L_2\ldots L_n = L_1 + L_2L_3\ldots L_n - I_n =$
$= \ldots = L_1+L_2+\ldots+L_n - (n-1)I_n$
o lo que es igual:
$L = L_1+L_2-I_n+L_3-I_n+\ldots+L_n-I_n$

ii) $U_n\ldots U_2U_1 = U_n + U_{n-1}\ldots U_1 - I_n$
$= \ldots = U_n +\ldots+ U_2 + U_1 - (n-1)I_n =$
$= U_n + U_{n-1} - I_n +\ldots+ U_1 - I_n$

COROLARIO:

i) La i-ésima columna de L es la i-ésima columna de L_i.
ii) La i-ésima fila de U es la i-ésima fila de U_i.

Según este esquema, tanto las columnas de L como las filas de U se calculan una a una; se trata de una implementación columna a columna y fila a fila respectivamente. Al mismo tiempo, cada paso supone la modificación de la submatriz H_i por el producto externo v_iu_i/d_i para dar H_i, que es la submatriz que queda por factorizar El acceso a las entradas de A durante la secuencia de la computación se representa en la figura (1.a). Por zona que ya no se abordará mas, queremos decir que las entradas que figuran en ella no serán usadas nunca más en las etapas restantes del proceso de

factorización.

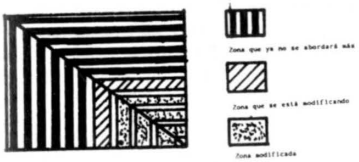

Figura (1.a): Producto externo

Así las fórmulas (2) con $H=\bar{H}-vu^t/d$ y el corolario anterior nos dan las siguientes relaciones reiterativas, cuyos algoritmos desarrollaremos posteriormente.

$a_{ii} \neq 0$

$l_{ki} = a_{ki}/a_{ii} \quad i < k \leq n$

$l_{kj} = a_{kj} - l_{ki} u_{ij}/a_{ii} \quad i < j < k \leq n$

$u_{ij} = a_{ij}/a_{ii} \quad i < j \leq n$

$u_{kj} = a_{kj} - l_{ki} u_{ij}/a_{ii} \quad i < k < j \leq n$

$d_{kk} = a_{kk} - a_{ki} a_{ik}/a_{ii} \quad i < k \leq n$

2. Esquema del orlado

Una formulación alternativa para el proceso de factorización es el esquema del orlado. Supongamos la matriz A particionada como sigue:

$$A = \begin{pmatrix} M & u \\ v^t & s \end{pmatrix},$$

donde ya se ha obtenido, la factorización: $L_M D_M U_M$, de la submatriz principal M de tamaño (n - 1) puesto que siempre se puede encontrar una submatriz M regular, con las permutaciones convenientes. Entonces la factorización de A viene dada por :

$$\begin{pmatrix} L_M D_M U_M & u \\ v^t & s \end{pmatrix} = \begin{pmatrix} L_M & 0 \\ z^t & 1 \end{pmatrix} \begin{pmatrix} D_M & 0 \\ 0 & t \end{pmatrix} \begin{pmatrix} U_M & w \\ 0 & 1 \end{pmatrix}$$

siendo: $u = (L \cdot D)w \Rightarrow w = (L \cdot D)^{-1}u$

$$v^t = z^t(D \cdot U) \Rightarrow z^t = v^t(D \cdot U)^{-1}$$

$$s = z^t Dw + t \Rightarrow t = s - z^t Dw$$

Así, el esquema obtenido es el clásico que resulta de identificar las entradas de A con las de la factorización LDU (véase Tesis Almeida (1.989), pag. 36).

Con este esquema, tanto las filas de L como las columnas de U se calculan una cada vez, se implementa fila a fila y columna a columna respectivamente; la parte de la matriz que aún queda por factorizar no será abordada hasta que las partes correspondientes de L y U se vayan a calcular. La secuencia de la computación se ilustra en la figura (2.b). Indicaremos que una zona está ya calculada cuando lo están las entradas que corresponden a dicha zona, que una zona es abordada cuando se emplean sus entradas en la etapa i de la factorización.

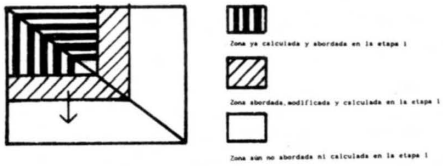

Figura (2.b): Orlado.

3. Esquema del producto interno.

Por último, veamos el esquema del producto interno para determinar las entradas de L y U. Este esquema se basa en la identificación de las entradas de A con aquéllas de L·D·U que ocupan su misma posición. Omitimos su desarrollo, dado que se trata, de un algoritmo bastante conocido (vease Burden (1988) y otros).

Como en la versión del producto externo del algoritmo, tanto las columnas de L como las filas de U se calculan una por una, se trata pues de una factorización columna a columna y fila a fila respectivamente, pero la parte de la matriz que queda sin factorizar como es bien conocido, no se aborda durante el procesamiento del esquema, por ello se denomina interno. L a secuencia de los cálculos se describe en la figura (3.c) Indicaremos que una zona está calculada cuando ya lo están sus en entradas en la etapa j de la factorización, que una

zona es abordada cuando se usan entradas de esta zona para la factorización en la etapa j, y entenderemos por zona modificada aquélla en la que varía el valor de las entradas durante la realización de la etapa j de la factorización.

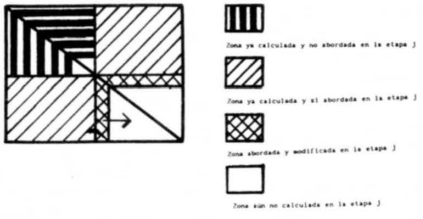

Figura (3.c): Producto interno.

INCIDENCIA EN LA RESOLUCION DE SISTEMAS SPARSE.

Cuando el método de Crout se aplica a una matriz A Sparse, generalmente experimenta rellenos de huecos, de forma que los factores L y U tienen no ceros en posiciones que son ceros en A. Para alguna matriz de permutación P, se puede en su lugar factorizar PAP^t en $\bar{L}.\bar{U}$, y $\bar{L}$ puede ser mucho más atractiva que L, de acuerdo con algún criterio.

El objetivo del estudio de técnicas de matrices dispersas para la resolución de sistemas lineales consiste en reducir costes explotando la dispersión del sistema dado. Es posible obtener reducciones drásticas en el almacenamiento y demandas aritméticas, cuando se comparan las soluciones de sistemas densos con las de los sistemas Sparse.

Existen varios tipos de almacenamientos dispersos, que difieren en las formas en que se explotan los ceros. Algunos pueden almacenar algún cero a cambio de un almacenamiento más simple; otros explotan todos los ceros del sistema.

La elección de un método de almacenamiento, afecta a la demanda de memoria y al empleo de estrategias de ordenamiento (elección de la matriz de permutación P). Además, tiene un impacto en la implementación de la factorización y resolución, y, por tanto, en la complejidad del programa y el tiempo de ejecución.

Sin embargo, con independencia del esquema de almacenamiento disperso que se emplee, hay cuatro fases evidentes que se pueden identificar durante todo el proceso computacional.

Paso 1: Ordenamiento

Encontrar un buen ordenamiento (permutación P) para la matriz dada A, con respecto al método de almacenamiento elegido.

Paso 2: Asignación del almacenamiento

Determinar la información necesaria acerca de los factores L y U de PAP^t para establecer el esquema de almacenamiento.

Paso 3: Factorización

Factorizar la matriz PAP^t en L.U

Paso 4: Resolución triangular

Resolver $Ly = b$ y $Uz = y$. Luego establecer $x = P^t z$.

En efecto:
$Ax = b \Rightarrow PAx = Pb = \bar{b} \Rightarrow PAP^tPx = Pb = \bar{b} \Rightarrow L(L^tPx) = \bar{b}$

Llamando $L^tPx = y$, se tiene $Ly = \bar{b}$. Por otra parte, si hacemos $Px = z$, tenemos $L^tz = y$; finalmente, si $Px = z \Rightarrow x = P^tz$

Incluso con el método de almacenamiento descrito anteriormente, hay muchas formas de encontrar ordenamientos, determinando la estructura del correspondiente almacenamiento de L, y ejecutando la computación numérica real. Nos referiremos al esquema de almacenamiento disperso y a una combinación asociada colectivamente a ordenamiento / asignación / factorización / resolución como un método de resolución.

Los objetivos más comunmente citados para elegir un método de resolución son:

a) reducir el espacio de almacenamiento del ordenador,

b) reducir el tiempo de ejecución,

c) reducir alguna combinación de

almacenamiento y ejecución que refleje la forma en que se tasan las cargas al usuario del sistema de ordenador. Aunque existen otros criterios que a veces se imponen en la elección del método, estos son los principales y muestran las complicaciones que supone la evaluación de una estrategia.

Con el fin de poder afirmar que método es mejor que otro con respecto a una de las medidas citadas anteriormente, tenemos que poder evaluar con precisión la medida para cada método, y esta evaluación es sustancialmente más complicada de lo que se podría esperar. En primer lugar trataremos el criterio de almacenamiento en el ordenador.

1. **Demandas de memoria**

La memoria del ordenador empleada para matrices dispersas consta propiamente de dos partes, memoria principal empleada para retener los valores numéricos, y memoria suplementaria usada para indicadores, subíndices y cualquier otra información necesaria para registrar la estructura de la matriz y facilitar el acceso a los valores numéricos. Toda vez que hemos de pagar por la memoria en el ordenador, independientemente de como sea empleada, cualquier evaluación de la demanda de memoria para un método de resolución ha de incluir una descripción de la forma en que la matriz o matrices implicadas van a ser almacenadas, de manera que la memoria suplementaria se pueda incluir junto con la memoria principal en la demanda de memoria. La comparación de dos estrategias distintas con respecto al criterio de almacenamiento puede implicar basicamente estructuras de datos diferentes, teniendo memorias suplementarias muy distintas. Así, un método que es superior en términos de reducción de memoria principal puede ser inferior cuando la memoria suplementaria se incluye en la comparación. En resumen, los puntos principales son:

1) Los esquemas de almacenamiento para matrices dispersas suponen dos componentes: memoria principal y memoria suplementaria.

2) Las comparaciones de las estrategias de ordenamiento han de tener en cuenta el esquema de almacenamiento a emplear, y la comparación va a ser prácticamente relevante.

ASPECTOS COMPUTACIONALES

Señalamos tres aspectos:

1.- Usamos la matriz de partida como espacio de almacenamiento de los factores L D U. Minimizando el uso de memoria RAM y disco fijo.

2.- Se efectúa la programación en los lenguajes TurboC2.0-Borland en S.O. MS-DOS y en SCO-C en SCO-XENIX, usando en ambos casos como espacio de almacenamiento de la matriz el formato de archivo binario en disco, siendo admisibles las opciones de usar disco virtual o disco fijo.

A este respecto es importante señalar que el uso de disco virtual permite superar la limitación de uso de memoria RAM impuesta por el S.O. MS-DOS, sin perjuicio de la velocidad de cálculo. En lo que se refiere al uso de archivo en disco fijo en XENIX plantea el inconveniente de la disminución de la velocidad del procesamiento aunque presenta la ventaja de no plantear más limitación al tamaño de la matriz que el tamaño del disco. Por ejemplo, una matriz 500x500 en d.p. ocupa 2Mb. lo cual hace que deba ser tratada en MS-DOS en el formato de archivo en disco virtual. Una matriz 2000x2000 (32 Mb.) está fuera de la mayor parte de la RAM de un microordenador, sin embargo, cabe perfectamente en un disco fijo.

3.- El esquema de resolución va dirigido a sistemas que corresponden al planteamiento teórico AX = B , donde A es la matriz del sistema, cuadrada de rango y orden n; B es una matriz n x m que se corresponde a m columnas de términos independientes y X es una matriz n x m formada por m columnas de soluciones.

La resolución final es $X = U^{-1}D^{-1}L^{-1}B$, siendo LDU la factorización de CROUT de la matriz A.

3.1.- El esquema en seudocódigo de la factorización LDU es:

```
Abrir archivo binario para lectura/escritura
Reserva espacio en memoria para dos vectores fila v y u
  Desde i=1 hasta n-1
  Posicionar puntero de archivo en posición i,i
  Leer desde posición i,i hasta final de fila en v
  Si a_ii = 0 ir a función de permutación de filas
    Desde k = i+1 hasta final de vector v
    Hacer v(k) = v(k)/v(i)
  Retroceder el puntero de archivo n - i + 1 lugares
  Escribir en archivo vector v
    Desde j = i+1 hasta n (las filas por debajo de i)
    Puntero en posición j,i
    Leer desde j,i hasta final de fila en vector u
      Desde k = i+1 hasta final vector u
      Hacer u(k) = u(k) - u(i)* v(k)
    Hacer u(i) = u(i)/v(i)
    Retroceder n - i + 1 posiciones el puntero de archivo
      Escribir el vector u
Cerrar archivo
Fin de factorización
Llamar al programa de resolución
```

3.2.- El esquema computacional del producto L^{-1} B utilizado es la triangulación de Gauss en L (inversión al ser L triangular inferior) actuando sobre B.

3.3.- D^{-1} { $L^{-1}B$ }es dividir cada fila i de la matriz { $L^{-1}B$ } por el elemento d_i de D.

3.4.- El producto U^{-1} { D^{-1} { $L^{-1}B$ }} se realiza por el proceso 3.2.

CONCLUSIONES

Los procedimientos mostrados representan una aproximación al problema de la resolución de sistemas de ecuaciones bajo la óptica de los métodos directos; los procesos informáticos utilizados permiten la implementación de grandes sistemas con microordenador usando el S.O. más común en el mercado mediante el uso de archivo en disco virtual y un enlace entre estos dispositivos y los grandes ordenadores a través del S. O. XENIX-UNIX.

REFERENCIAS

George A. (1975) A note on fill for sparse matrices SIAM J. Num. Anal. 12

Almeida P. (1989)- Tesis Doctoral. Las Palmas de G. C., Universidad de Las Palmas de G. C.

Bernard D. , Menegezzi P.(30/5/89). Gazette Leodufel n. 9. INRIA. Paris. Anexo 5, Cours Modulef 1989. "Resolution de Systemes lineaires", Rocquencourt. Paris.

Cuthill E.,McKee J. (1969a). Reducing the bandwidth of sparse symetric matrices. Proc. 24th Nat. Conf. Assoc. Comput. Math., ACM Publ.

Cuthill E., McKee J. (1969b) Reducing the bandwith of sparse symetric matrices. Proc. A. C. M. National Conference New York.

George A., Liu W. H. (1979) An implementation of a pseudoperipherical node finder. A. C. M. Transactions on Math. Soft. Vol 5 .

Liu G. (1981) Computer Solution of Large Sparse Positive Definite Systems. Edit. Prentice Hail. Series in computational mathematics

Golub (1989) Matrix Computations. The Johns Hopkins University Press (pg. 133-136)

Fox L. (1964) An Introduction to Numerical Linear Algebra. Oxford University Press (pg. 131-149).

Winter Althaus G. y Conde Lázaro C.(1990) Métodos y algoritmos básicos del Algebra Numérica. Barcelona, Reverté.

SUMMARY

This communication is a study about computational treatment for resolution of sparse equation systems both linear and non linear.

We use several schemes for the Crout algorithm for matrix factorization.

The computational implementation for these schemes is made using disk file as media storage.